Lecture 19

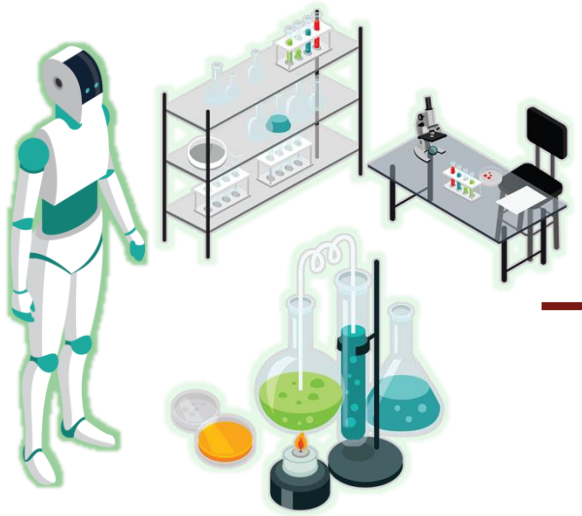
CS 6260 Automated Planning and Learning

Learning PDDL Models

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Learning Action Models from Passive Observations



$\langle s_0, a_1, s_1 \rangle$
$\langle s_1, a_2, s_2 \rangle$
$\vdots$
$\langle s_{n-1}, a_n, s_n \rangle$

[Input]

Learner

```
(:action pick-up-beaker
   :parameters (?x - beaker)
      :precondition (and (ontable ?x)
         (not-in-use ?x)
         (handempty))
      :effect (and (not (ontable ?x))
          (not (handempty))
          (holding ?x)))

(:action put-on-shelf
   :parameters (?x - beaker)
      :precondition (holding ?x)
      :effect (and (not (holding ?x))
          (handempty)
          (onshelf ?x)))
```

[PDDL Example]

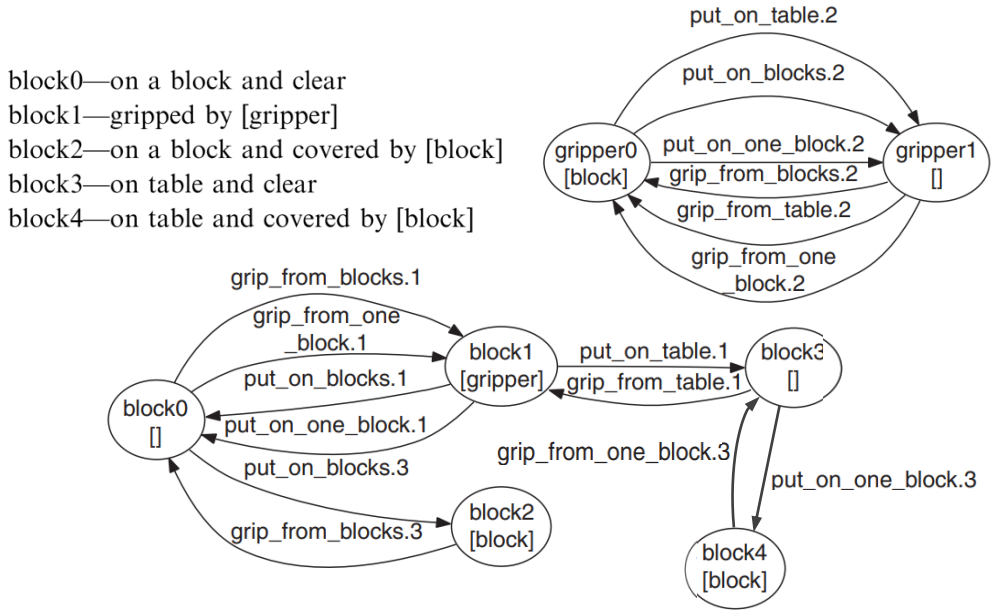
What kind of approaches these learners use?

Finite State Machine based Learners

- For each object type create a finite state machine.
- Create PDDL by combining them.

$\langle s_0, a_1, s_1 \rangle$
$\langle s_1, a_2, s_2 \rangle$
$\vdots$
$\langle s_{n-1}, a_n, s_n \rangle$

[Input]




```
(:action pick-up-beaker
   :parameters (?x - beaker)
      :precondition (and (ontable ?x)
         (not-in-use ?x)
         (handempty))
      :effect (and (not (ontable ?x))
          (not (handempty))
          (holding ?x)))

(:action put-on-shelf
   :parameters (?x - beaker)
      :precondition (holding ?x)
      :effect (and (not (holding ?x))
          (handempty)
          (onshelf ?x)))
```

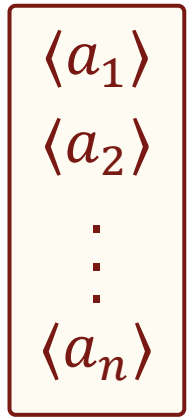
[PDDL Example]

LOCM, LOCM2 - Cresswell et al. (ICAPS'09 , ICAPS'11, Know. Engg. Rev.'13), LOP - Gregory et al. (ICAPS'15), NLOCM - Gregory et al. (ICAPS'16)

Learning Object-Centred Models (LOCM)

- Input: Action Sequence
  - List of action names with object arguments

- What is not part of the input?
  - Any description of states before, during, or after actions
  - Any predicates or fluents
  - The initial state or goal state
  - Any type/sort information about objects
  - Any information about what the actions mean or how they work
  - Any knowledge of agent rationality or plan optimality



[Input]

LOCM: High Level Approach

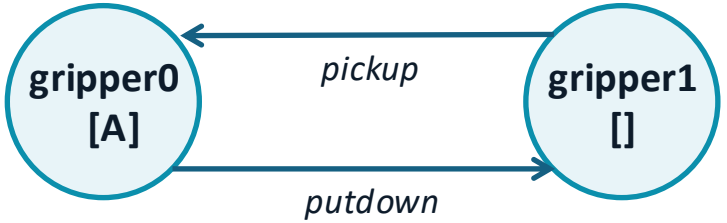
- Takes a single action sequence as the only input
- Identifies unique object **sorts** (types) in the domain
- Builds Finite State Machines (FSMs) for each sort
- Augments FSMs with inter-object parameter relationships
- Outputs lifted PDDL action models

7 Steps + Sort Formation

0	Sort Formation
1	Induce FSMs
2	Zero Analysis
3	Parameterize FSMs
4	Create & Merge Params
5	Remove Flawed Params
6	Static Preconditions
7	Output PDDL Schema

Finite State Machines (FSMs) for LOCM

- Each object sort gets its own FSM
- States represent the condition of an object
- Actions cause transitions between states
- Consecutive actions must share a state
- Same action always produces the same start → end transition



Objects transition between states as actions are applied.
LOCM reconstructs these FSMs purely from the action sequence.

Parameterization: Learning Object Relationships

Step 3

Form Hypotheses

If two consecutive transitions share an argument of the same sort, hypothesize that their shared state has a parametric association to that sort.

Step 4

Merge Parameters

Check all pairs of hypotheses. When the same incoming or outgoing action references the same position, unify the parameter variables using set merging.

Step 5

Remove Flawed Params

Remove parameters not set by all incoming transitions — these are indeterminate and would produce incorrect action models.

Result: Fully parameterized FSMs capturing both object states and cross-sort relationships