Lecture 21

CS 6260 Automated Planning and Learning

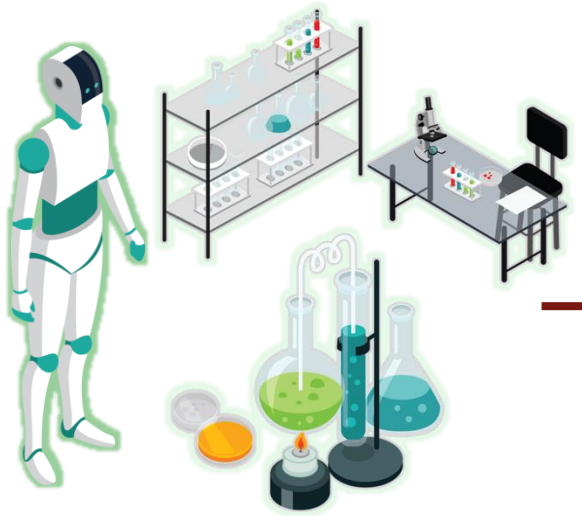
# Learning Object-Centred Models II

Department of Computer Science and Engineering

Indian Institute of Technology Madras

# Recap: LOCM

# 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]

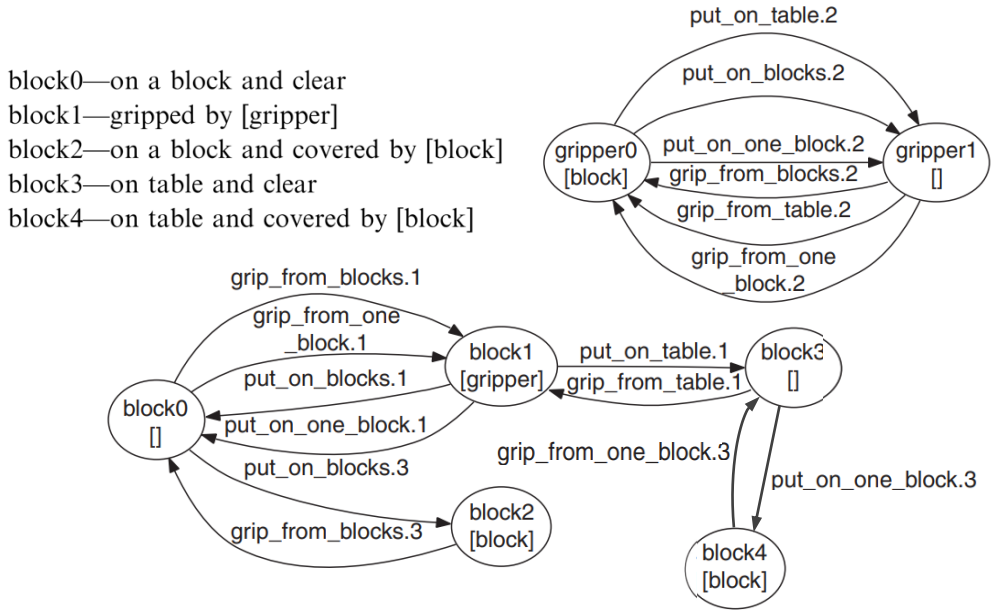
What kind of approaches these learners use?

# 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)))
```

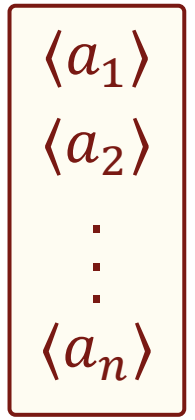
[PDDL Example]

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 |

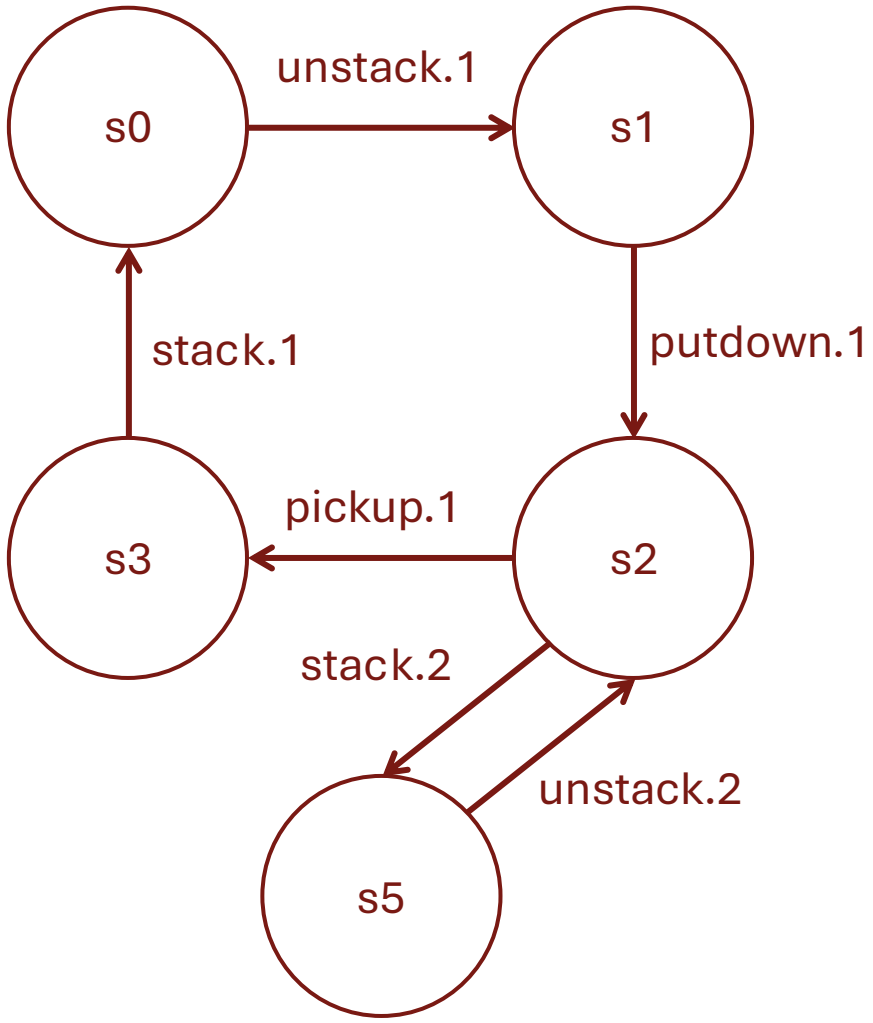
# Inducing FSMs

| | | |
|---|---|---|
| 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 | |

| | Transition | Start | End | Reason |
|---|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ | first time seen, assign $S_0 \rightarrow S_1$ |
| | putdown.1 | $S_1$ | $S_2$ | Continuity: start must = end of prev ($S_1$). Assign fresh end $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ | Continuity: start = $S_2$. Assign fresh end $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ | Continuity: start = $S_3$. Assign fresh end $S_4$ |
| B | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ | first time seen, assign $S_5 \rightarrow S_6$ |
| | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ | Using 1-1 rule |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ | |
| | unstack.1 | $S_0$ | $S_1$ | |
| | putdown.1 | $S_1$ | $S_2$ | |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ | |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ | |
| | unstack.2 | $S_5$ | $S_2$ | |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |

# Inducing FSMs

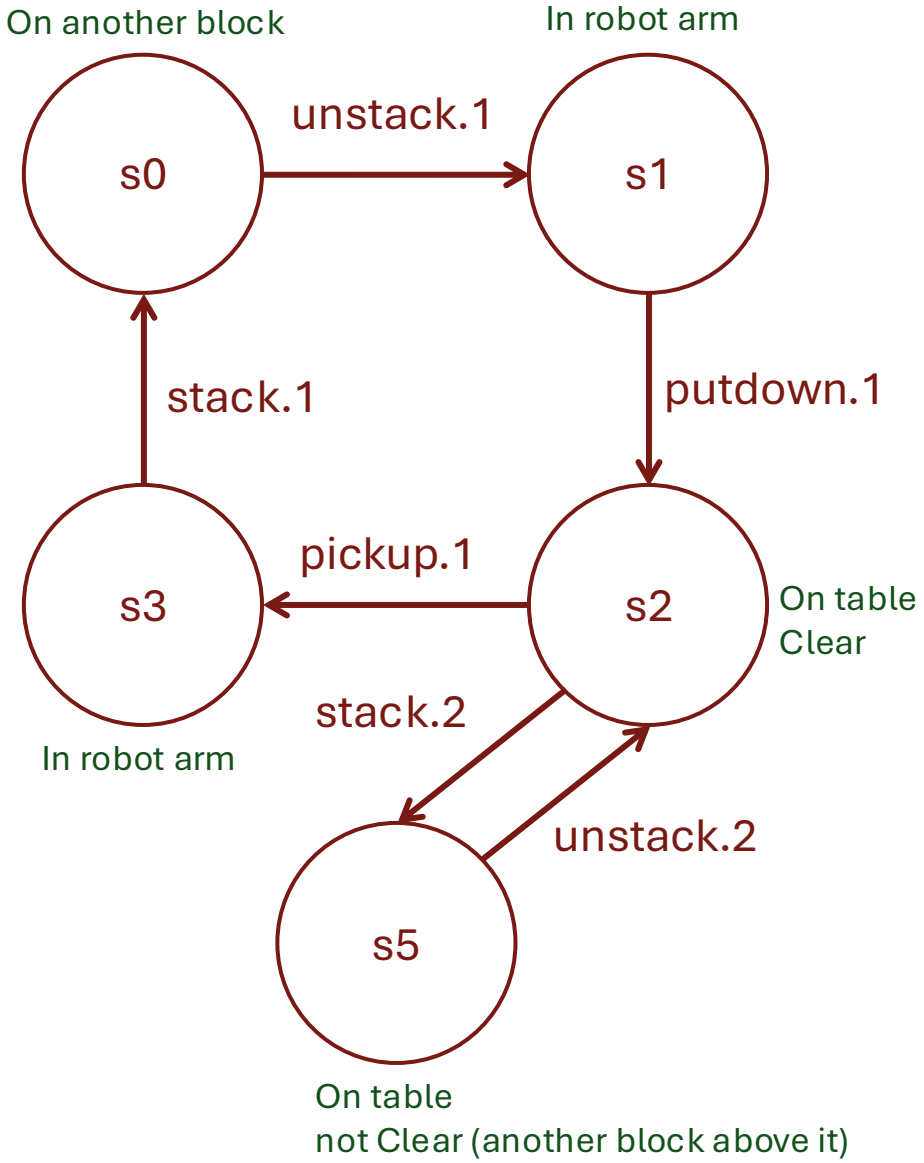



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |

# Inducing FSMs

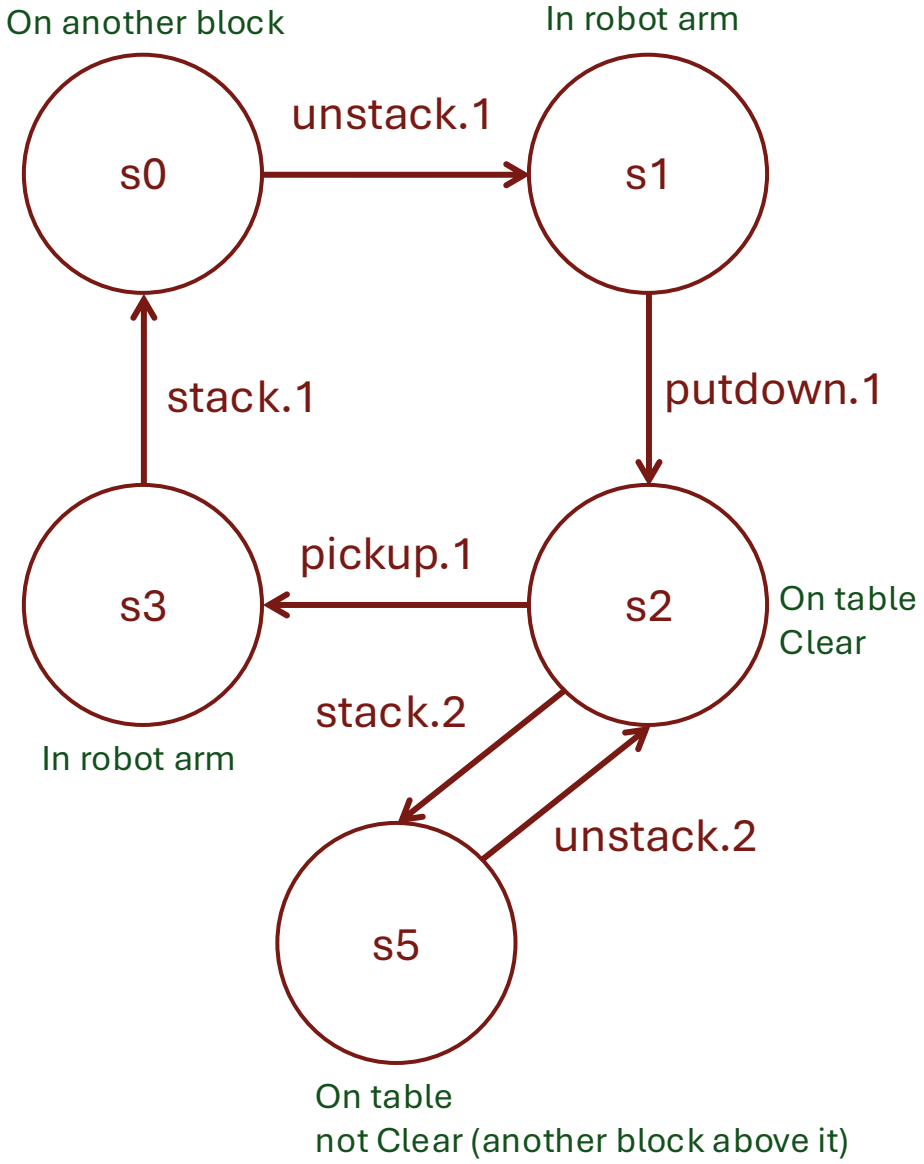



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |

# Inducing FSMs

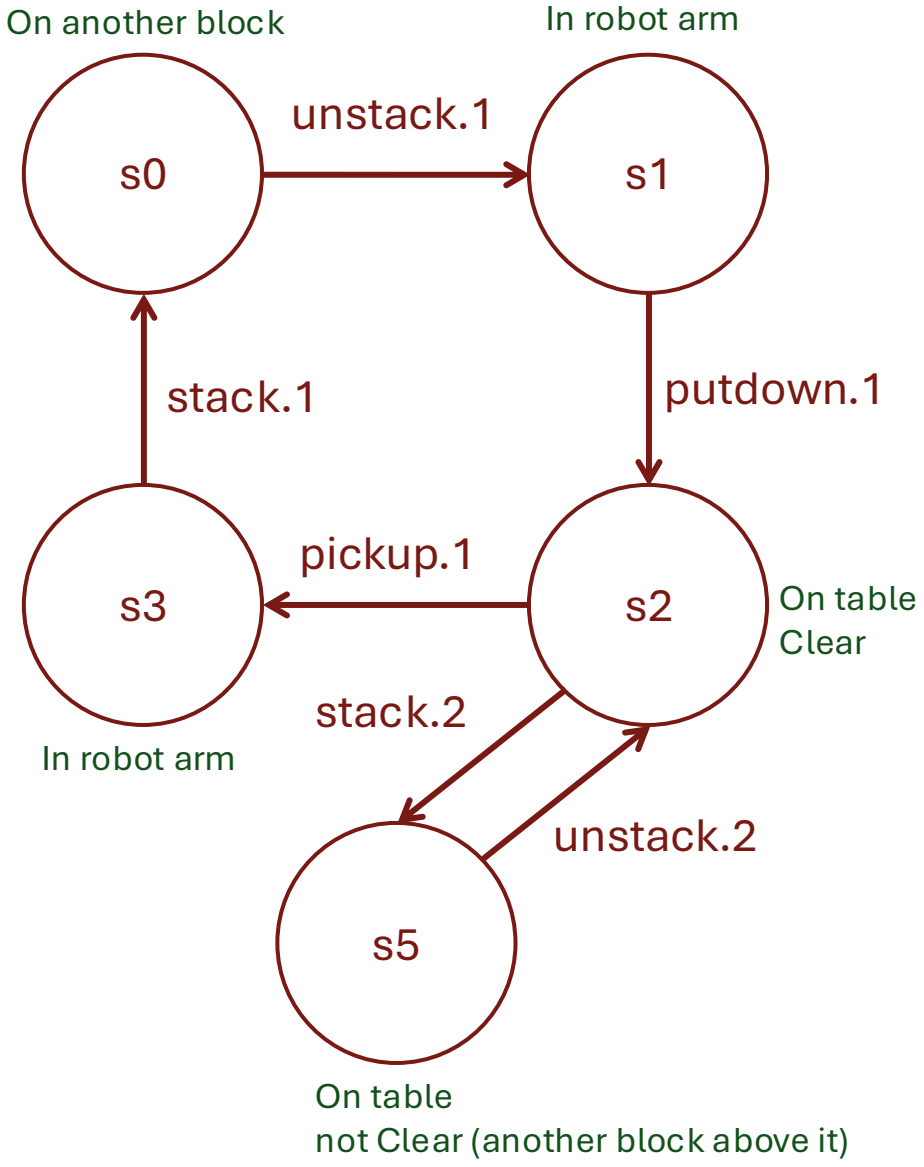



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Inducing FSMs

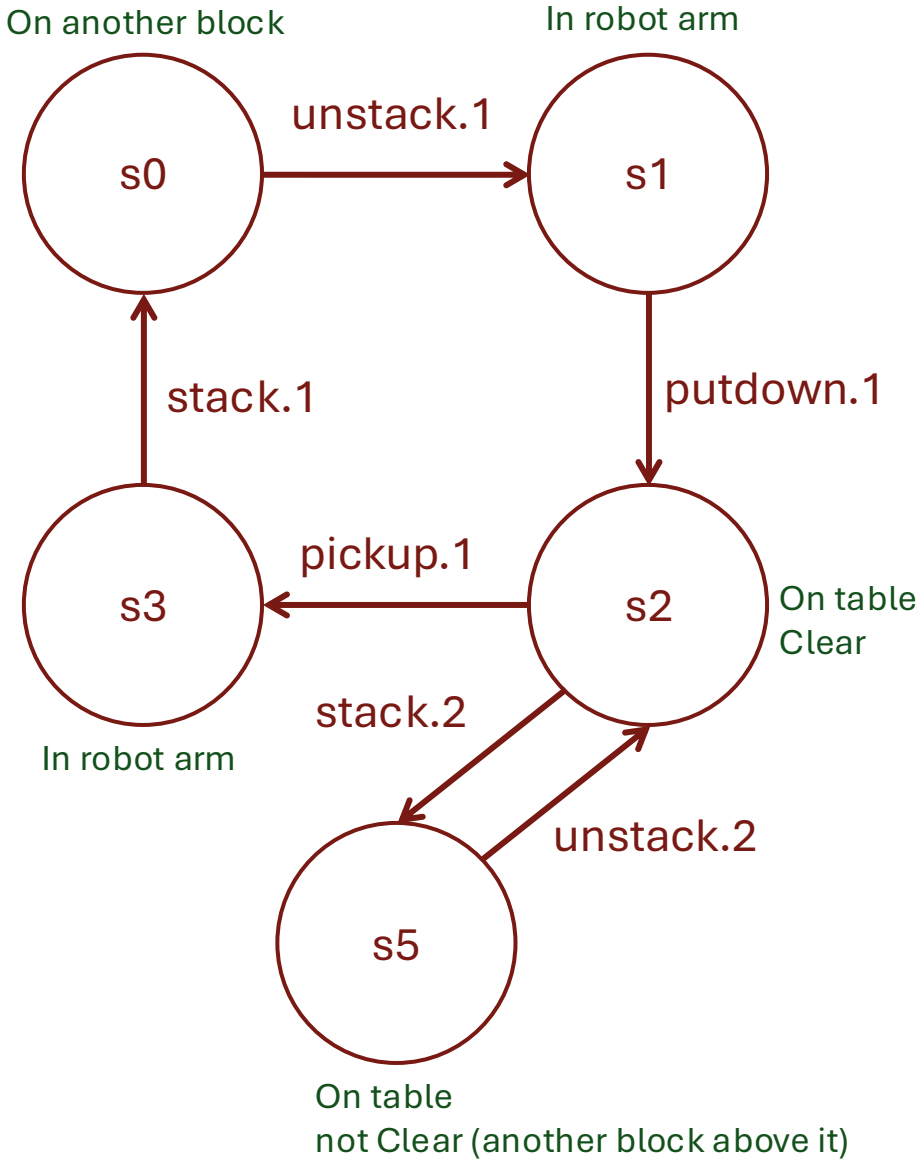



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | | | |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Inducing FSMs

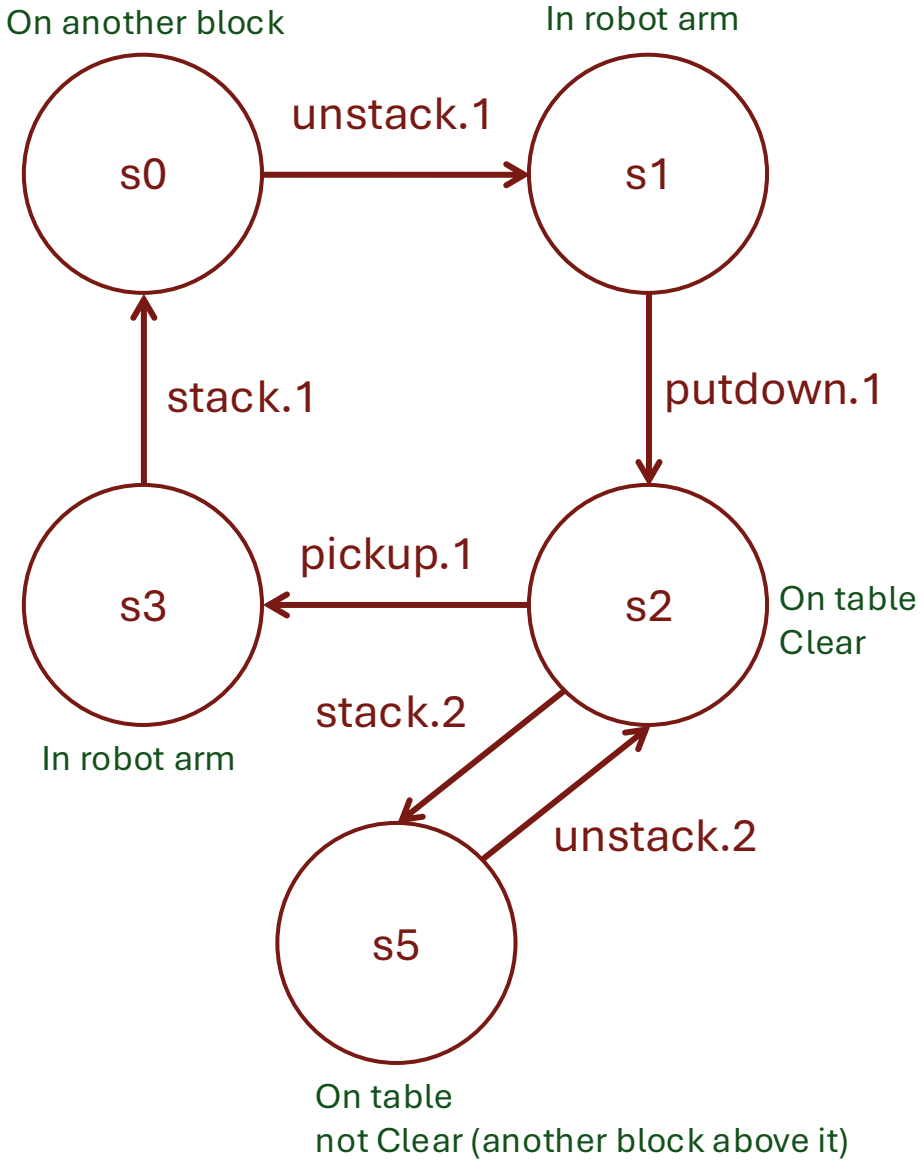



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |
| | pickup.1 | $S_2$ | $S_3$ |
| | putdown.1 | $S_1$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Inducing FSMs

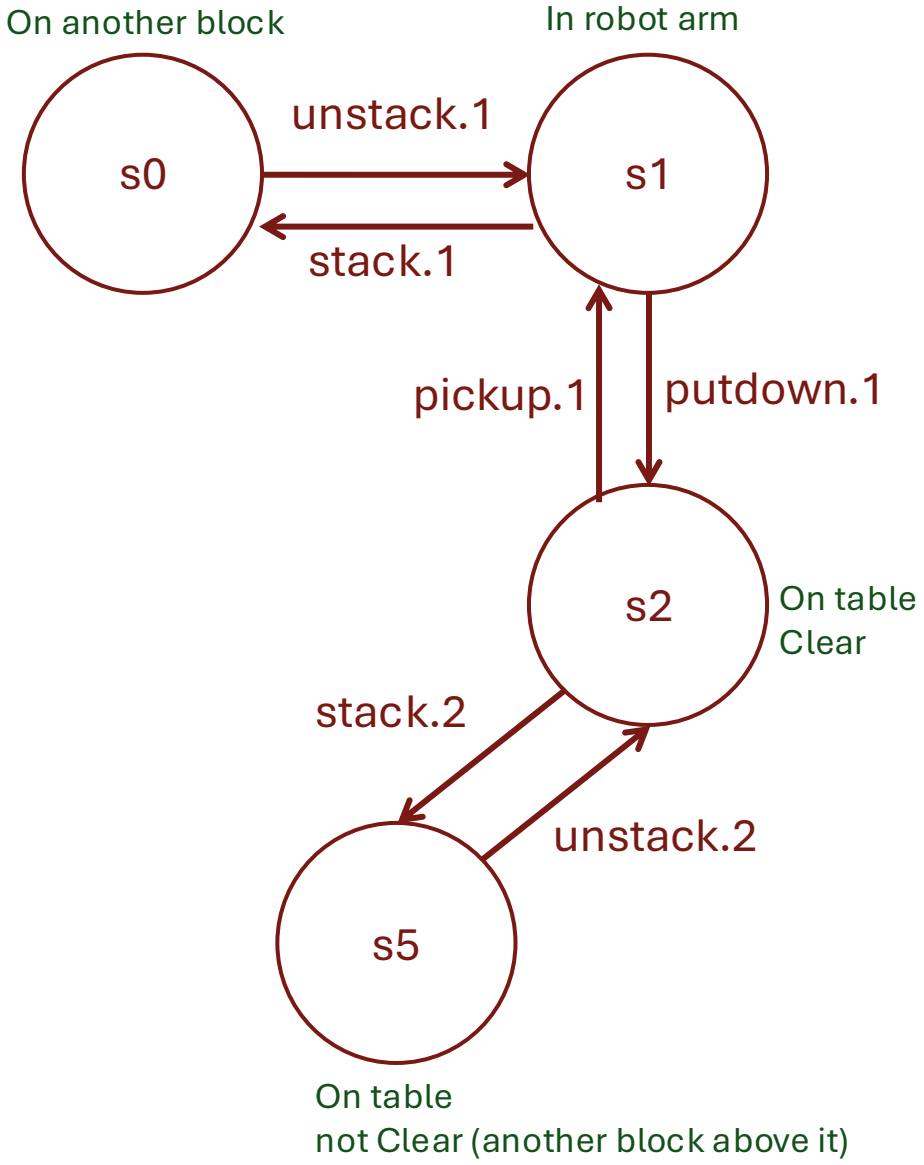



| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | ~~$S_3$~~=$S_1$ |
| | stack.1 | ~~$S_3$~~ $S_1$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |
| | pickup.1 | $S_2$ | ~~$S_3$~~=$S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Inducing FSMs




| | Transition | Start | End |
|---|---|---|---|
| A | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | pickup.1 | $S_2$ | ~~$S_3$~~=$S_1$ |
| | stack.1 | ~~$S_3$~~ $S_1$ | ~~$S_4$~~=$S_0$ |
| | unstack.2 | $S_5$ | ~~$S_6$~~=$S_2$ |
| B | pickup.1 | ~~$S_6$~~ $S_2$ | ~~$S_7$~~ $S_3$ |
| | stack.1 | $S_3$ | ~~$S_4$~~=$S_0$ |
| | unstack.1 | $S_0$ | $S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |
| | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| C | stack.2 | $S_2$ | ~~$S_7$~~=$S_5$ |
| | unstack.2 | $S_5$ | $S_2$ |
| | pickup.1 | $S_2$ | ~~$S_3$~~=$S_1$ |
| | putdown.1 | $S_1$ | $S_2$ |

| | |
|---|---|
| 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 |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Zero Analysis

| | |
|---|---|
| 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 |

- **Goal:** Detect implicit "background" objects,like the robot arm, that are never named in the trace but whose state constrains what can happen next.
- Treat every action as if it also acts on a single invisible "zero object" at position 0. Build its FSM using the same rules as Step 1.

| Transition | Start | End |
|---|---|---|
| unstack.0 | | |
| putdown.0 | | |
| pickup.0 | | |
| stack.0 | | |
| unstack.0 | | |
| putdown.0 | | |
| pickup.0 | | |
| stack.0 | | |
| pickup.0 | | |
| putdown.0 | | |

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

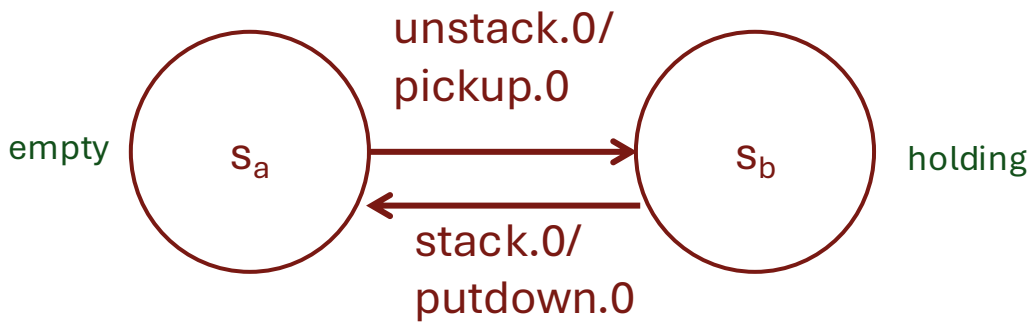
unstack.0 → putdown.0 → pickup.0 → stack.0 → unstack.0 → putdown.0 → pickup.0 → stack.0 → pickup.0 → putdown.0

# Zero Analysis

| | |
|---|---|
| 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 |

- **Goal:** Detect implicit "background" objects,like the robot arm, that are never named in the trace but whose state constrains what can happen next.
- Treat every action as if it also acts on a single invisible "zero object" at position 0. Build its FSM using the same rules as Step 1.

| Transition | Start | End |
|---|---|---|
| unstack.0 | | |
| putdown.0 | | |
| pickup.0 | | |
| stack.0 | | |
| unstack.0 | | |
| putdown.0 | | |
| pickup.0 | | |
| stack.0 | | |
| pickup.0 | | |
| putdown.0 | | |




| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |

# Parameterization

| | |
|---|---|
| 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 |

- **Goal:** Discover associations between objects across sorts.
- E.g., when the arm is in state H1 (holding), which block is it holding?
- This requires linking the arm's FSM to the block sort's FSM via a shared parameter.
- 3 Steps:
  - Parameterize FSMs
  - Create and Merge Parameters
  - Remove Flawed Parameters.

| # | Action | arg1 | arg2 |
|---|---|---|---|
| 1. | unstack | A | B |
| 2. | putdown | A | - |
| 3. | pickup | B | - |
| 4. | stack | B | C |
| 5. | unstack | B | C |
| 6. | putdown | B | - |
| 7. | pickup | A | - |
| 8. | stack | A | B |
| 9. | pickup | C | - |
| 10 | putdown | C | - |