At the end of the class you should be able to:

- the model of deterministic planning
- represent a problem using both STRIPs and the feature-based representation of actions.
Planning

- Planning is deciding what to do based on an agent’s ability, its goals, and the state of the world.
- Planning is finding a sequence of actions to solve a goal.
- Initial assumptions:
  - The world is deterministic.
  - There are no exogenous events outside of the control of the robot that change the state of the world.
  - The agent knows what state it is in.
  - Time progresses discretely from one state to the next.
  - Goals are predicates of states that need to be achieved or maintained.
A deterministic **action** is a partial function from states to states.

The **preconditions** of an action specify when the action can be carried out.

The **effect** of an action specifies the resulting state.
Delivery Robot Example

Features:
- $RLoc$ – Rob’s location
- $RHC$ – Rob has coffee
- $SWC$ – Sam wants coffee
- $MW$ – Mail is waiting
- $RHM$ – Rob has mail

Actions:
- $mc$ – move clockwise
- $mcc$ – move counterclockwise
- $puc$ – pickup coffee
- $dc$ – deliver coffee
- $pum$ – pickup mail
- $dm$ – deliver mail
Explicit State-space Representation

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Resulting State</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨lab, ¬rhc, swc, ¬mw, rhm⟩</td>
<td>mc</td>
<td>⟨mr, ¬rhc, swc, ¬mw, rhm⟩</td>
</tr>
<tr>
<td>⟨lab, ¬rhc, swc, ¬mw, rhm⟩</td>
<td>mcc</td>
<td>⟨off, ¬rhc, swc, ¬mw, rhm⟩</td>
</tr>
<tr>
<td>⟨off, ¬rhc, swc, ¬mw, rhm⟩</td>
<td>dm</td>
<td>⟨off, ¬rhc, swc, ¬mw, ¬rhm⟩</td>
</tr>
<tr>
<td>⟨off, ¬rhc, swc, ¬mw, rhm⟩</td>
<td>mcc</td>
<td>⟨cs, ¬rhc, swc, ¬mw, rhm⟩</td>
</tr>
<tr>
<td>⟨off, ¬rhc, swc, ¬mw, rhm⟩</td>
<td>mc</td>
<td>⟨lab, ¬rhc, swc, ¬mw, rhm⟩</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Feature-based representation of actions

For each action:
- **precondition** is a proposition that specifies when the action can be carried out.

For each feature:
- **causal rules** that specify when the feature gets a new value and
- **frame rules** that specify when the feature keeps its value.
Example feature-based representation

Precondition of pick-up coffee (puc):

\[ RLoc = cs \land \neg rhc \]

Rules for location is cs:

\[ RLoc' = cs \leftarrow RLoc = \text{off} \land Act = mcc \]
\[ RLoc' = cs \leftarrow RLoc = mr \land Act = mc \]
\[ RLoc' = cs \leftarrow RLoc = cs \land Act \neq mcc \land Act \neq mc \]

Rules for “robot has coffee”

\[ rhc' \leftarrow rhc \land Act \neq dc \]
\[ rhc' \leftarrow Act = puc \]
Divide the features into:

- primitive features
- derived features. There are rules specifying how derived can be derived from primitive features.

For each action:

- precondition that specifies when the action can be carried out.
- effect a set of assignments of values to primitive features that are made true by this action.

STricks assumption: every primitive feature not mentioned in the effects is unaffected by the action.
Example STRIPS representation

Pick-up coffee ($puc$):
- precondition: $[cs, \neg rhc]$
- effect: $[rhc]$

Deliver coffee ($dc$):
- precondition: $[off, rhc]$
- effect: $[\neg rhc, \neg swc]$