Lecture 14

Planning Wrap Up

Slide 2

Lecture Overview

- Recap of Lecture 13
- Planning as CSP
  - Details on CSP representation -

  Solving the CSP planning problem
- Intro to Logic (time permitting)

Solving planning problems

- STRIPS lends itself to solve planning problems either

- As pure search problems
- As CSP problems

- We will look at one technique for each approach

Slide 4

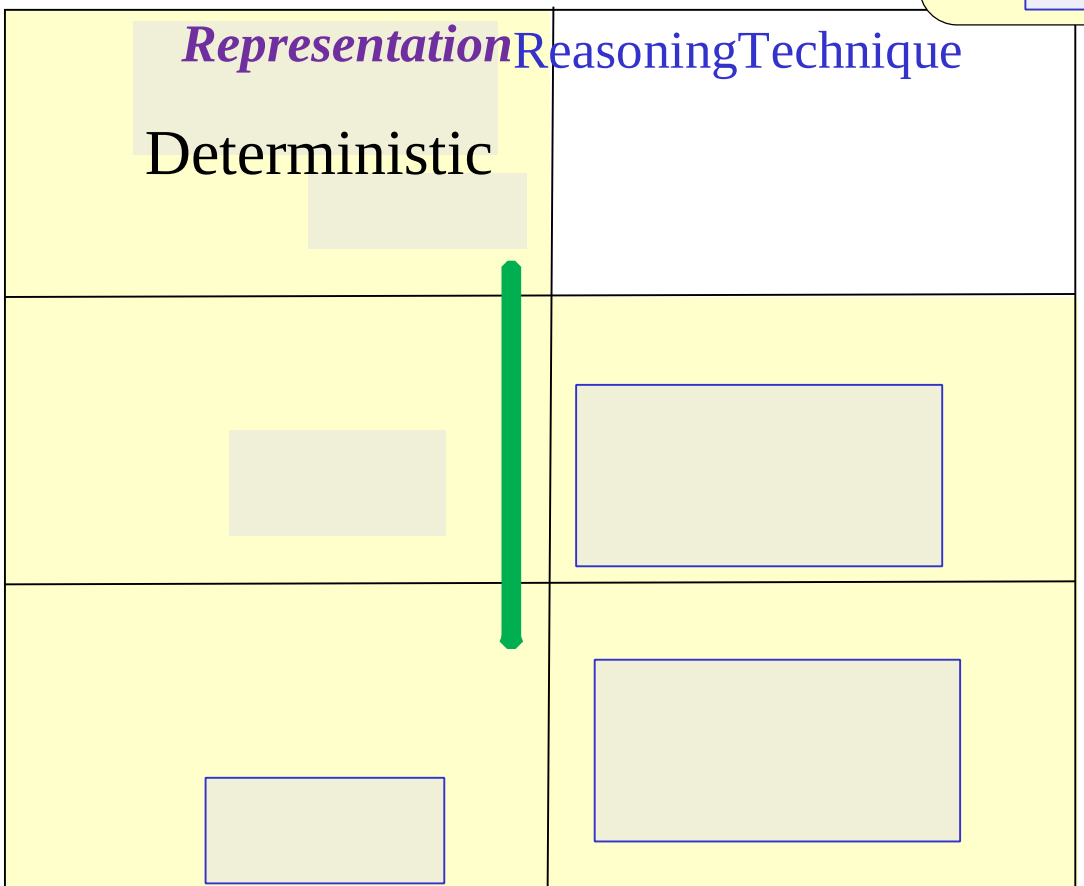
Course Overview

Problem Type

Arc **Environment**Stochastic

RepresentationReasoningTechnique

Deterministic



Consistency

Constraint Satisfaction ***Vars + Constraints*** Search

Static ***Belief Nets***

Query ***Logics*** Variable

Search

Elimination

Sequential ***STRIPS*** ***Decision Nets*** Variable

Planning Search Elimination

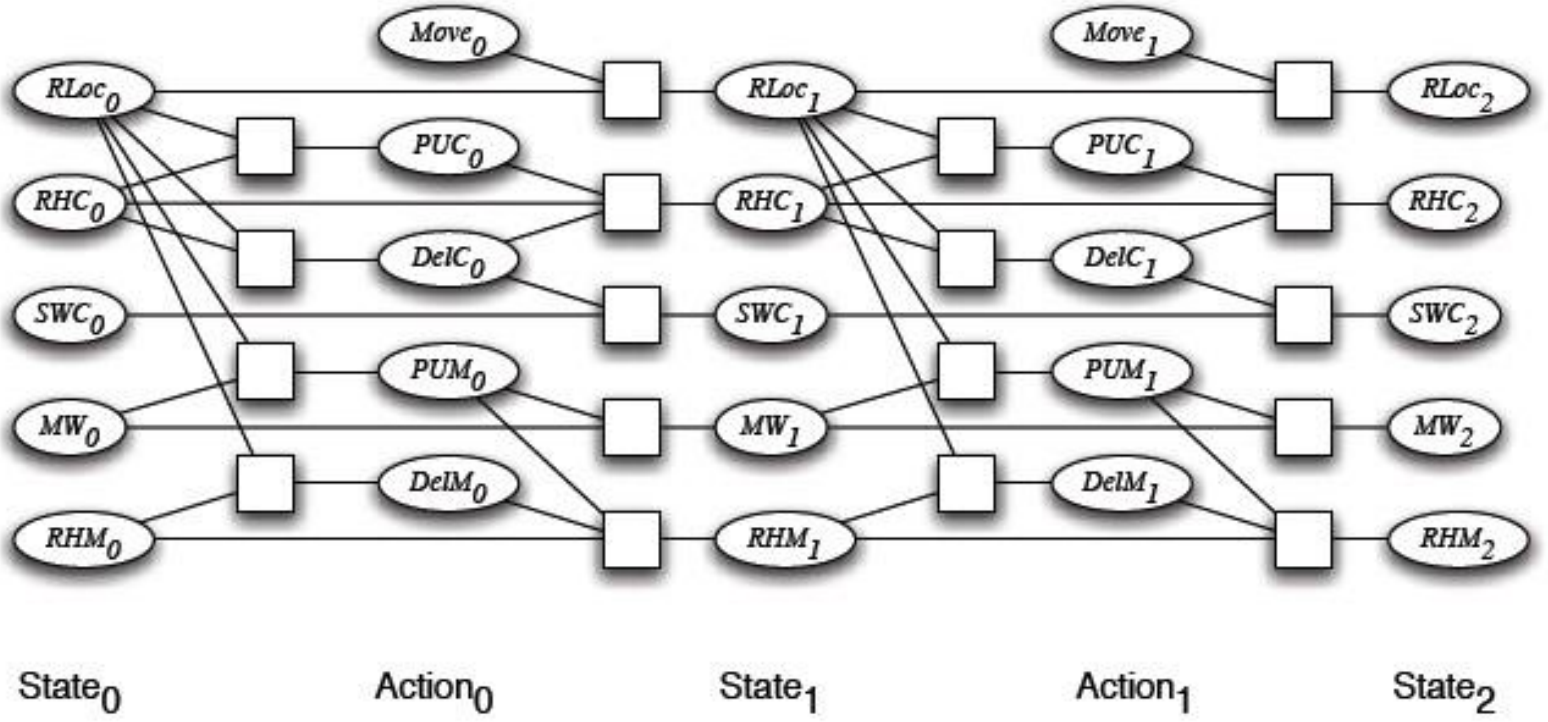
Markov Processes

Value

Iteration

Planning as a CSP: General Idea

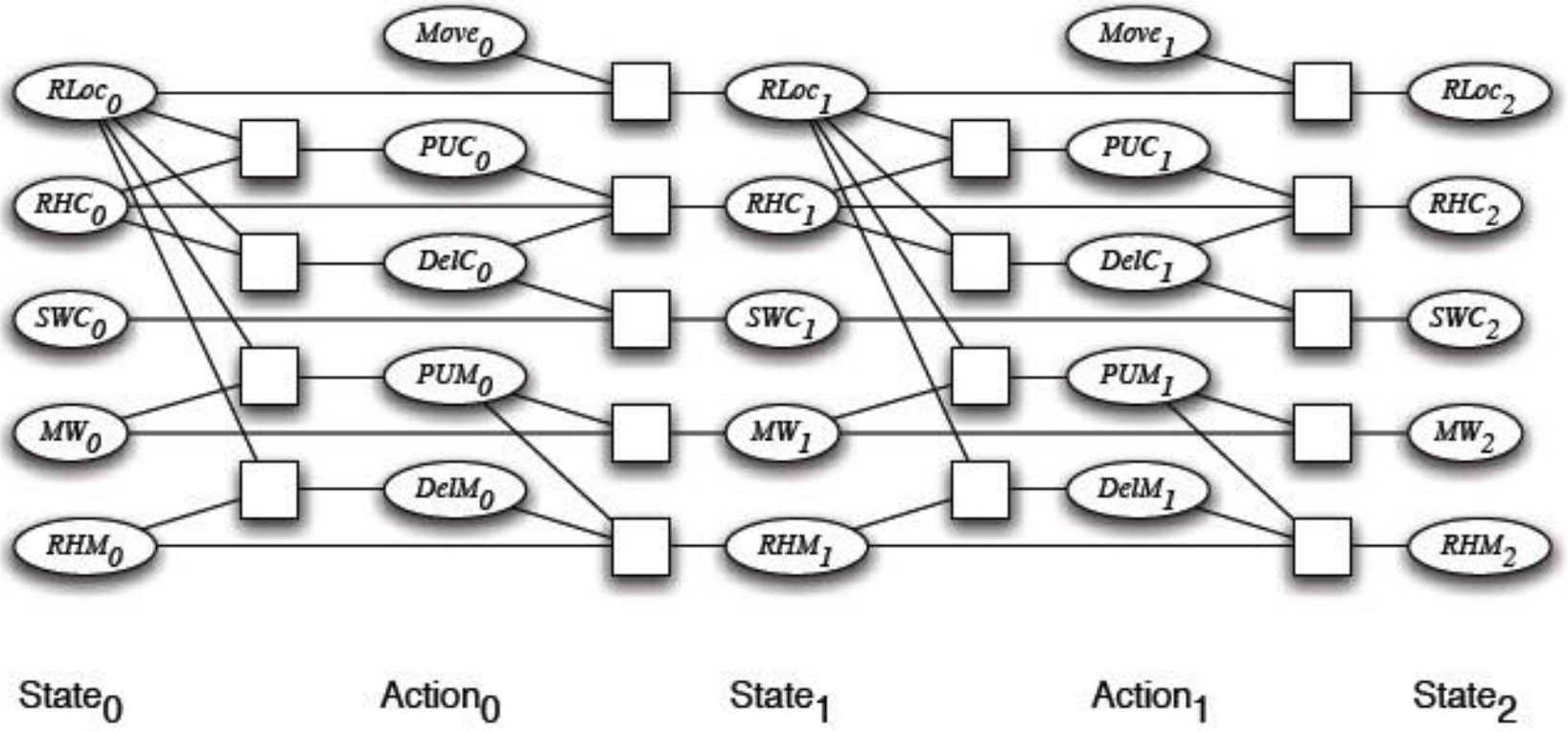
- Both features and actions are CSP variables
- Action preconditions and effects are constraints among
- the action,
- the states in which it can be applied
- the states that it can generate



Planning as a CSP: General Idea

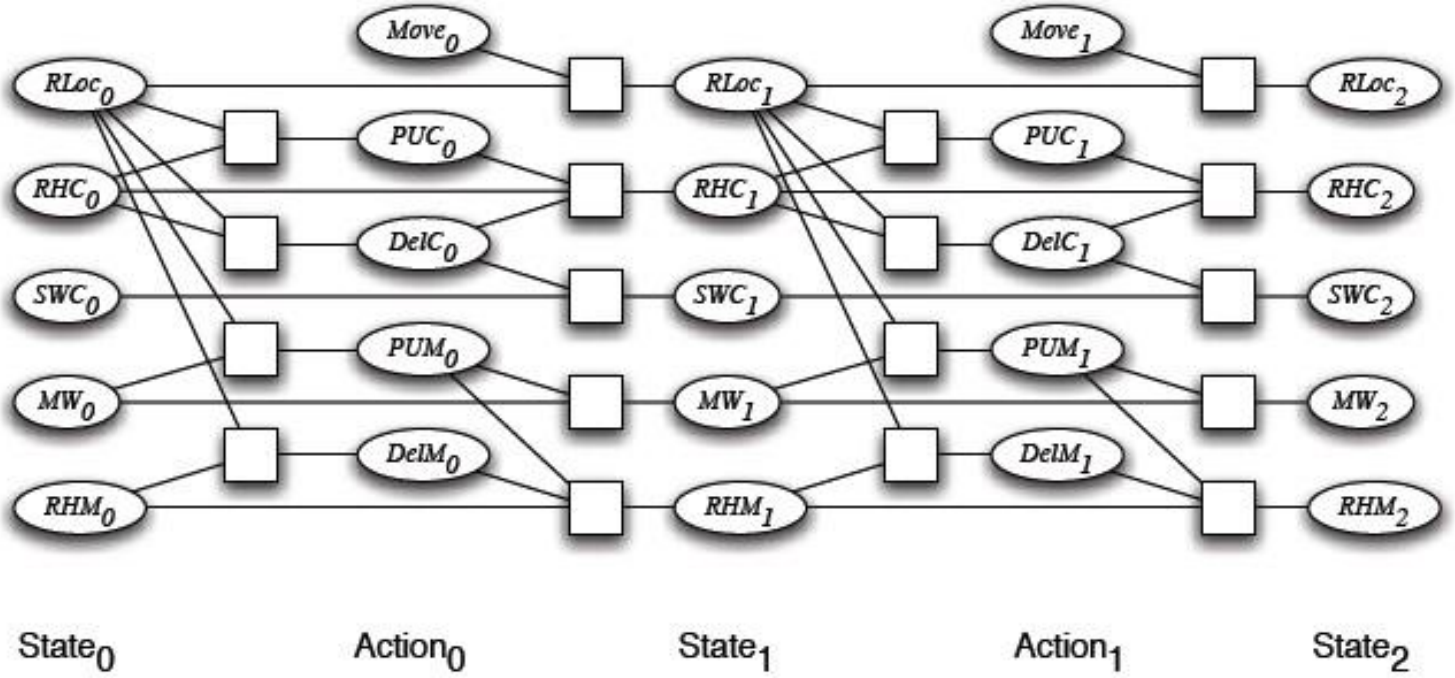
- These action constraints relate to states at a given time t, the corresponding valid actions and the resulting states at t +1

- we need to have as many state and action variables as we have planning steps



Planning as a CSP: Variables

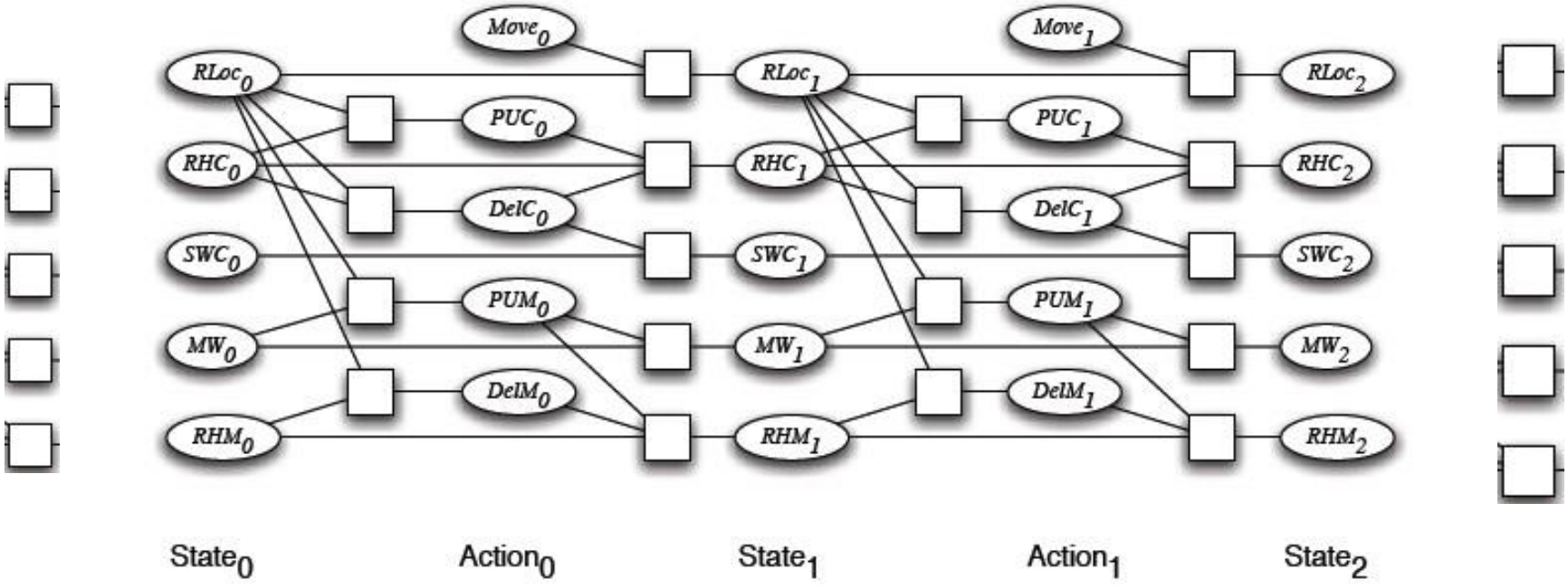
- We need to 'unroll the plan' for a fixed number of steps: this is called the horizon k
- To do this with a horizon of k:
- construct a CSP variable for each STRIPS state variable at each time step from 0 to k



- construct a Boolean CSP variable for each STRIPS action at each time step from 0 to k - 1.

Initial and Goal Constraints

- initial state constraints: unary constraints on the values of the state variables at time 0



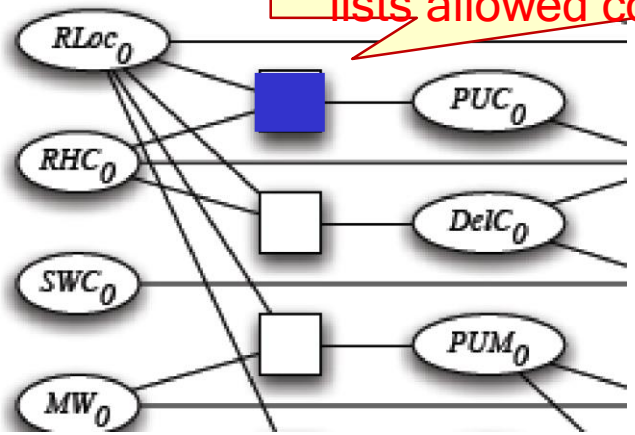
- goal constraints: unary constraints on the values of the state variables at time k

CSP Planning: Precondition Constraints

precondition constraints

- between state variables at time t and action variables at time t
- specify when actions may be taken. E.g.,

Truth table for this constraint:
lists allowed combinations of values



$RLoc_0$	RHC_0	PUC_0
cs	T	F
cs	F	T
cs	F	F
mr	*	F
lab	*	F
off	*	F

Need to allow for the option of *not* taking an action even when it is valid

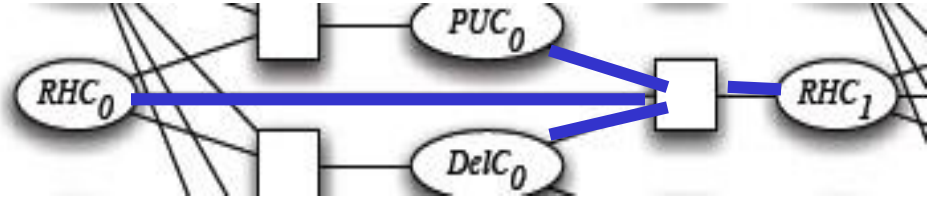
- ✓robot can only pick up coffee when Loc=cs (coffee shop) and RHC = false (don't have coffee already)

CSP Planning: Effect Constraints
Lecture Overview

- Recap of Lecture 13
- Planning as CSP
  - More details on CSP representation
    - Solving the CSP planning problem
- Intro to Logic

Effect constraints

- Between action variables at time t and state variables at time t+1
  - ✓ Specify how each state variable v at time t+1 can be modified by actions at t
- Also depend on state variable v at time t (frame rule!)
- E.g. let's consider variable RHC at time t and t+1 and fill in a few rows in this table



RHC_t	$DelC_i$	PUC_i	RHC_{t+1}
T	T	T	
T	T	F	
T	F	T	
T	F	F	

CSP Planning: Effect Constraints

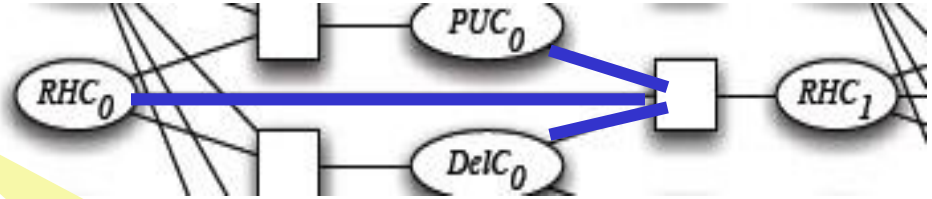
F	T	T	
F	T	F	
F	F	T	
F	F	F	

CSP Planning: Effect Constraints

Effect constraints

- Between action variables at time t and state variables at time t+1
  - ✓ Specify how each state variable v at time t+1 can be modified by actions at t
- Also depend on state variable v at time t (frame rule!)

RHC_t	$DelC_i$	PUC_i	RHC_{t+1}
T	T	T	
T	T	F	F
T	F	T	
T	F	F	T
F	T	T	
F	T	F	
F	F	T	T
F	F	F	F



Does not quite matterwhat we put here since other constraints enforce that these configurations won't happen

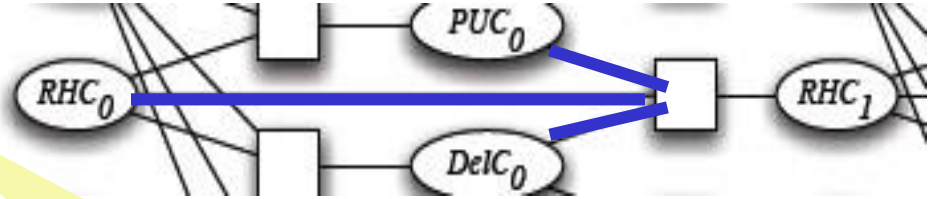
CSP Planning: Effect Constraints

- E.g. let's consider variable RHC at time t and t+1 and fill in a few rows in this table

Effect constraints

- Between action variables at time t and state variables at time t+1
  - ✓ Specify how each state variable v at time t+1 can be modified by actions at t
- Also depend on state variable v at time t (frame rule!)
- E.g. let's consider variable RHC at time t and t+1 and fill in a few rows in this table

RHC_t	$DelC_i$	PUC_i	RHC_{t+1}
T	T	T	F
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F



Does not quite matterwhat we put here since other constraints enforce that these configurations won't happen

CSP Planning: Effect Constraints

Additional constraints in CSP Planning

Other constraints we may want are action constraints:

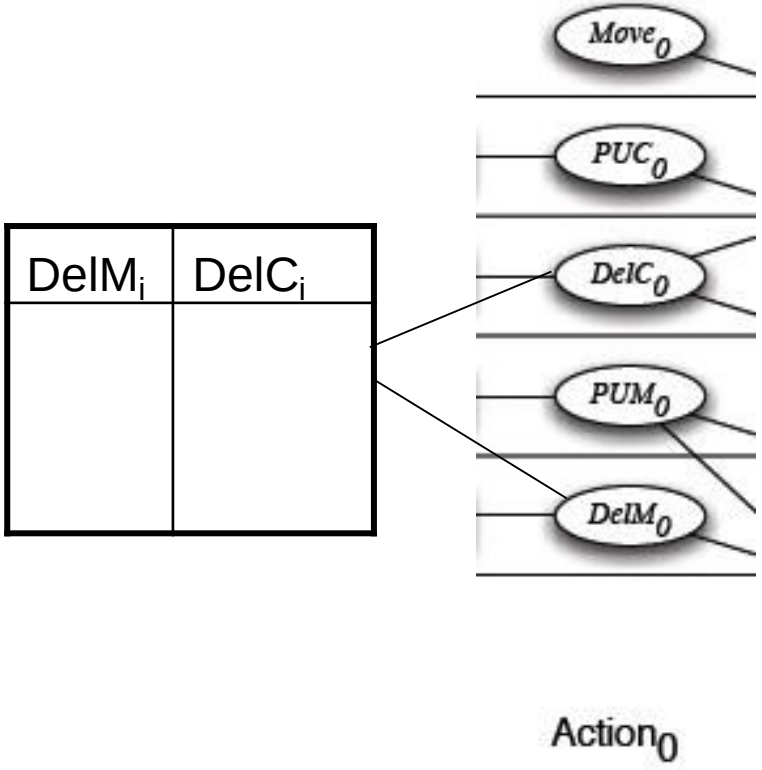
- specify which actions cannot occur simultaneously

- these are often called mutual exclusion (mutex) constraints

E.g. in the Robot domain

DelMand DelCcan occur in any sequence (or simultaneously)

But we can enforce that they do not happen simultaneously



CSP Planning: Constraints Contd.

Other constraints we may want are action constraints:

- specify which actions cannot occur simultaneously

$DelM_i$	$DelC_i$
??	

A.

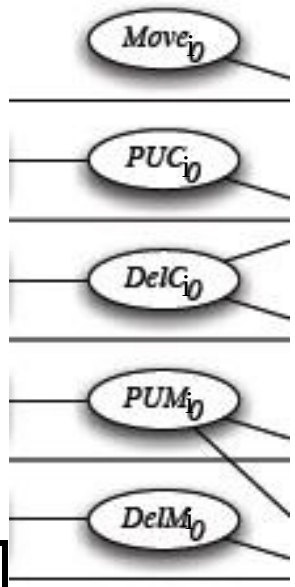
$DelM_i$	$DelC_i$

Handling mutex constraints in Forward Planning

T	T
T	F
F	T

- these are sometimes called mutual exclusion (mutex) constraints

How can we specify that DelMand DelCcannot occur simultaneously ?



B.

DelM$_i$	DelC$_i$
T	F
F	T

C.

DelM$_i$ D.	DelC$_i$
T	F
F	T
F	F

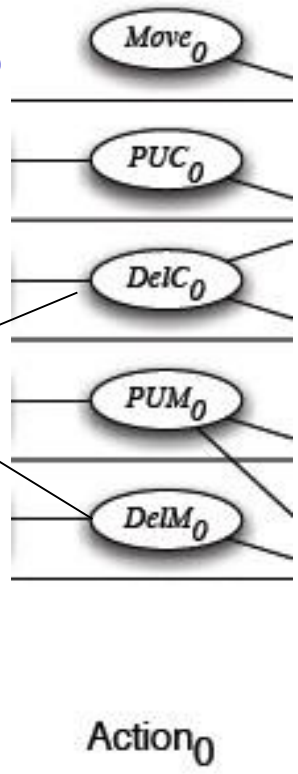
Handling mutex constraints in Forward P

If we don't want DelMand DelC to occur simultaneously

C.

D.

$DelM_i$	$DelC_i$
T	F
F	T
F	F

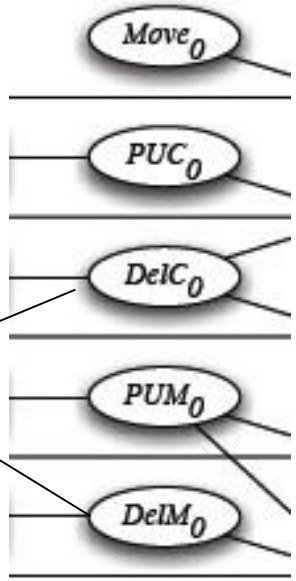


Handling mutex constraints in Forward Planning

E.g., let's say we don't want DelMand DelC to occur simultaneously

How would we encode this into STRIPS for forward planning?

$DelM_i$	$DelC_i$
T	F
F	T
F	F



A. Via the actions' preconditions

B. Via the actions' effects

C. No need to enforce this constraint in Forward Planning

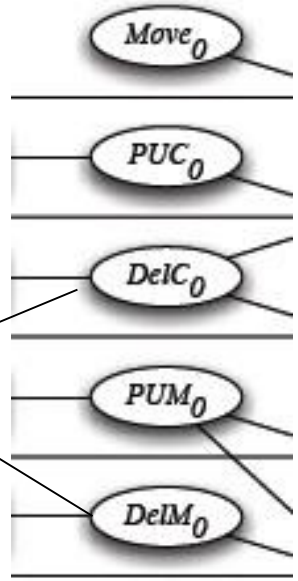
D. None of the above

Handling mutex constraints in Forward Planning

E. g., let's say we don't want DelMand DelC to occur simultaneously

How would we encode this into STRIPS for forward planning?

$DelM_i$	$DelC_i$
T	F
F	T
F	F



No need to enforce this constraint in Forward Planning

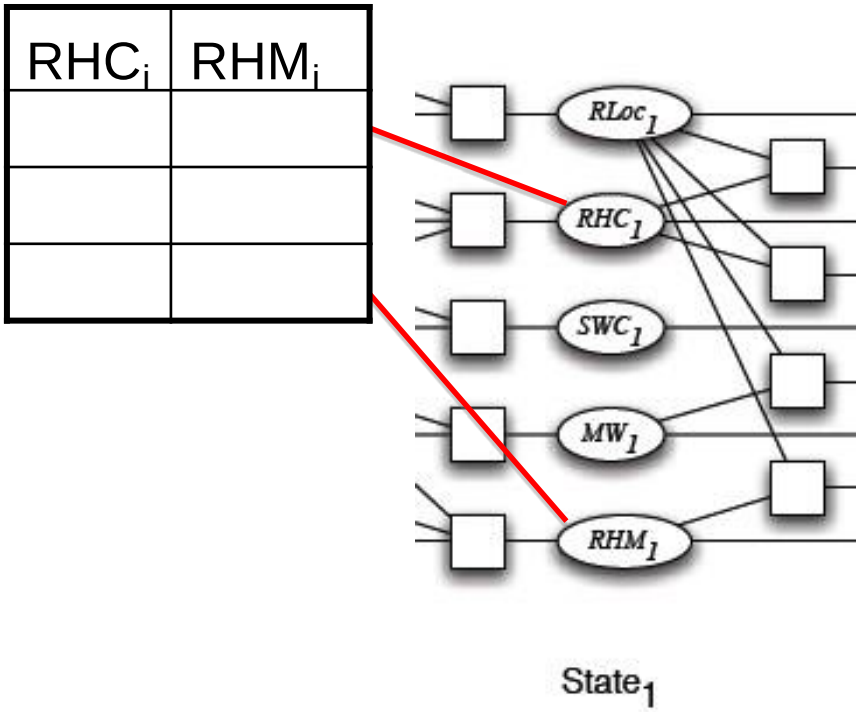
Handling mutex constraints in Forward Planning

Because forward planning gives us an ordered sequence of actions: only one action is carried out at one time

Additional constraints in CSP Planning

Other constraints we may want are state constraints

- hold between variables at the same time step
- they can capture physical constraints of the system (e.g., robot cannot hold coffee and mail)

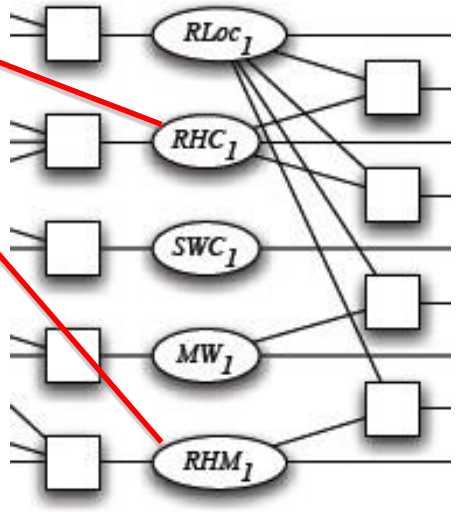


Additional constraints in CSP Planning

Other constraints we may want are state constraints

- hold between variables at the same time step
- they can capture physical constraints of the system (e.g., robot cannot hold coffee and mail)

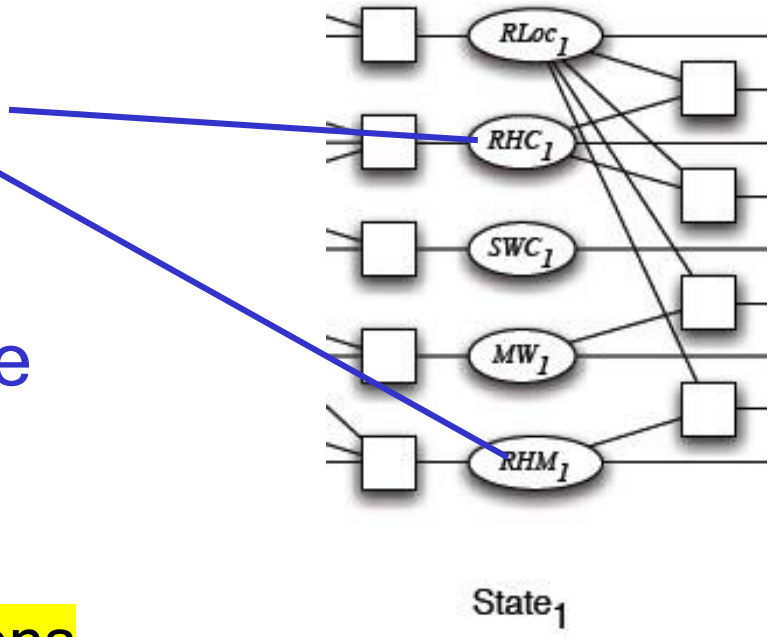
RHC_i	RHM_i
T	F
F	T
F	F



State1

Handling state constraints in Forward Planning

RHC_i	RHM_i
T	F
F	T
F	F



How could we handle these constraints in STRIPS for forward planning?

A. Via the actions' preconditions

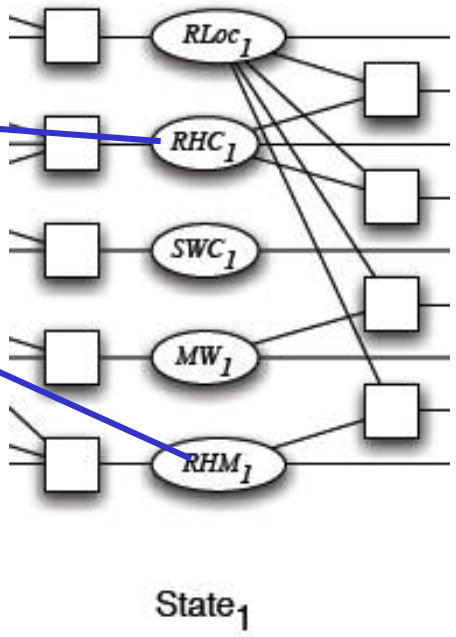
B. Via the actions' effects

C. No need to enforce this constraint in Forward Planning

D. None of the above

Handling state constraints in Forward Planning

RHC_i	RHM_i
T	F
F	T
F	F



How could we handle these constraints in STRIPS for forward planning?

We can use preconditions

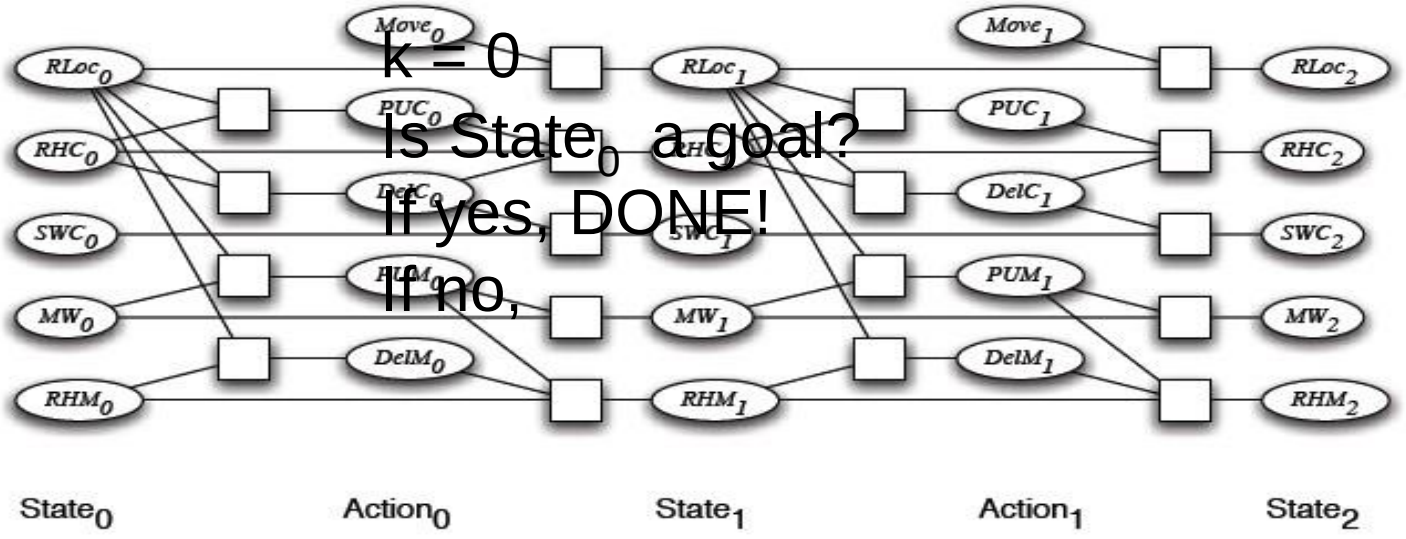
- Robot can pick up coffee only if it does not have coffee and it does not have mail
- Robot can pick up mail only if it does not have mail and it does not have coffee

Lecture Overview

- Recap of Lecture 13
- Planning as CSP
  - More details on CSP representation
  - Solving the CSP planning problem
- Intro to Logic

CSP Planning: Solving the problem

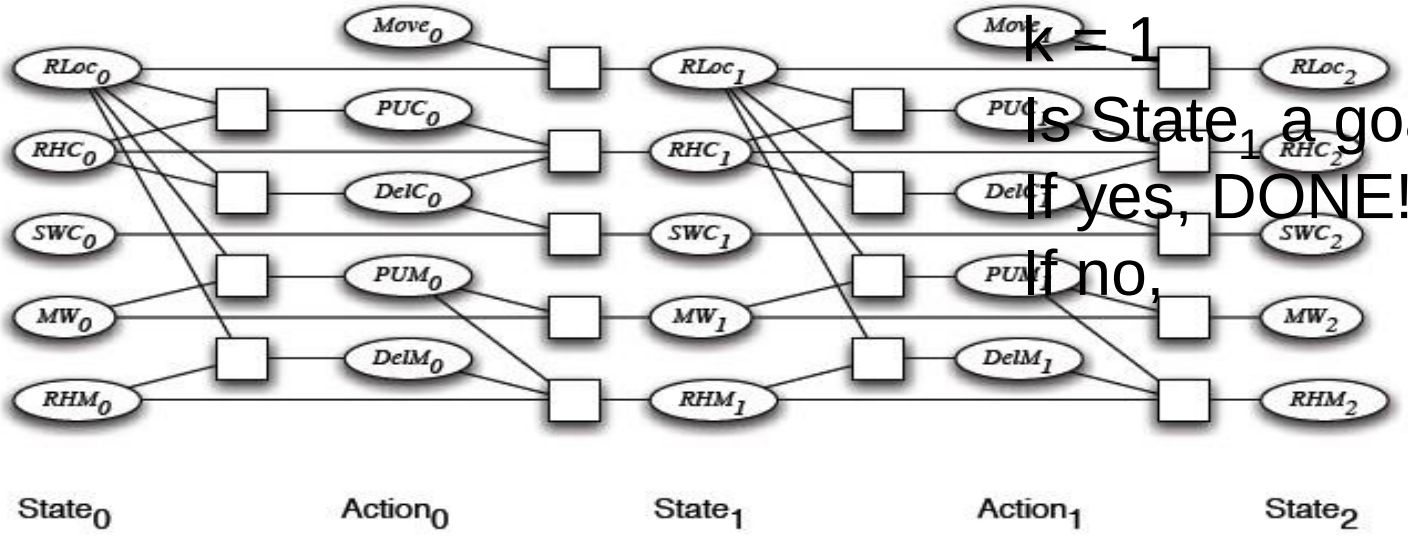
Map STRIPS Representation for horizon 1, 2, 3, …, until solution found



CSP Planning: Solving the problem

Map STRIPS Representation for horizon k =1

Solve the CSP



k = 1

Is State$_1$ a goal

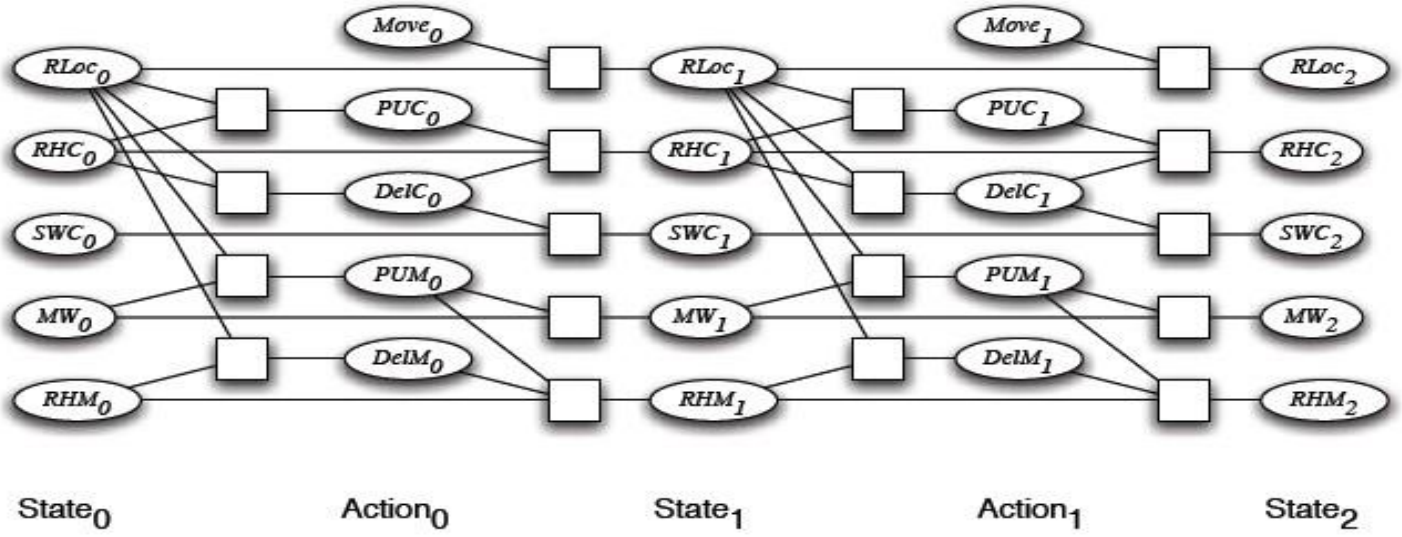
If yes, DONE!

If no,

CSP Planning: Solving the problem

Map STRIPS Representation for horizon k = 2

Solve the CSP



k = 2: Is $State_2$ a goal If
yes, DONE!
If no….continue

Solve Planning as CSP: pseudo code

```
solved = false horizon = 0

While solved = false map STRIPS
   into CSP with horizon solve CSP
   -> solution if solution then
   solved = T
      else horizon = horizon +
           1
```

```
Return solution
```

Solving Planning as CSP: pseudo code

```
solved = false for horizon h=0,1,2,… map
STRIPS into a CSP csp with horizon h solve
that csp if solution exists then return
solution
    else horizon = horizon +
        1
end
```

Which method would you use to solve each of these CSPs?

A Stochastic Local Search

B Arc consistency + domain splitting

Solving Planning as CSP: pseudo code

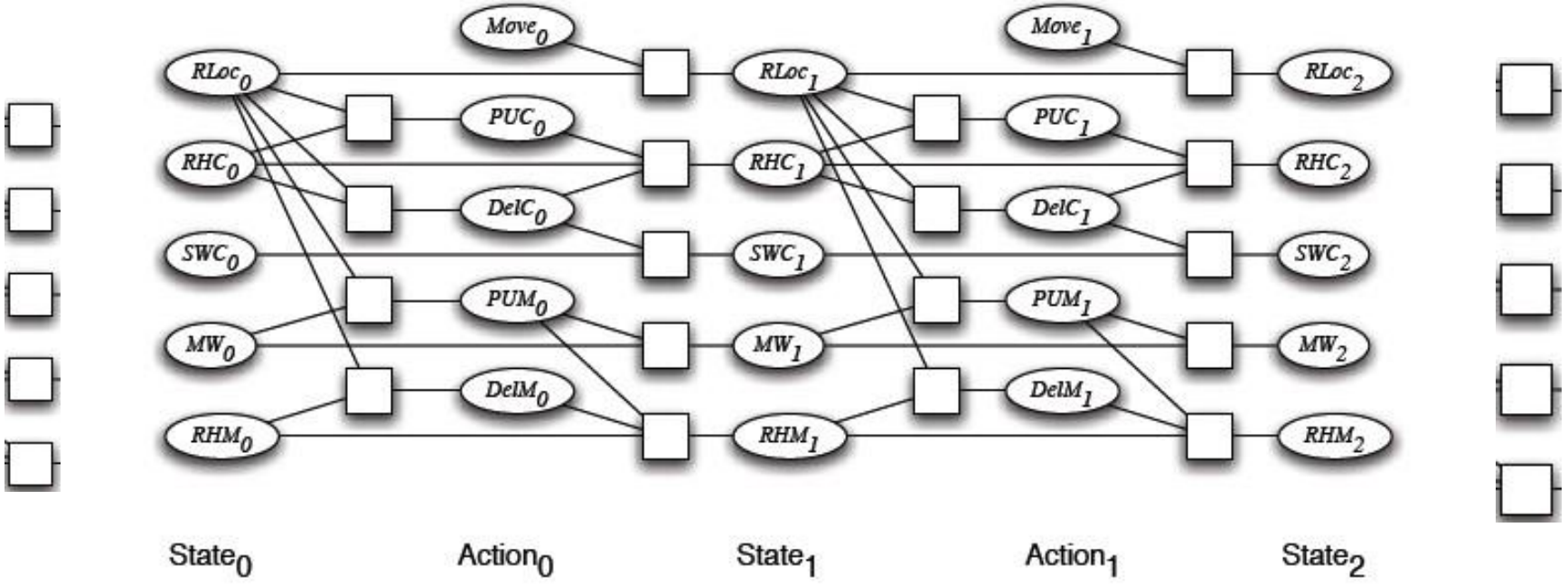
```
solved = false for horizon h=0,1,2,… map
STRIPS into a CSP csp with horizon h solve
that csp if solution exists then return
solution
   else horizon = horizon +
        1
end
```

Which method would you use to solve each of these CSPs?

Arc consistency + domain splitting Not SLS! SLS cannot determine that no solution exists!

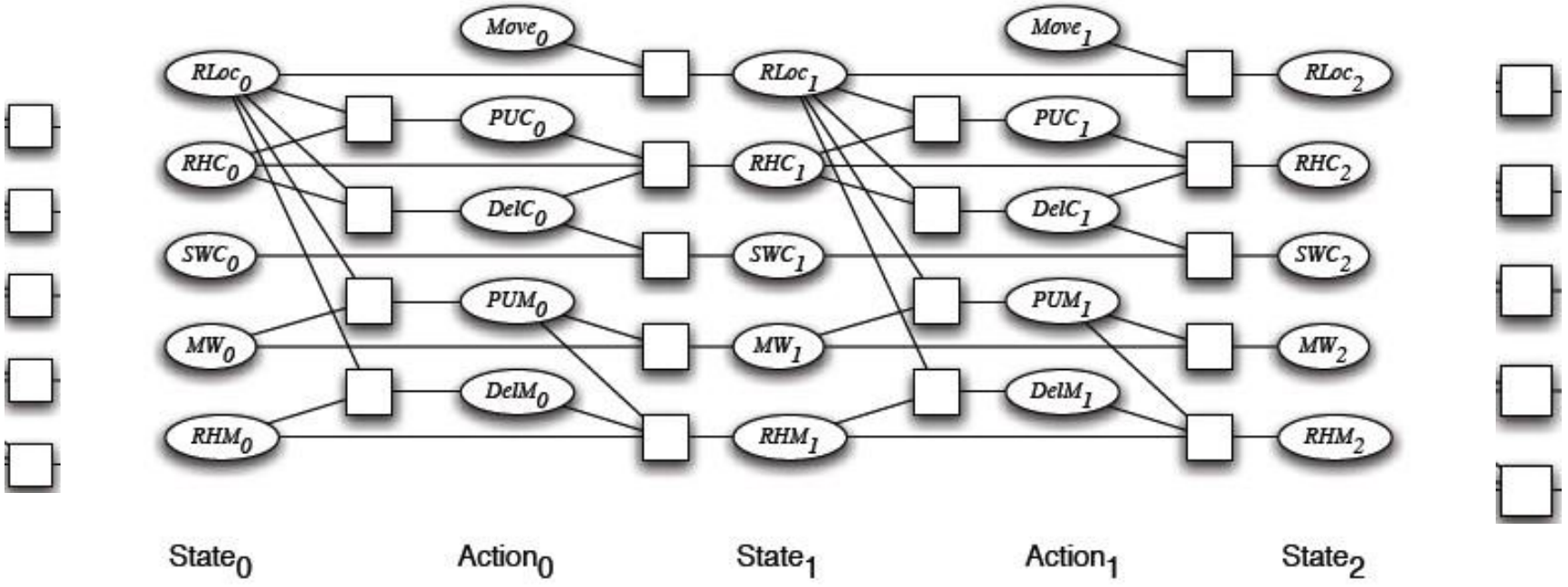
How Does the Solution Look Like?

Namely, how does one find the plan in the constraint graph?



How Does the Solution Look Like?

Namely, how does one find the plan in the constraint graph?

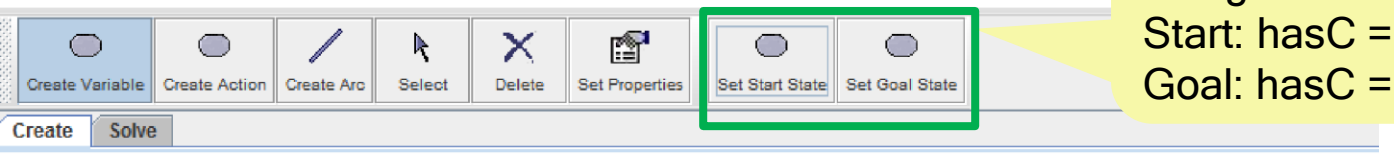


Look for all the action variables that have value T

STRIPS to CSP applet

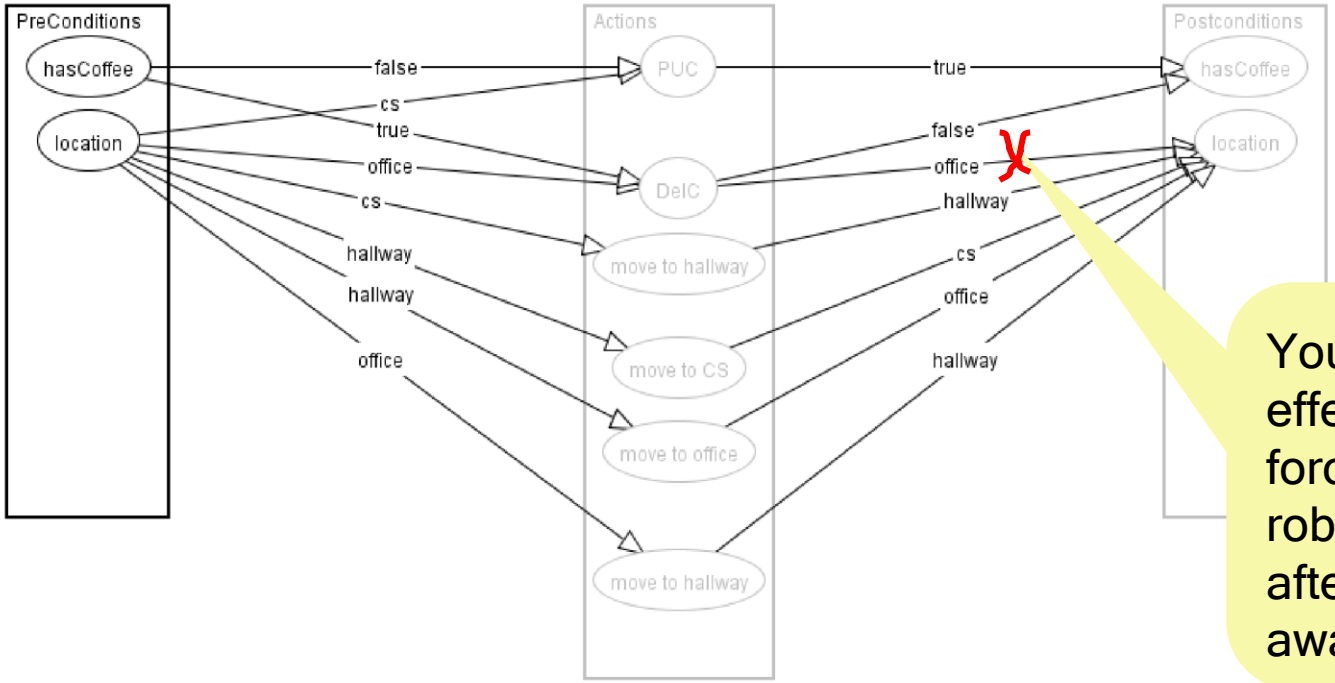
Allows you to:

1. specify a planning problem in STRIPS
2. map it into a CSP for a given horizon
3. the CSP translation is automatically loaded into the CSP applet where it can be solved

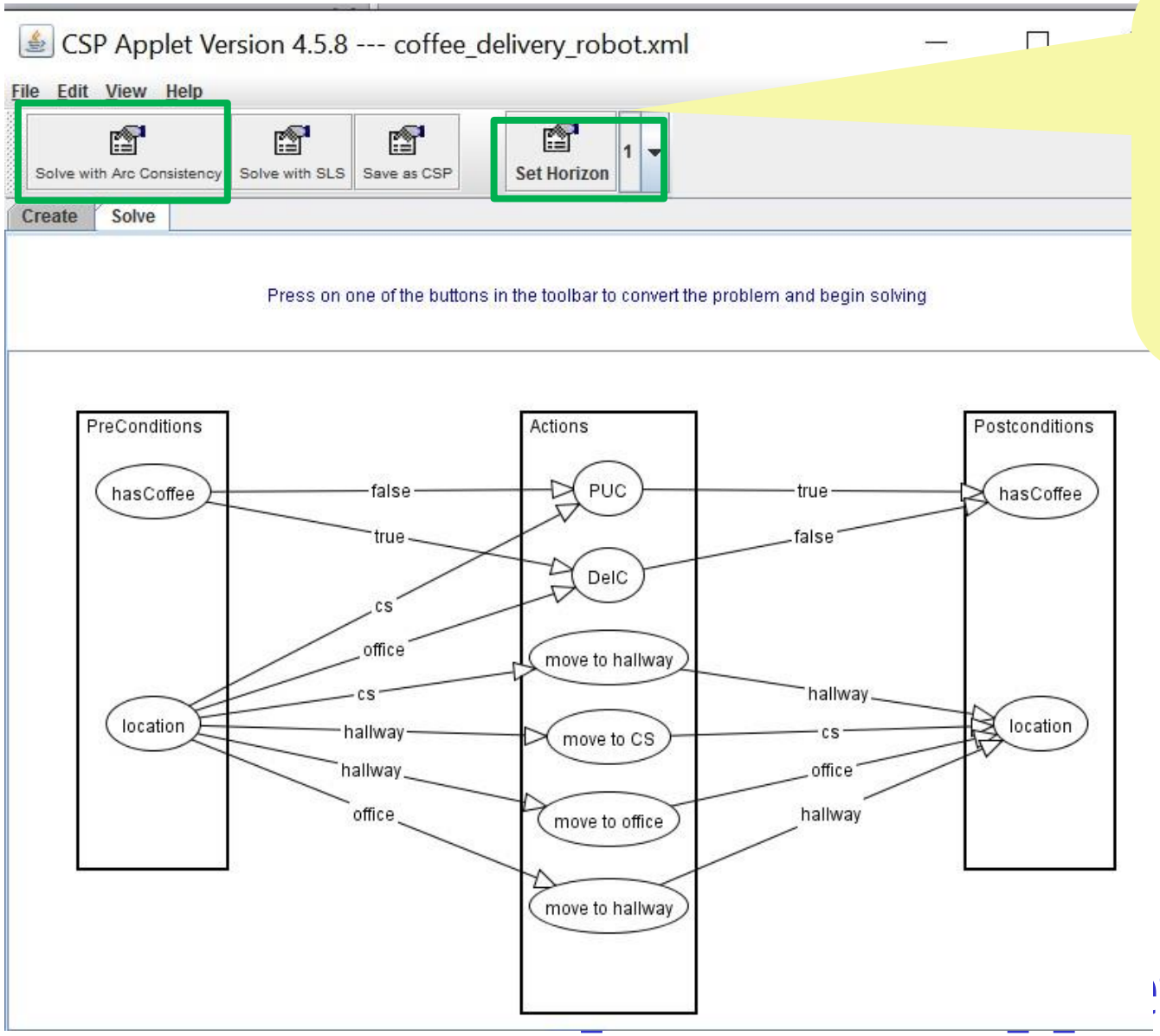


In “Create” mode you can set the start and goal states, e.g.
Start: hasC = T, loc = cs
Goal: hasC = F

Visual STRIPS representation of a simplified version of the delivery robot problem

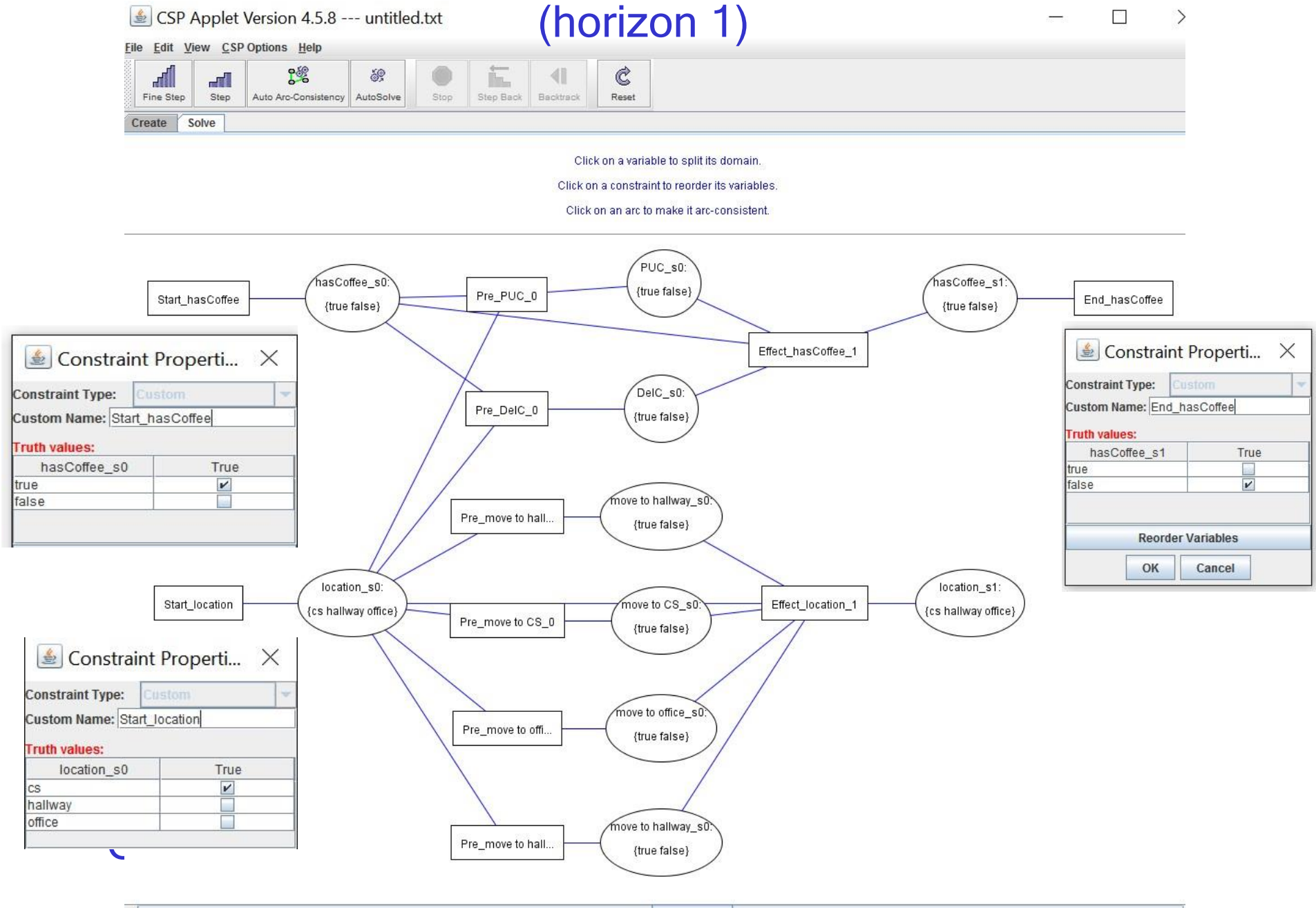


You should delete this effect of DelC which forces the position of the robot to stay in the office after giving the coffee away

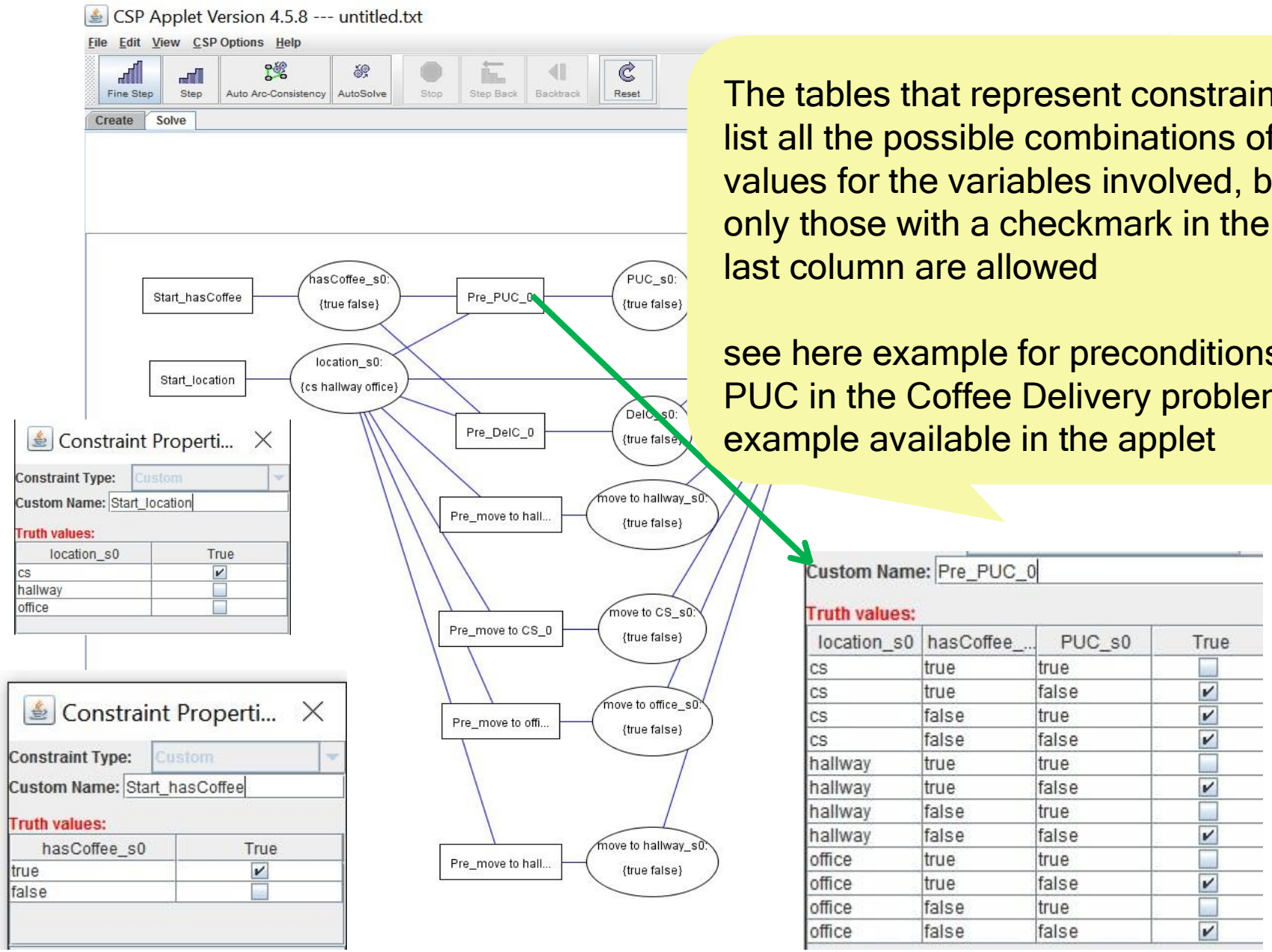


In “Solve” mode, set the horizon and then press “Solve with Arc Consistency” to open the CSP applet with the CSP representation of this problem

(horizon 1)

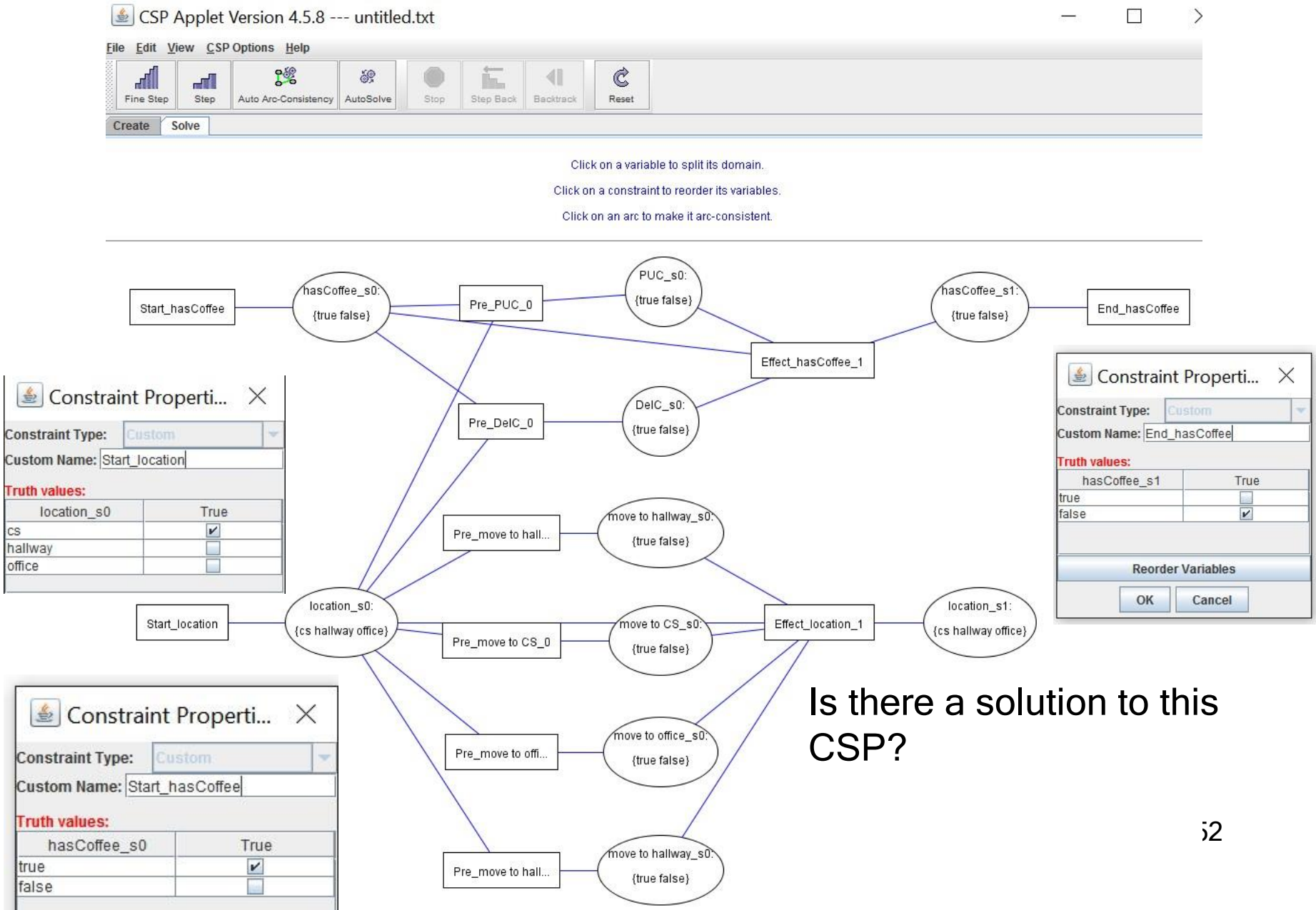

CSP Applet Version 4.5.8 --- untitled.txt
File Edit View CSP Options Help
Fine Step
Step
Auto Arc-Consistency
AutoSolve
Stop
Step Back
Backtrack
Reset
Create
Solve
Click on a variable to split its domain.
Click on a constraint to reorder its variables.
Click on an arc to make it arc-consistent.
Start_hasCoffee
hasCoffee_s0: {true false}
Pre_PUC_0
PUC_s0: {true false}
Effect_hasCoffee_1
hasCoffee_s1: {true false}
End_hasCoffee
Pre_DelC_0
DelC_s0: {true false}
Pre_move to hall...
move to hallway_s0: {true false}
Start_location
location_s0: {cs hallway office}
Pre_move to CS_0
move to CS_s0: {true false}
Effect_location_1
location_s1: {cs hallway office}
Pre_move to offi...
move to office_s0: {true false}
Pre_move to hall...
move to hallway_s0: {true false}
Constraint Properti...
Constraint Type: Custom
Custom Name: Start_hasCoffee
Truth values:
hasCoffee_s0 | True
true
false
Constraint Properti...
Constraint Type: Custom
Custom Name: End_hasCoffee
Truth values:
hasCoffee_s1 | True
true
false
Reorder Variables
OK
Cancel
Constraint Properti...
Constraint Type: Custom
Custom Name: Start_location
Truth values:
location_s0 | True
cs
hallway
office

CPS version of the planning problem in the CSP applet (horizon 1)

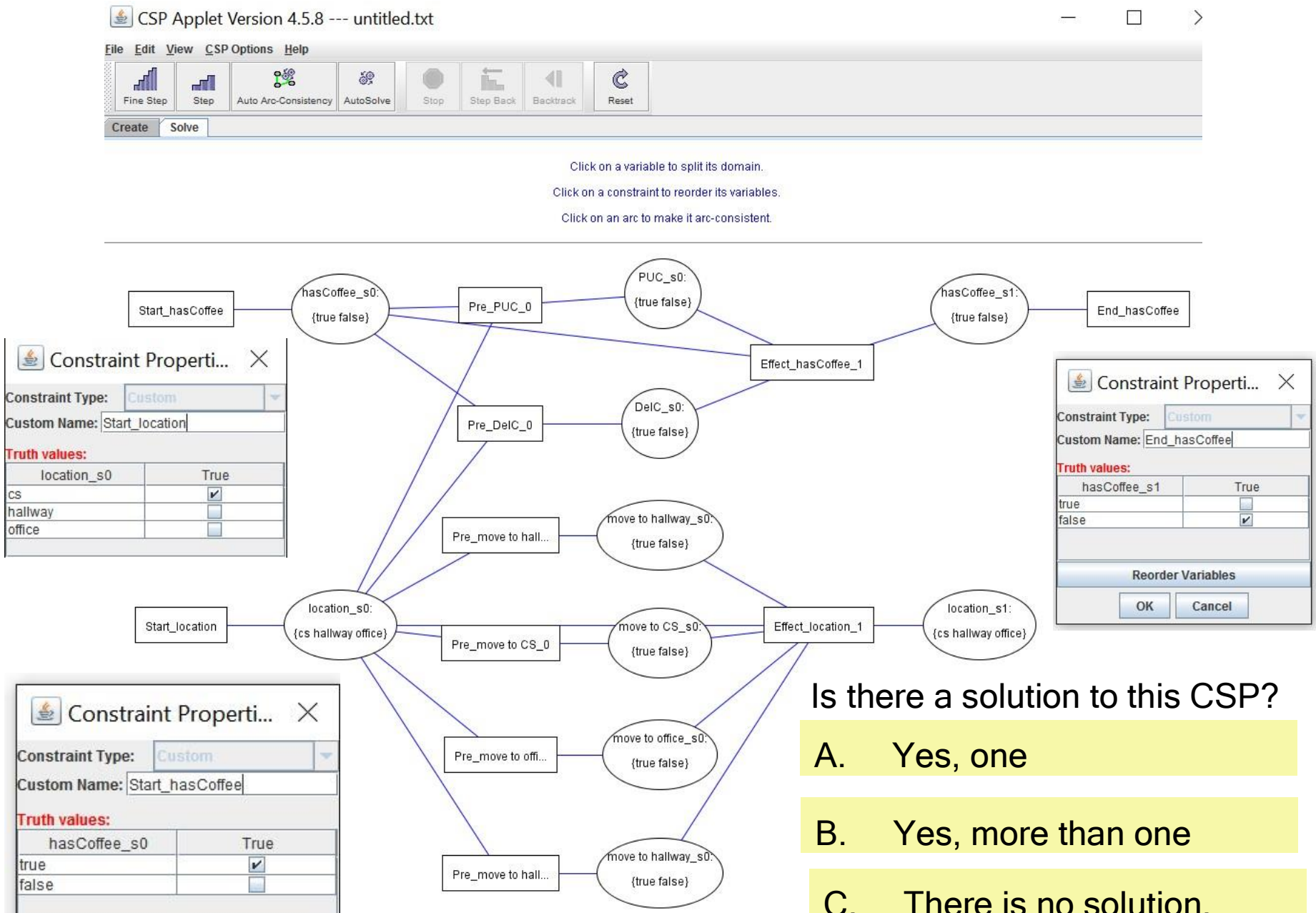


The tables that represent constraints list all the possible combinations of values for the variables involved, but only those with a checkmark in the last column are allowed

see here example for preconditions of PUC in the Coffee Delivery problem example available in the applet



Is there a solution to this CSP?

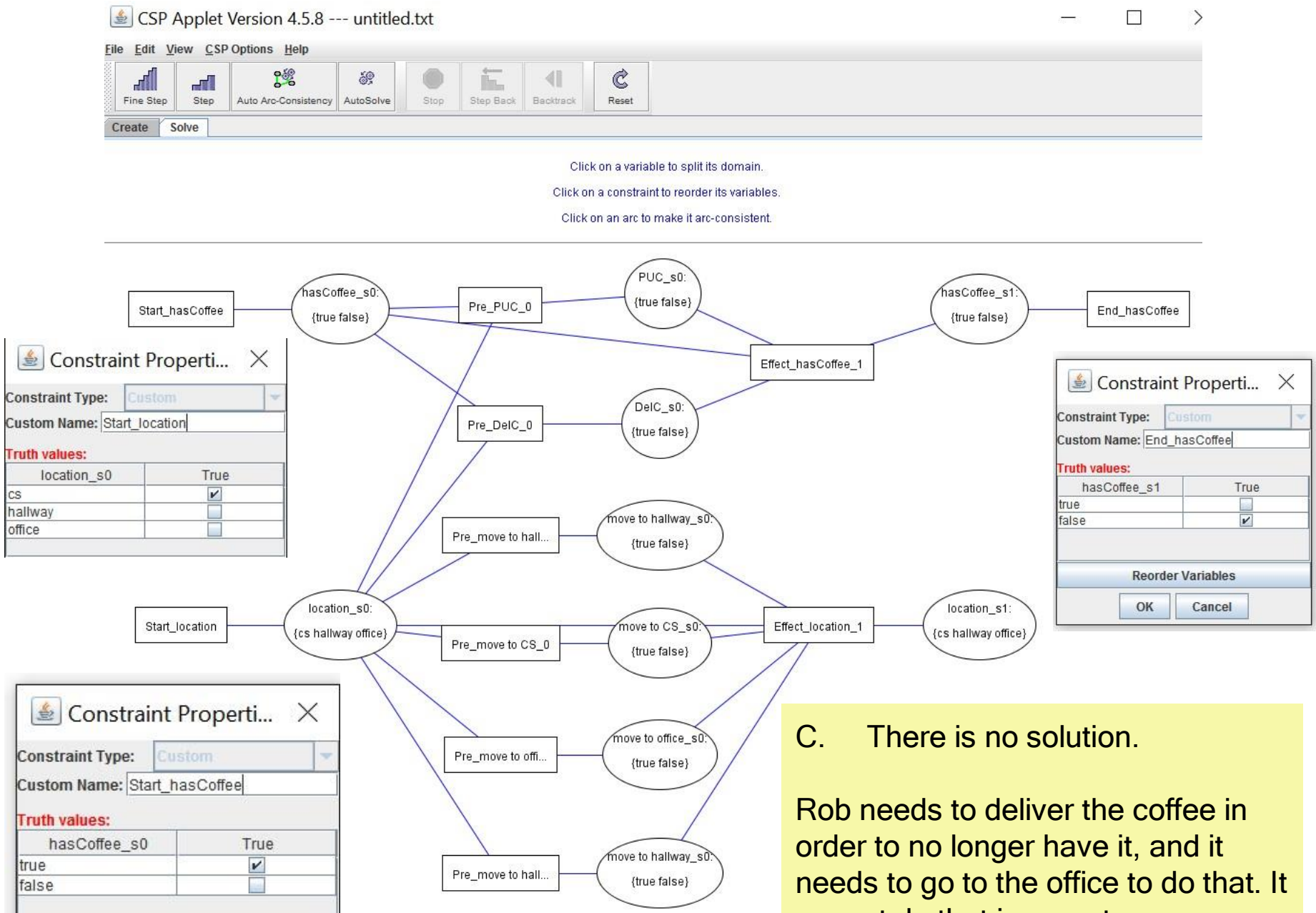


Is there a solution to this CSP?

A. Yes, one

B. Yes, more than one

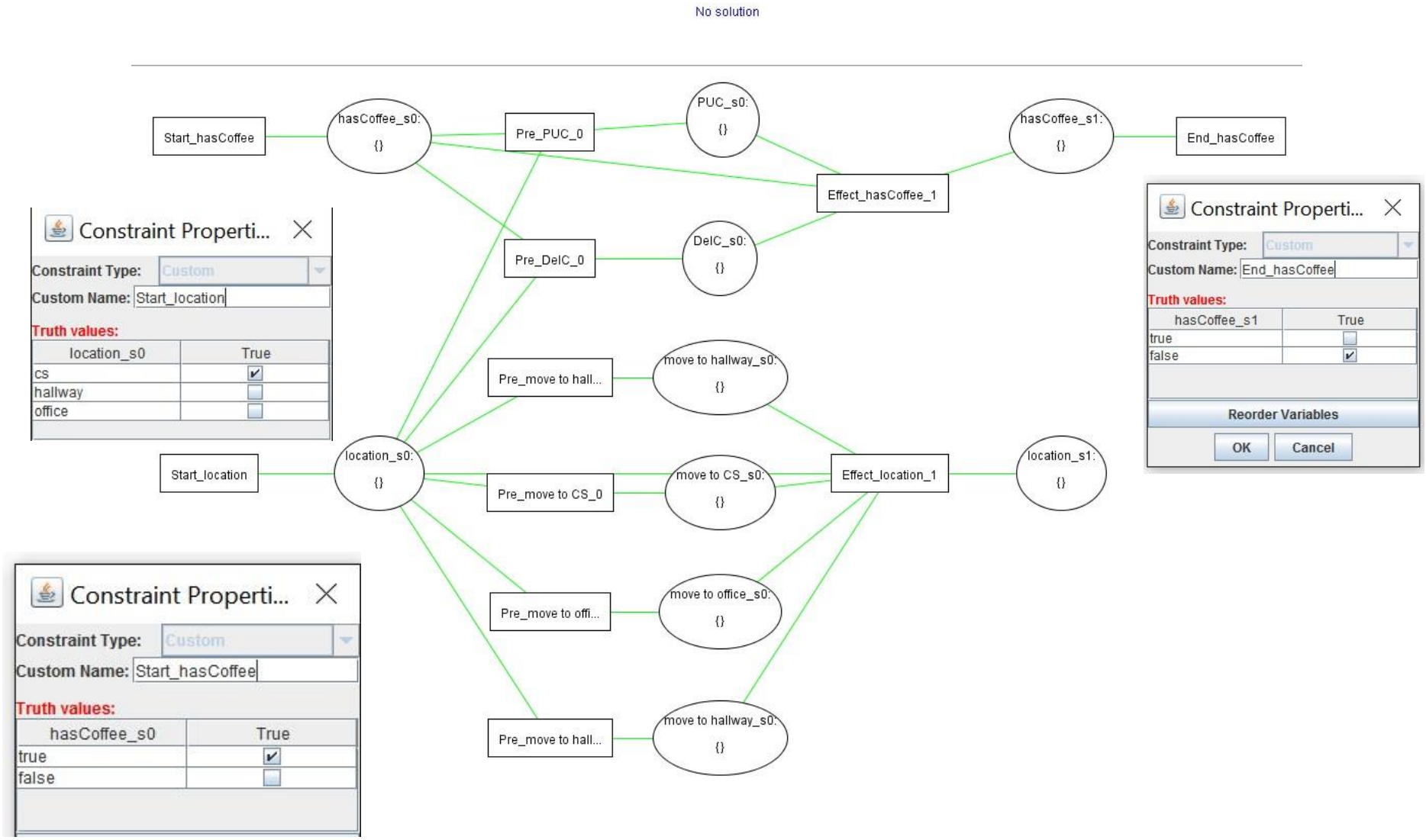
C. There is no solution,

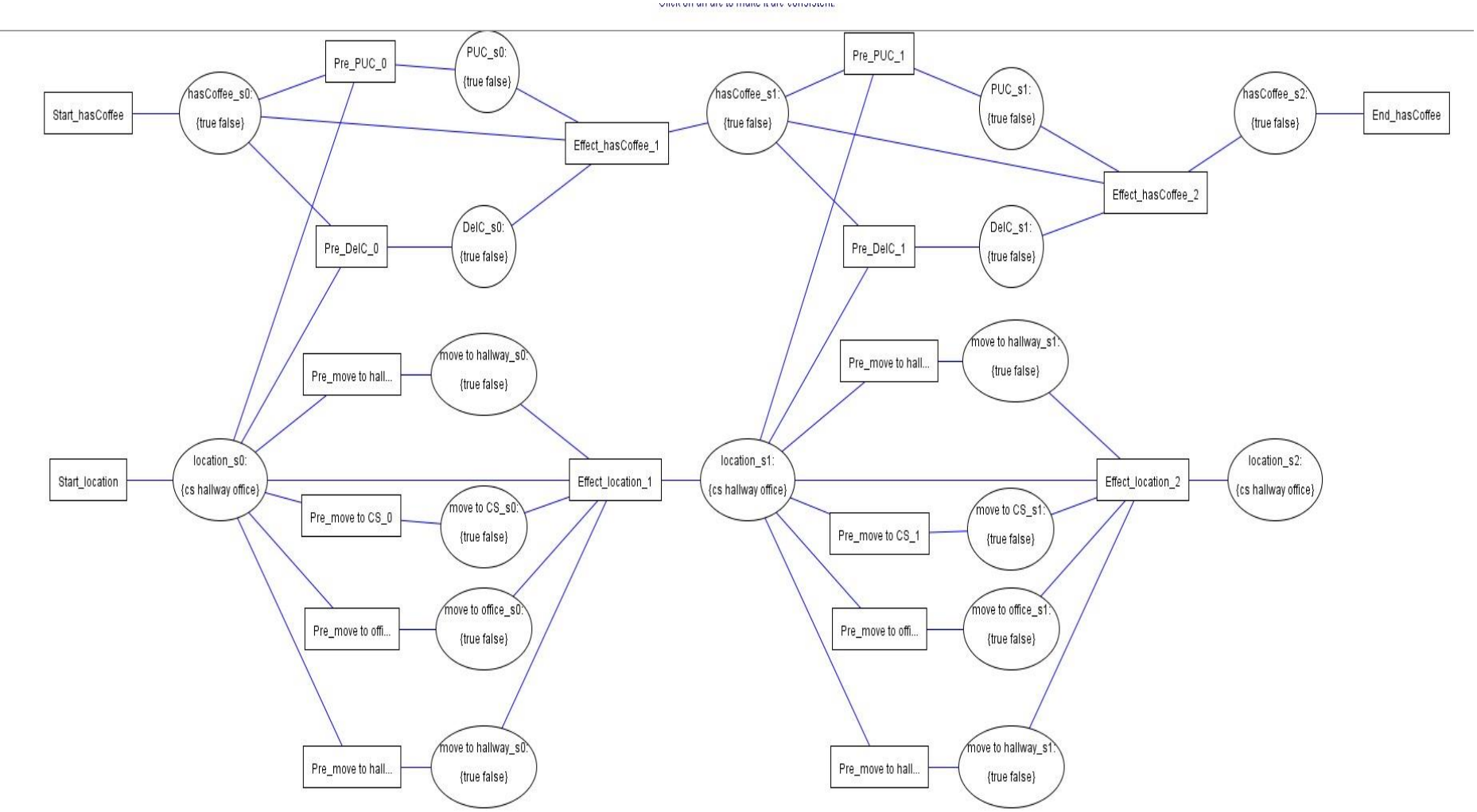


C. There is no solution.

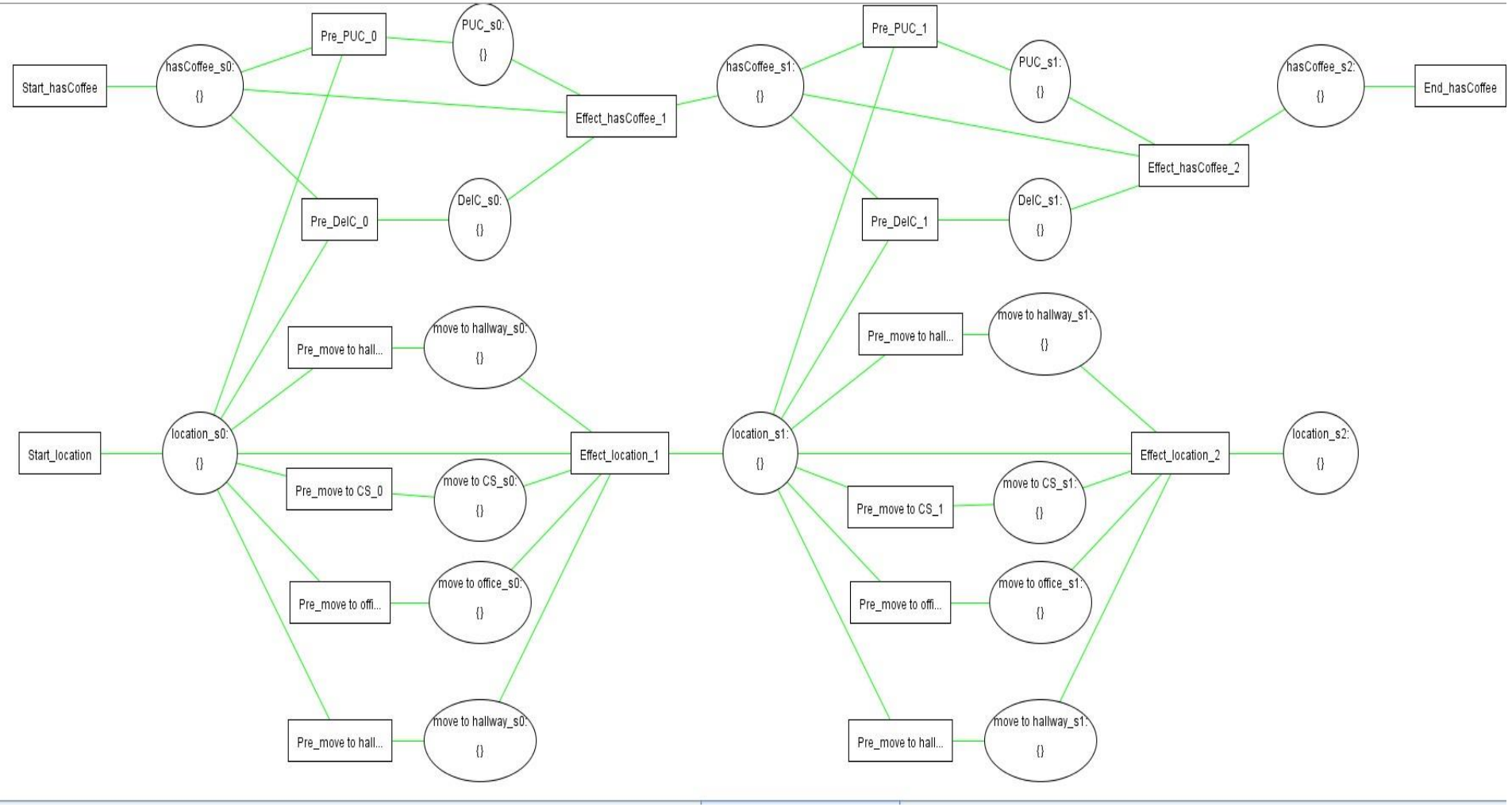
Rob needs to deliver the coffee in order to no longer have it, and it needs to go to the office to do that. It cannot do that in one step.

If you run arc consistency....


No solution
Start_hasCoffee
hasCoffee_s0:
{}
Pre_PUC_0
PUC_s0:
{}
Effect_hasCoffee_1
hasCoffee_s1:
{}
End_hasCoffee
Pre_DelC_0
DelC_s0:
{}
Constraint Properti...
Constraint Type: Custom
Custom Name: Start_location
Truth values:
location_s0 | True
cs | ✔
hallway |
office |
Constraint Properti...
Constraint Type: Custom
Custom Name: End_hasCoffee
Truth values:
hasCoffee_s1 | True
true |
false | ✔
Reorder Variables
OK
Cancel
Pre_move to hall...
move to hallway_s0:
{}
Start_location
location_s0:
{}
Pre_move to CS_0
move to CS_s0:
{}
Effect_location_1
location_s1:
{}
Pre_move to offi...
move to office_s0:
{}
Pre_move to hall...
move to hallway_s0:
{}
Constraint Properti...
Constraint Type: Custom
Custom Name: Start_hasCoffee
Truth values:
hasCoffee_s0 | True
true | ✔
false |

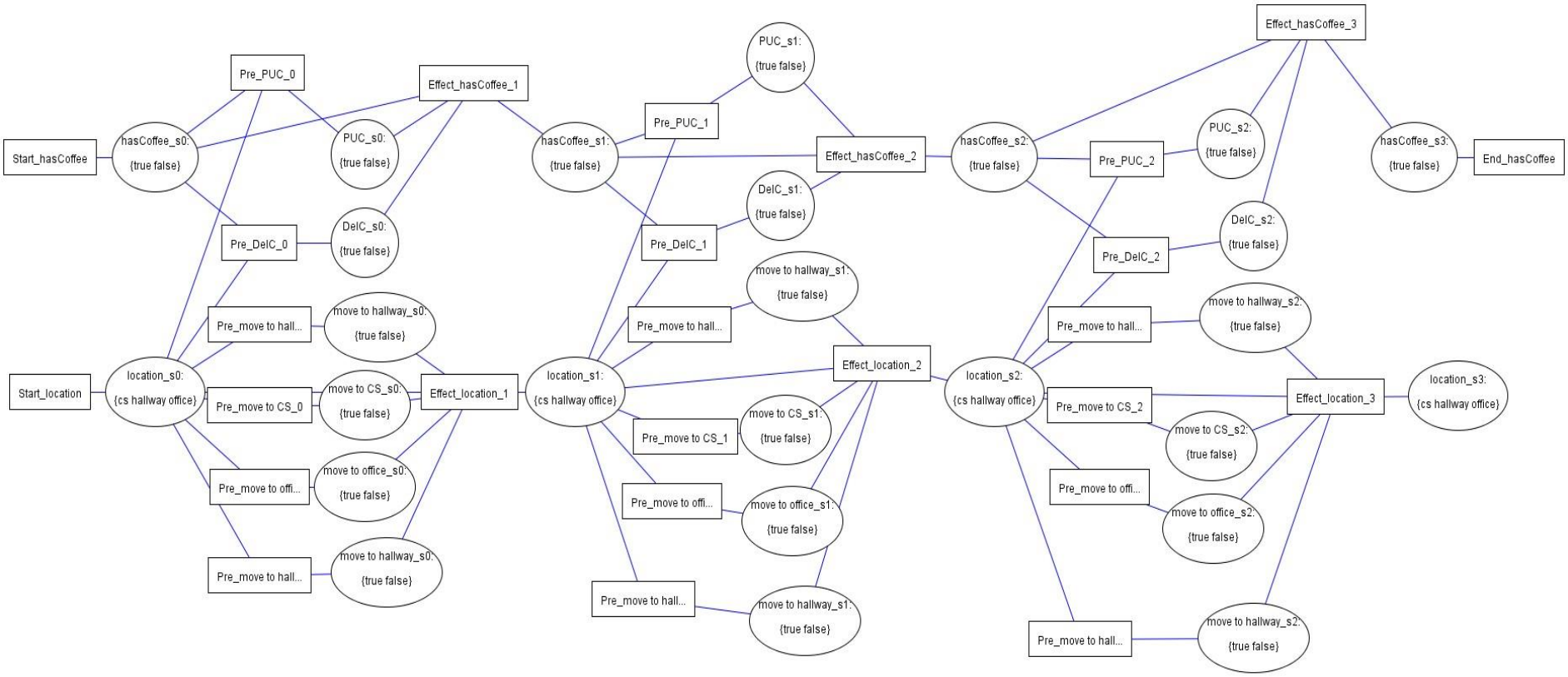


Same problem with Horizon 2



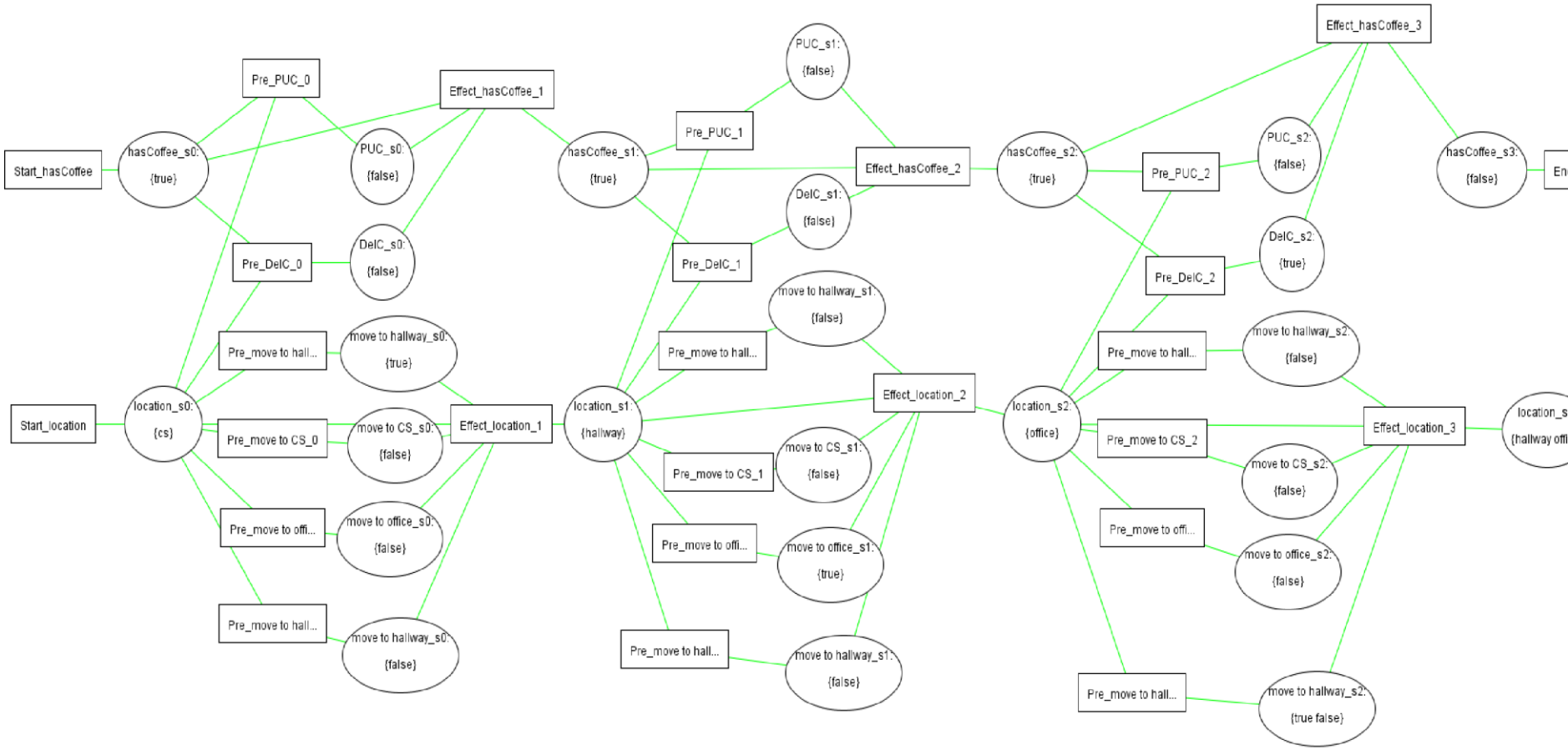
Same problem with Horizon 2

Rob needs to deliver the coffee in order to no longer have it. It needs to go to the office to do that. To go to the office it needs to go to the hallway first. Cannot do all this in two steps


Start_hasCoffee
hasCoffee_s0: {true false}
Pre_PUC_0
PUC_s0: {true false}
Effect_hasCoffee_1
Pre_DelC_0
DelC_s0: {true false}
Pre_move to hall...
move to hallway_s0: {true false}
Start_location
location_s0: {cs hallway office}
Pre_move to CS_0
move to CS_s0: {true false}
Effect_location_1
Pre_move to offi...
move to office_s0: {true false}
Pre_move to hall...
move to hallway_s0: {true false}
hasCoffee_s1: {true false}
PUC_s1: {true false}
Pre_PUC_1
Effect_hasCoffee_2
DelC_s1: {true false}
Pre_DelC_1
move to hallway_s1: {true false}
Pre_move to hall...
location_s1: {cs hallway office}
Effect_location_2
Pre_move to CS_1
move to CS_s1: {true false}
Pre_move to offi...
move to office_s1: {true false}
Pre_move to hall...
move to hallway_s1: {true false}
Effect_hasCoffee_3
hasCoffee_s2: {true false}
Pre_PUC_2
PUC_s2: {true false}
hasCoffee_s3: {true false}
End_hasCoffee
Pre_DelC_2
DelC_s2: {true false}
Pre_move to hall...
move to hallway_s2: {true false}
location_s2: {cs hallway office}
Pre_move to CS_2
Effect_location_3
location_s3: {cs hallway office}
move to CS_s2: {true false}
Pre_move to offi...
move to office_s2: {true false}
Pre_move to hall...
move to hallway_s2: {true false}

Horizon 3

Do we have a solution to this CSP?


Start_hasCoffee
hasCoffee_s0: {true}
Pre_PUC_0
PUC_s0: {false}
Effect_hasCoffee_1
Pre_DelC_0
DelC_s0: {false}
Start_location
location_s0: {cs}
Pre_move to hall...
move to hallway_s0: {true}
Pre_move to CS_0
move to CS_s0: {false}
Pre_move to offi...
move to office_s0: {false}
Pre_move to hall...
move to hallway_s0: {false}
Effect_location_1
hasCoffee_s1: {true}
PUC_s1: {false}
Pre_PUC_1
Effect_hasCoffee_2
DelC_s1: {false}
Pre_DelC_1
move to hallway_s1: {false}
Pre_move to hall...
location_s1: {hallway}
Effect_location_2
Pre_move to CS_1
move to CS_s1: {false}
Pre_move to offi...
move to office_s1: {true}
Pre_move to hall...
move to hallway_s1: {false}
Effect_hasCoffee_3
hasCoffee_s2: {true}
Pre_PUC_2
PUC_s2: {false}
hasCoffee_s3: {false}
Pre_DelC_2
DelC_s2: {true}
Pre_move to hall...
move to hallway_s2: {false}
location_s2: {office}
Pre_move to CS_2
Effect_location_3
move to CS_s2: {false}
Pre_move to offi...
move to office_s2: {false}
Pre_move to hall...
move to hallway_s2: {true false}

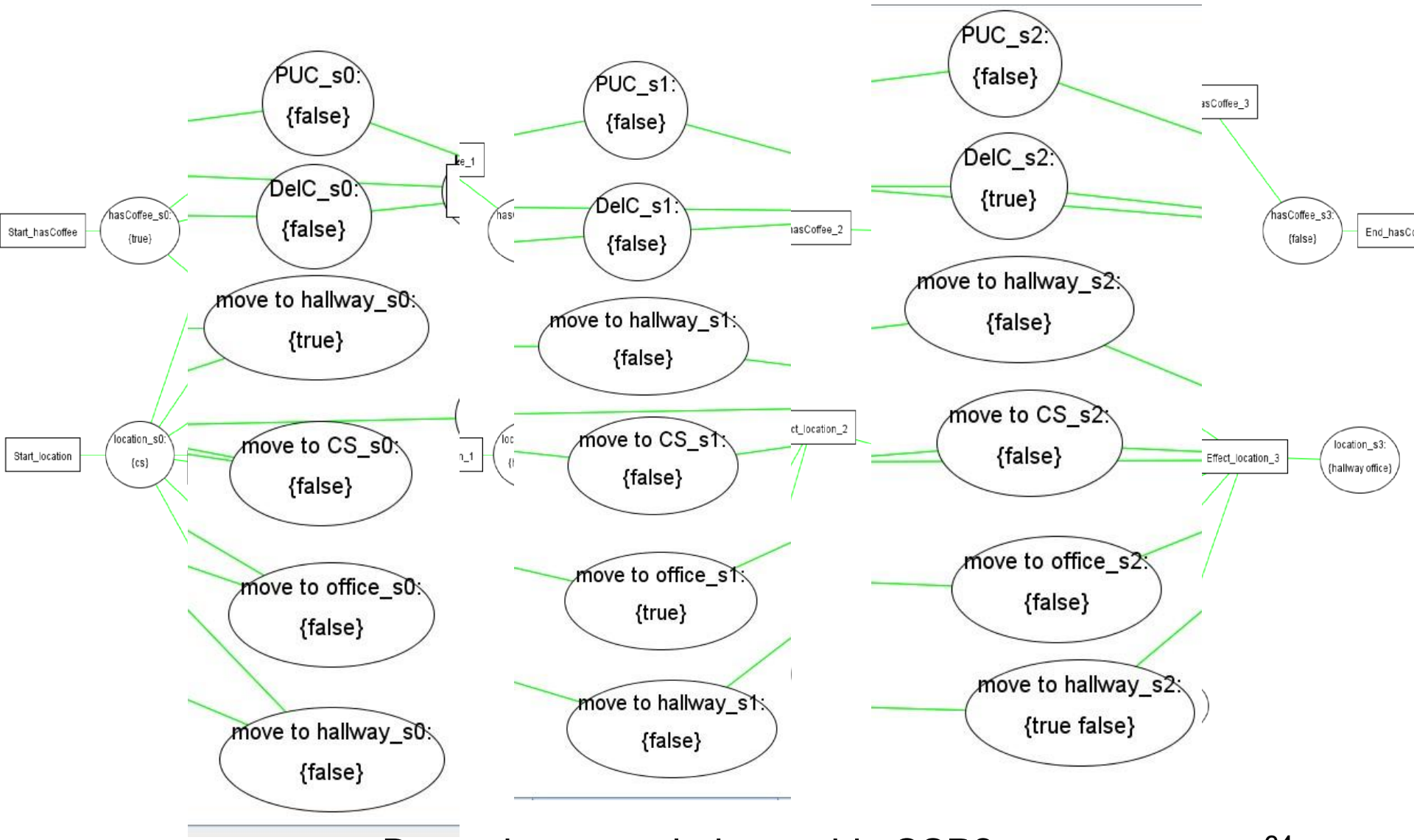
Horizon 3

Do we have a solution to this CSP?

A. Yes, exactly one

B. Yes, more than one

C. There is no solution

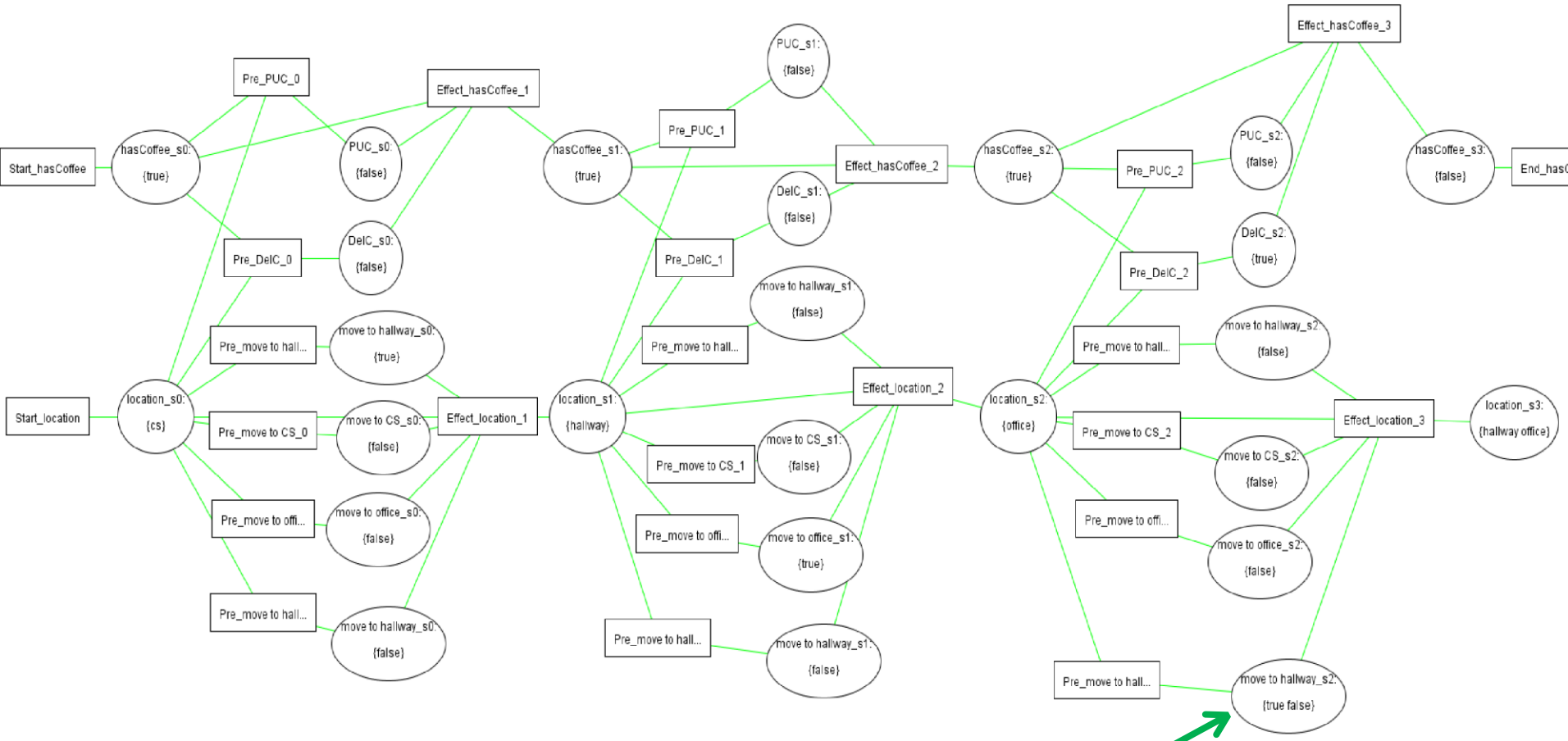


Do we have a solution to this CSP?

A. Yes, exactly one

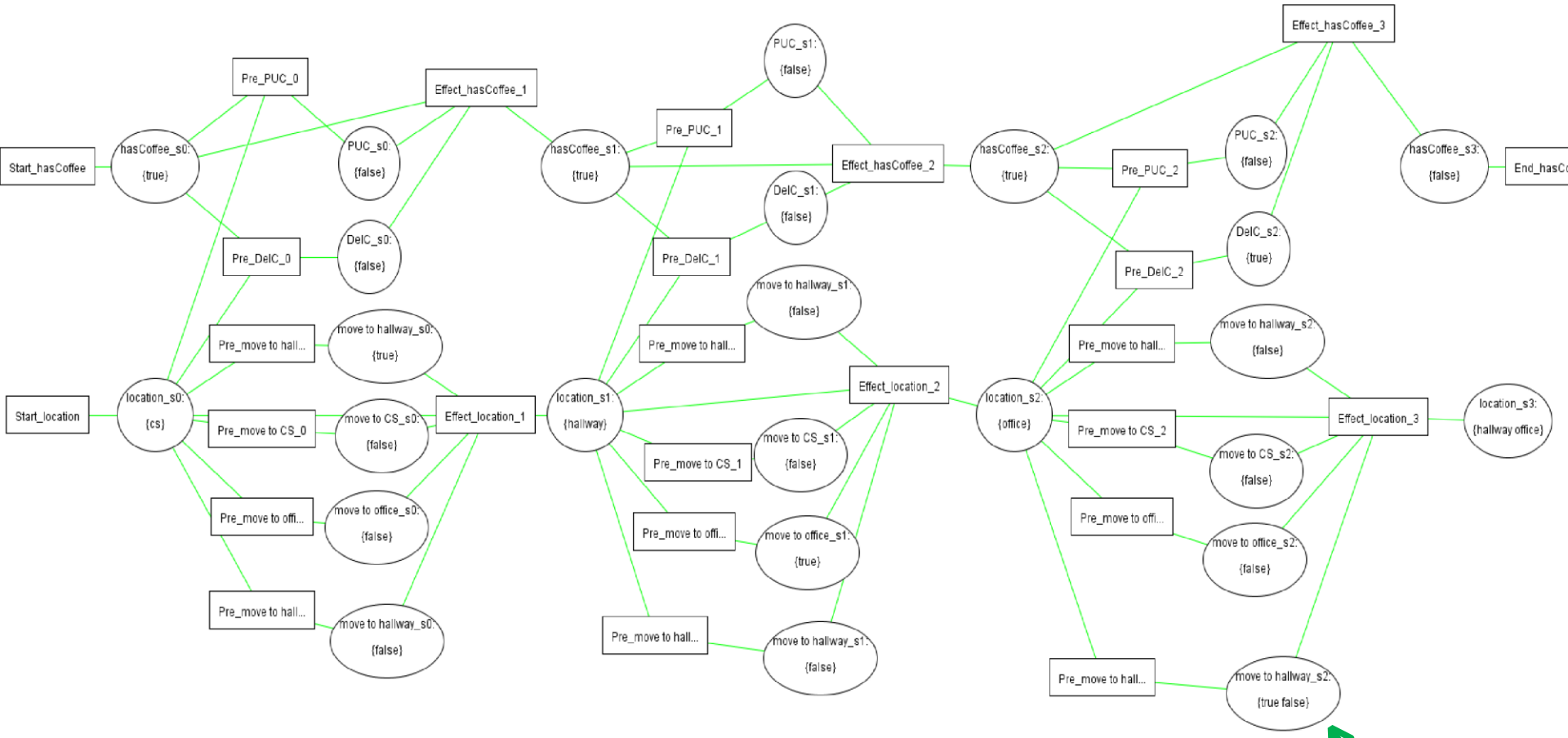
B. Yes, more than one

C. There is no solution



B. Yes, more than one: one with move_to_hallway_s2 = T and one with move_to_hallway_s2 = F

WHY?



Because both Move_to_hallway and Del_Coffee have as precondition Loc = cs, and there are no mutex constraints that prevent them from happening at the same time

The two solutions are found with domain splitting on move_to_hallway_s2

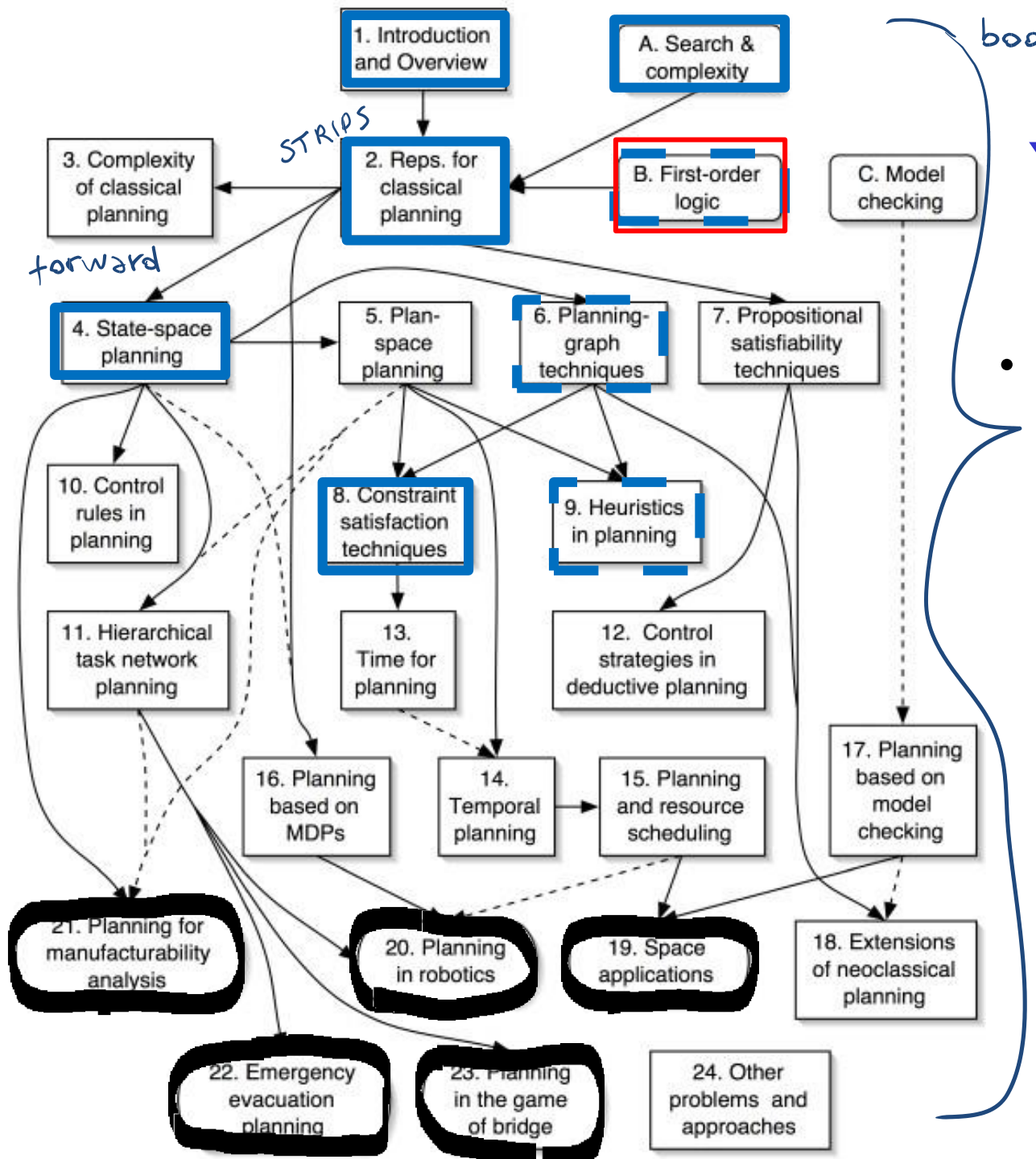
Think about what happens with horizons higher than 3

State of the art planner

- A similar process is implemented (more efficiently) in the Graphplan planner

- In general, Planning graphs are an efficient way to create a representation of a planning problem that can be used to

- Achieve better heuristic estimates
- Directly construct plans



You know the key ideas!

- Ghallab, Nau, and Traverso
  Automated Planning: Theory and Practice

Web site:

- ✓ http://www.laas.fr/planning
- ✓ Also has lecture notes

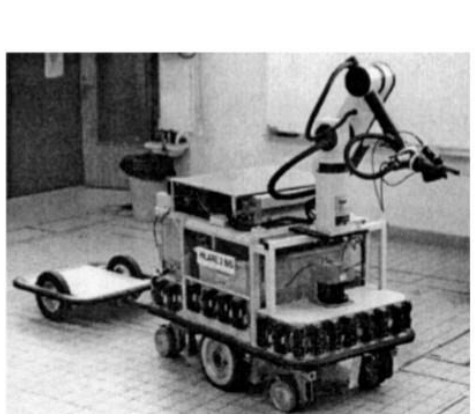
You know

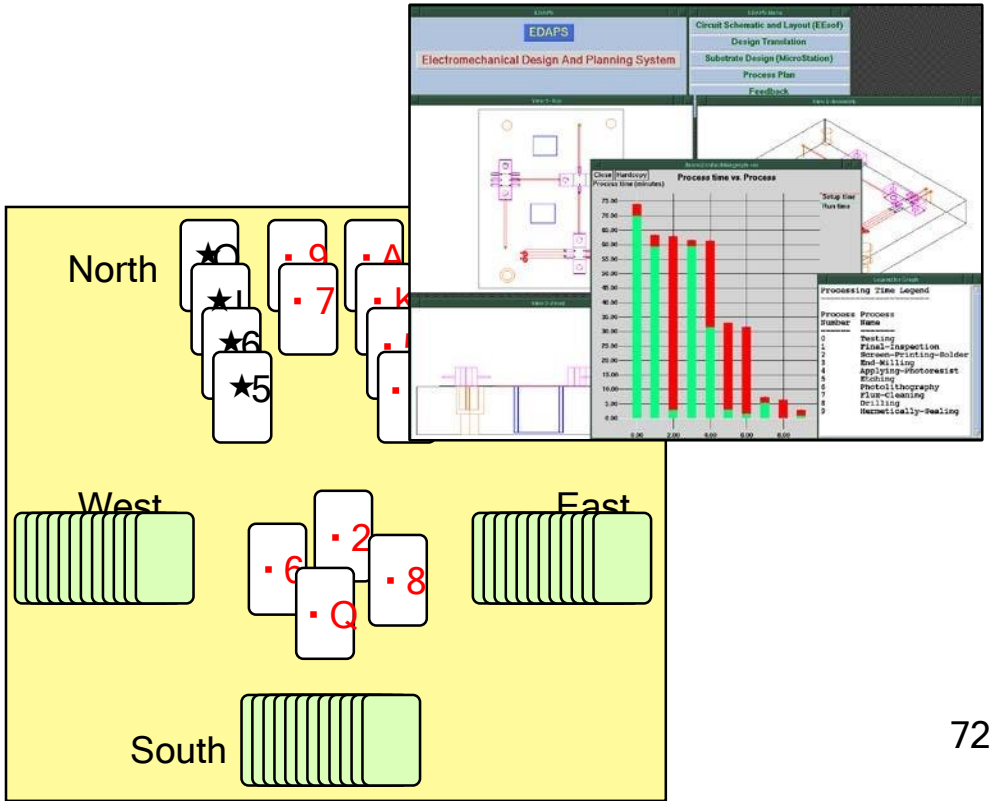
You know a little

Applications

Some applications of planning

- Emergency Evacuation
- Robotics
- Space Exploration
- Manufacturing Analysis
- Games (e.g., Bridge)
- Generating Natural language
- Product Recommendations ….





Learning Goals for Planning

- STRIPS
- Represent a planning problem with the STRIPS representation
- Explain the STRIPS assumption

- Forward planning

- Solve a planning problem by search (forward planning). Specify states, successor function, goal test and solution.
- Construct and justify a heuristic function for forward planning

- CSP planning
- Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
- Solve a planning problem with CSP by expanding the horizon.

On to the next topic: logic!