Lecture 14
Planning Wrap Up
Announcements

Midterm mark computed as follows

\[ I_{Q1} + \max \left( I_{rest}, \ 0.85 \times I_{rest} + 0.15 \times G_{rest} \right) \]

\( I_{Q1} \) = total mark obtained on Q1 done individually

\( I_{rest} \) = total mark obtained on Q2, Q3 and Q4 done individually

\( G_{rest} \) = total mark obtained on Q2, Q3 and Q3 done in group

Average 77%, Max 100%, Min 35%.

- Discussion of solutions for problem questions have been posted in Connect
  - Not always the actual solution, but ways to derive it
- Solutions for the short questions are in the course material

Assign 3 will be posted by Thursday

- You can already do the first 2 questions
Recap of Lecture 13

• Planning as CSP
  - Details on CSP representation
  - Solving the CSP planning problem

• Intro to Logic
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
Course Overview

Environment

Deterministic
- Arc Consistency
- Search

Stochastic
- Belief Nets
  - Variable Elimination
- Decision Nets
  - Variable Elimination
- Markov Processes
  - Value Iteration

Problem Type

Static
- Constraint Satisfaction

Sequential
- Query
- Planning

Representation

Reasoning Technique

Logics

STRIPS
- Search

Vars + Constraints
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

![Diagram of CSP variables and actions](image-url)
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 $k - 1$. 

![Diagram of CSP variables and actions]
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.,
  - robot can only pick up coffee when Loc=cs (coffee shop) and RHC = false (don’t have coffee already)

Truth table for this constraint:
- lists allowed combinations of values

<table>
<thead>
<tr>
<th>RLoc₀</th>
<th>RHC₀</th>
<th>PUC₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>cs</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>cs</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>mr</td>
<td>*</td>
<td>F</td>
</tr>
<tr>
<td>lab</td>
<td>*</td>
<td>F</td>
</tr>
<tr>
<td>off</td>
<td>*</td>
<td>F</td>
</tr>
</tbody>
</table>
Lecture Overview

- Recap of Lecture 13
- Planning as CSP
  - More details on CSP representation
    - Solving the CSP planning problem
- Intro to Logic
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!)
• E.g. let’s consider variable RHC at time t and t+1 and fill in a few rows in this table

<table>
<thead>
<tr>
<th>RHC_t</th>
<th>DelC_i</th>
<th>PUC_i</th>
<th>RHC_{t+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Does not quite matter what we put here since other constraints enforce that these configurations won’t happen
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!)
- E.g. let’s consider variable $RHC$ at time $t$ and $t+1$ and fill in a few rows in this table

<table>
<thead>
<tr>
<th>$RHC_t$</th>
<th>DelC$_i$</th>
<th>PUC$_i$</th>
<th>$RHC_{t+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Does not quite matter what we put here since other constraints enforce that these configurations won’t happen
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!)
• E.g. let’s consider variable RHC at time t and t+1 and fill in a few rows in this table

<table>
<thead>
<tr>
<th>RHC(_t)</th>
<th>DelC(_i)</th>
<th>PUC(_i)</th>
<th>RHC(_{t+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Does not quite matter what we put here since other constraints enforce that these configurations won’t happen.
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  

*DelM* and *DelC* can occur in any sequence (or simultaneously)  
But we can enforce that they do not happen simultaneously

<table>
<thead>
<tr>
<th>DelM_i</th>
<th>DelC_i</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:  

- **Move**  
- **PUC**  
- **DelC**  
- **PUM**  
- **DelM**  
- **Action**
CSP Planning: Constraints Contd.

Other constraints we may want are **action constraints**:

- specify which actions cannot occur simultaneously
- these are sometimes called **mutual exclusion (mutex)** constraints

How can we specify that *DelM* and *DelC* cannot occur simultaneously?

<table>
<thead>
<tr>
<th></th>
<th>DelM</th>
<th>DelC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DelM</th>
<th>DelC</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DelM</th>
<th>DelC</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Handling mutex constraints in Forward Planning

If we don’t want $\text{DelM}$ and $\text{DelC}$ to occur simultaneously

How would we encode this into STRIPS for forward planning?

<table>
<thead>
<tr>
<th>$\text{DelM}_i$</th>
<th>$\text{DelC}_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Handling mutex constraints in Forward Planning

E.g., let’s say we don’t want \( DelM \) and \( DelC \) to occur simultaneously

How would we encode this into 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 mutex constraints in Forward Planning

E.g., let’s say we don’t want $DelM$ and $DelC$ to occur simultaneously

How would we encode this into STRIPS for forward planning?

<table>
<thead>
<tr>
<th>$DelM_i$</th>
<th>$DelC_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

No need to enforce this constraint 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)

<table>
<thead>
<tr>
<th>$RHC_i$</th>
<th>$RHM_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Handling state constraints in Forward Planning

<table>
<thead>
<tr>
<th>RHC&lt;sub&gt;i&lt;/sub&gt;</th>
<th>RHM&lt;sub&gt;i&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

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
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

\[ k = 0 \]
Is State$_0$ a goal?
If yes, DONE!
If no,
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

```plaintext
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
STRIPS to CSP applet

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.
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
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.
Do we have a solution to this CSP?
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](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 

Active Sales Assistant™

SHOPS

These Digital Cameras best suit your needs...

BUSINESSES

increase sales on your site with Active Sales Assistant

Top products:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Brand &amp; Model</th>
<th>Avg. Street Price</th>
<th>Optimal Score</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Toshiba PDR-M25</td>
<td>$429.99</td>
<td>1792 x 1296 pixel</td>
<td>3x</td>
</tr>
<tr>
<td>2</td>
<td>Sony DCR-SR70E</td>
<td>$499.99</td>
<td>1280 x 960 pixel</td>
<td>3x</td>
</tr>
<tr>
<td>3</td>
<td>Nikon D3100</td>
<td>$649.99</td>
<td>1440 x 1080 pixel</td>
<td>3x</td>
</tr>
</tbody>
</table>
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!