Lecture 23

CS 6260 Automated Planning and Learning

LOCM2

Department of Computer Science and Engineering

Indian Institute of Technology Madras

Logistics

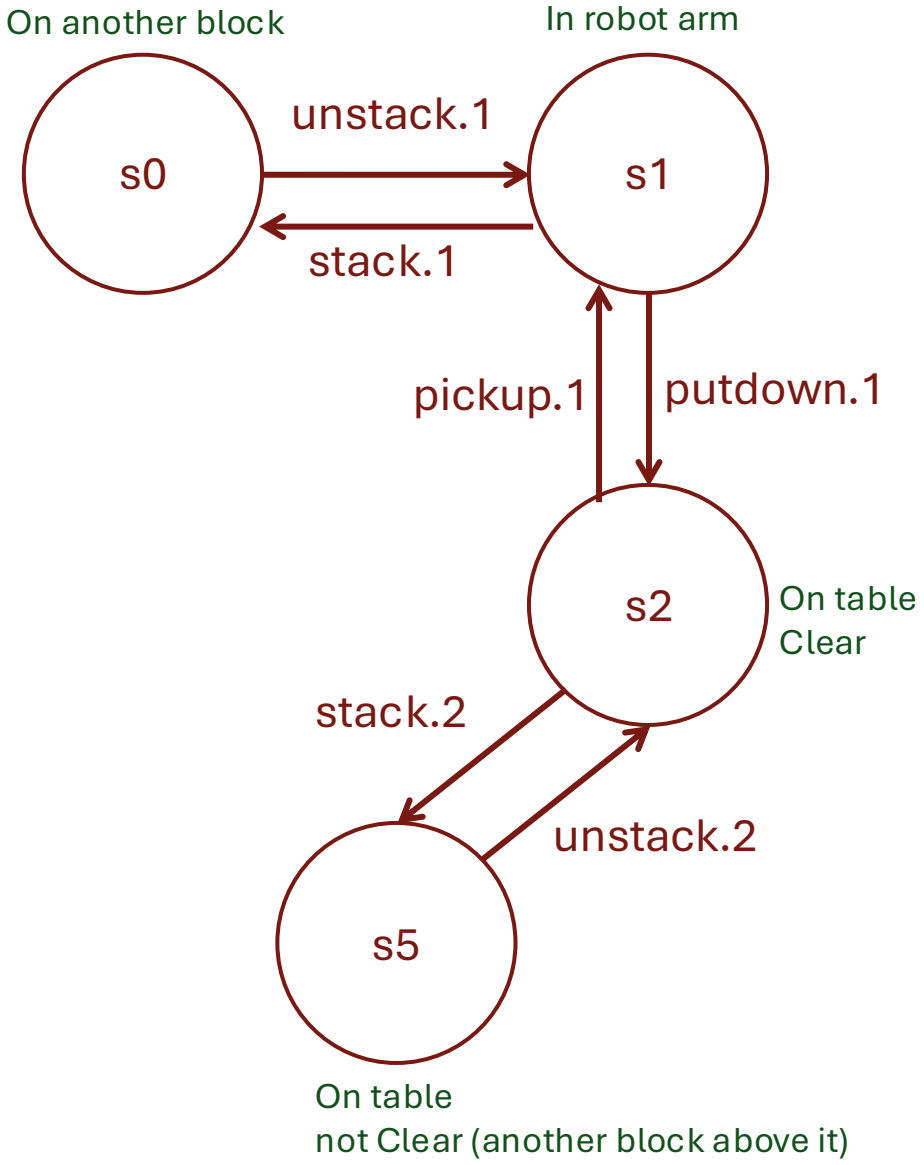
Logistics

- 3 Exams:
  1. [Quiz 1] Feb 19: 08:00 AM – 08:50 AM
  2. [Quiz 2] Apr 10: 05:00 PM – 05:50 PM
  3. [End Sem] May 12

- Office Hours: 1 hour after class every week Tu, W, Th

Recap: LOCM

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	-

LOCM2

Generalised Domain Model Acquisition from Action Traces

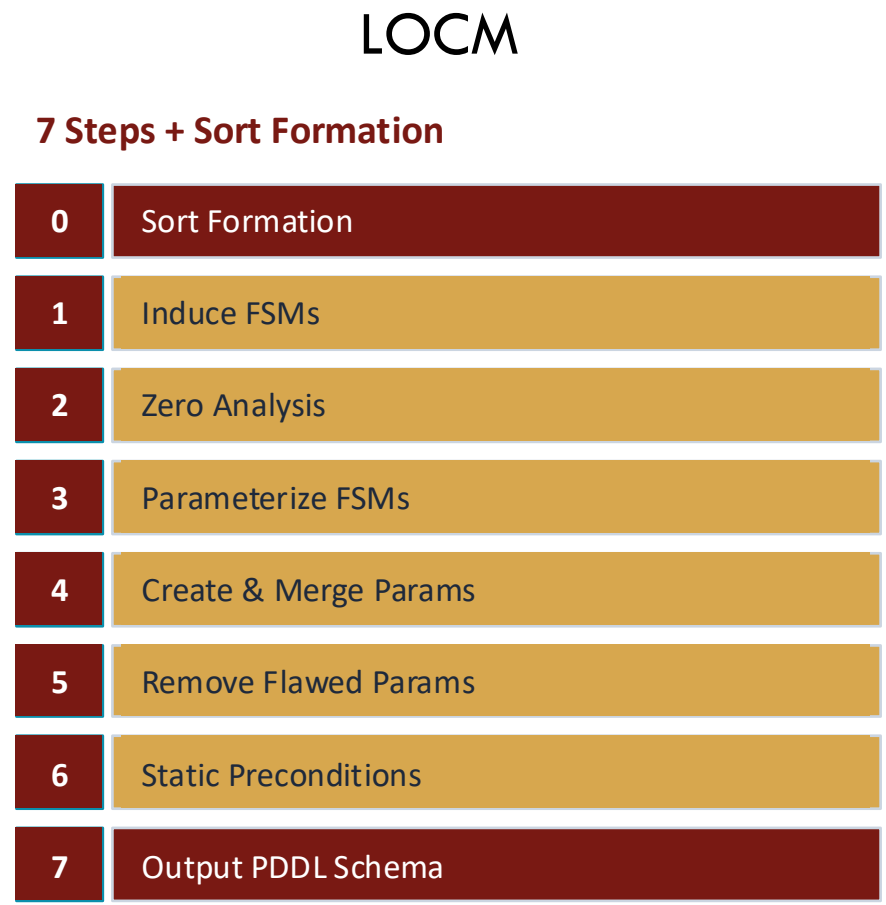
Stephen Cresswell and Peter Gregory

ICAPS 2011

https://ojs.aaai.org/index.php/ICAPS/article/view/13476/13325

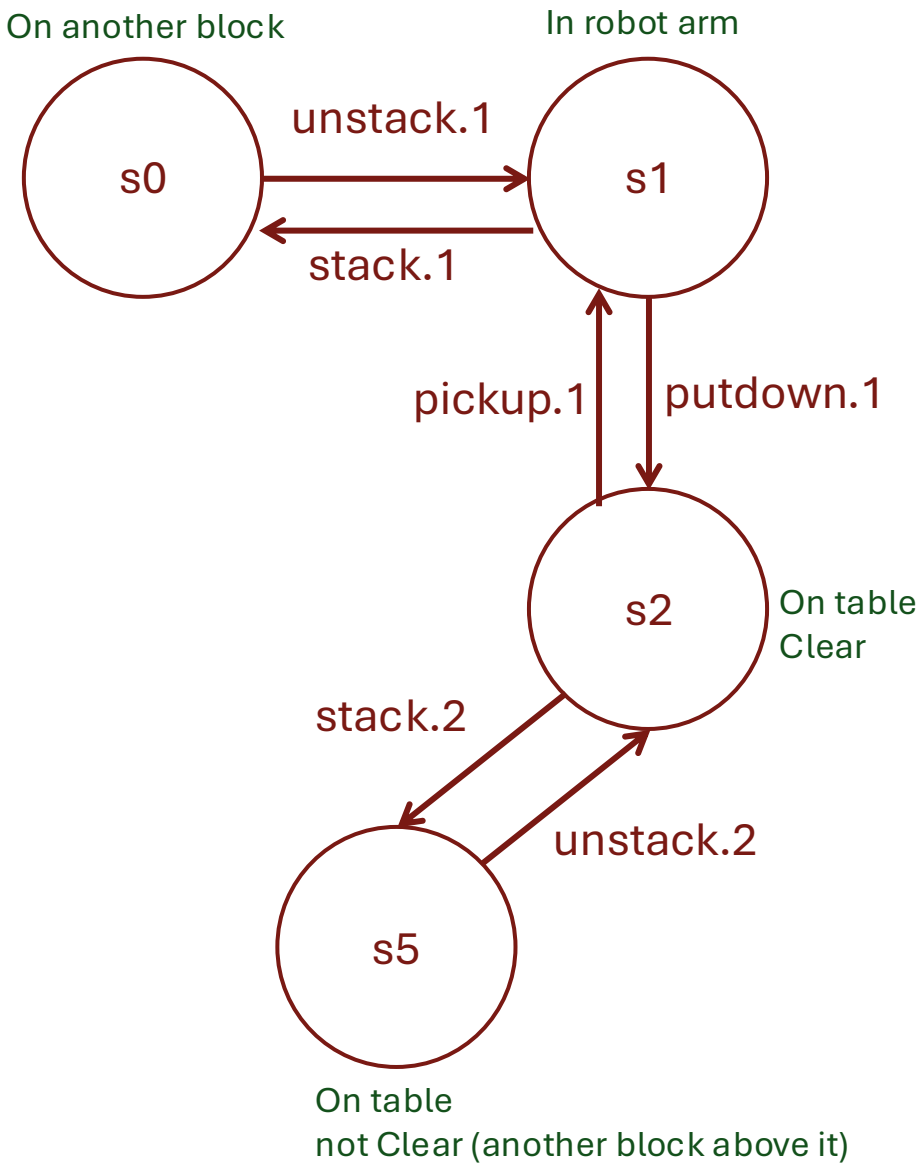
LOCM2

LOCM2



Do we need more than one FSM for each sort?

LOCM2



LOCM2 →

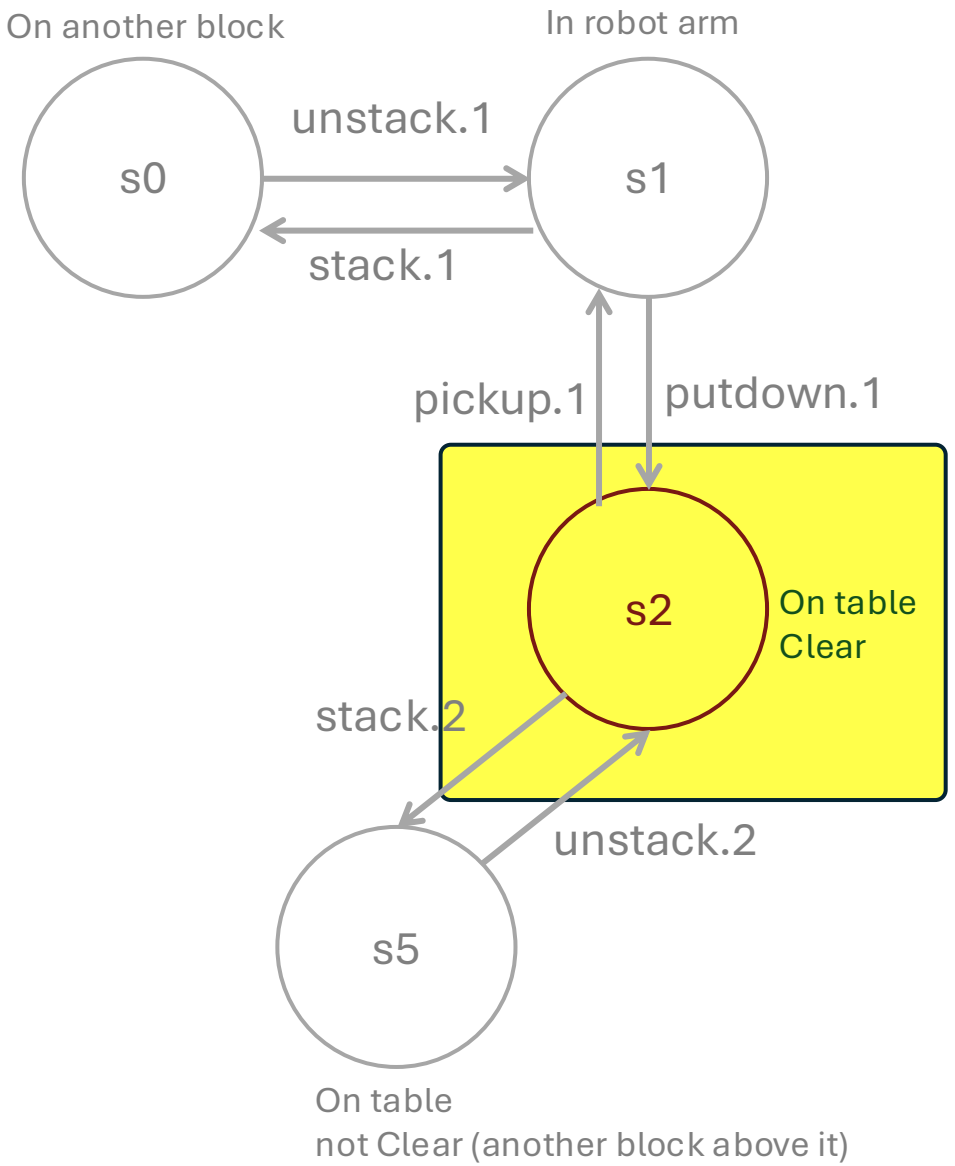
LOCM

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

Do we need more than one FSM for each sort?

LOCM2



LOCM2

LOCM

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

Do we need more than one FSM for each sort?

LOCM2

- Get unique transitions
  1. unstack.1
  2. putdown.1
  3. pickup.1
  4. stack.1
  5. unstack.2
  6. stack.2

- Create transition matrix

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1						
putdown.1						
pickup.1						
stack.1						
stack.2						
unstack.2						

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1						
pickup.1						
stack.1						
stack.2						
unstack.2						

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X			
pickup.1						
stack.1						
stack.2						
unstack.2						

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		x				
putdown.1			x			
pickup.1				x		
stack.1						
stack.2						
unstack.2						

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X			
pickup.1				X		
stack.1						
stack.2						
unstack.2		X				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X			
pickup.1				X		
stack.1						
stack.2						
unstack.2		X				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X			
pickup.1				X		
stack.1	X					
stack.2						
unstack.2		X				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X			
pickup.1				X		
stack.1	X					
stack.2						
unstack.2		X				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		x				
putdown.1			x		x	
pickup.1				x		
stack.1	x					
stack.2						
unstack.2		x				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X		X	
pickup.1				X		
stack.1	X					
stack.2						X
unstack.2		X				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X		X	
pickup.1				X		
stack.1	X					
stack.2						X
unstack.2			X			

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		x				
putdown.1			x		x	
pickup.1		x		x		
stack.1	x					
stack.2						x
unstack.2		x				

- Mark consecutive pairs as (r,c)

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2

- Create transition matrix

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		x				
putdown.1			x		x	
pickup.1		x		x		
stack.1	x					
stack.2						x
unstack.2			x			

- Mark consecutive pairs as (r,c)
- Find holes in the matrix

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2: Finding Holes in the Matrix

Input: Matrix M: the full set of transitions
Output: H — set of holes
1. Initialize H = {}
2. For every pair of rows r and r' in M (r ≠ r'):
   For every pair of columns c and c' in M (c ≠ c'):
   If M[r][c] = x (r is followed by c)
   AND M[r][c'] = x (r is also followed by c')
   AND M[r'][c] = x (r' is followed by c)
   AND M[r'][c'] ≠ x: (but r' is NOT followed by c')
   Add (r', c') to H ← this is a hole
3. Return H

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X		X	
pickup.1		X		X		
stack.1	X					
stack.2						X
unstack.2			X			

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

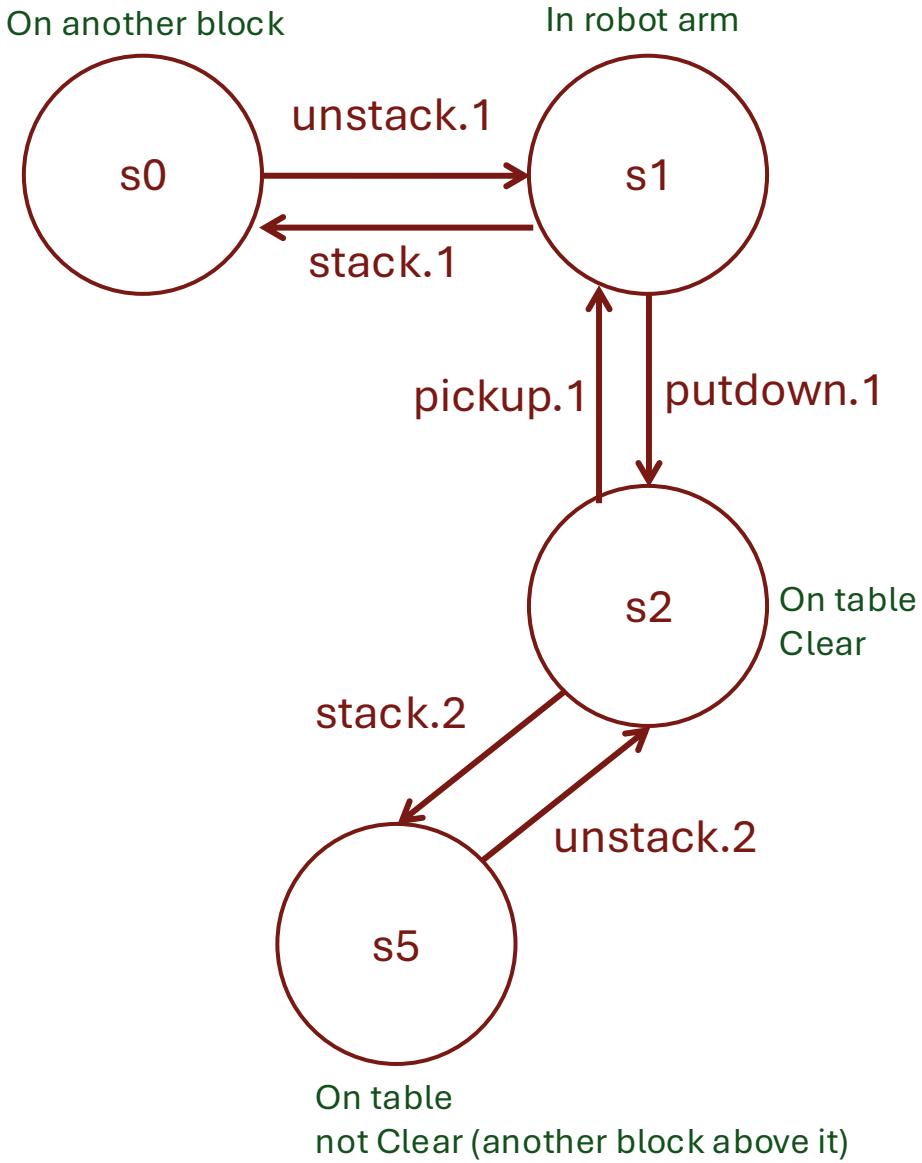
LOCM2: Finding Holes in the Matrix

Input: Matrix M: the full set of transitions
Output: H — set of holes
1. Initialize H = {}
2. For every pair of rows r and r' in M (r ≠ r'):
   For every pair of columns c and c' in M (c ≠ c'):
   If M[r][c] = x (r is followed by c)
   AND M[r][c'] = x (r is also followed by c')
   AND M[r'][c] = x (r' is followed by c)
   AND M[r'][c'] ≠ x: (but r' is NOT followed by c')
   Add (r', c') to H ← this is a hole
3. Return H

	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X		X	
pickup.1		X		X		
stack.1	X					
stack.2						X
unstack.2			X			

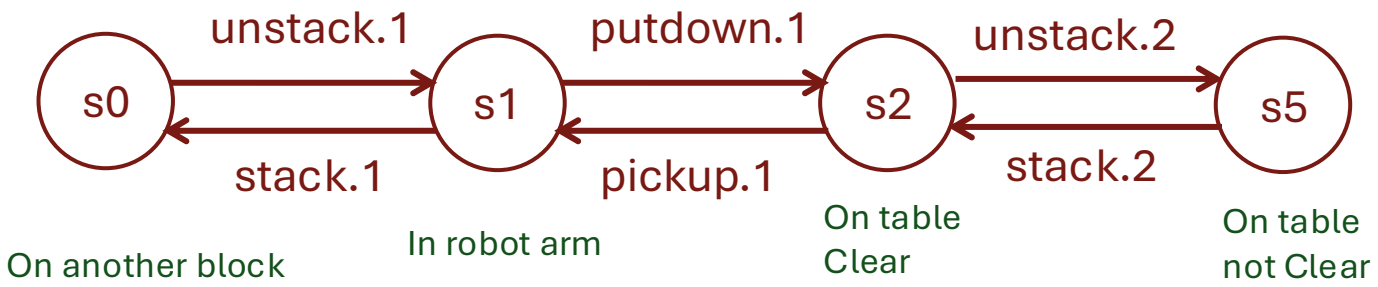
	Transition	Start	End
A	unstack.1	S_0	S_1
	putdown.1	S_1	S_2
	pickup.1	S_2	~~S_3~~=S_1
	stack.1	~~S_3~~ S_1	~~S_4~~=S_0
B	unstack.2	S_5	~~S_6~~=S_2
	pickup.1	~~S_6~~ S_2	~~S_7~~ S_3
	stack.1	S_3	~~S_4~~=S_0
	unstack.1	S_0	S_1
	putdown.1	S_1	S_2
	stack.2	S_2	~~S_7~~=S_5
C	stack.2	S_2	~~S_7~~=S_5
	unstack.2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

LOCM2: Analyzing Holes in the Matrix



	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2: Analyzing Holes in the Matrix



	u.1	pu.1	pi.1	s.1	s.2	u.2
unstack.1		X				
putdown.1			X		X	
pickup.1		X		X		
stack.1	X					
stack.2						X
unstack.2			X			

	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
B	unstack.2	S_5	~~S_6~~=S_2
	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

LOCM2: Creating FSMs from Holes

- Input: $\mathbf{T_{all}}$: all observed transitions for the sort; **H**: set of holes, each hole is an unordered pair {**t1**, **t2**} of transitions;
  P: set of observed consecutive pairs (**t1**, **t2**); **E** : the full example sequences
- Output: **S** : set of transition sets, each defining one FSM

S ← ∅
for each hole **h** in **H**:
    if there is already some **s** ∈ **S** such that **h** ⊆ **s**:
        skip #this hole is already covered
    else:
        by breadth-first search, find the smallest set **s** such that: **h** ⊆ **s** and **s** ⊂ $\mathbf{T_{all}}$ and
        the transition matrix formed from **s** using all pairs (**t1**,**t2**) ∈ **P** with {**t1**,**t2**} ⊆ **s** is well-formed and
        s is consistent with the example sequences **E** (no inappropriate dead-ends)
        if such **s** found:
            S ← **S** ∪ {**s**}
        else:
            discard #no valid subset covers this hole (the all-transitions machine will handle it)
for any two sets **s1**, **s2** ∈ **S**:
    if **s1** ⊂ **s2**:
        S ← **S** \ {**s1**}
S ← **S** ∪ {$\mathbf{T_{all}}$}
return **S**

LOCM2

- Continue normally after new FSM(s) generated

LOCM2

LOCM

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