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:**
- A **single** agent
- The world is **deterministic**.
- There are **no exogenous events** outside of the control of the agent that change the state of the world.
- The agent knows what state it is in (full **observability**).
- Time progresses **discretely** from one state to the next.
- Goals are **predicates** of states that need to be achieved or maintained (no complex goals).

Actions

- A deterministic **action** is a **partial function from states to states**.
- **Partial** function: some actions not possible in some 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 (Variables):
RLoc – Rob's location
      (4-valued: {cs,off,mr,lab})
RHC – Rob has coffee (binary)
SWC – Sam wants coffee (binary)
MW – Mail is waiting (binary)
RHM – Rob has mail (binary)

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

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.

Notation:
- Features are capitalized (e.g. $Rloc$, $RHC$)
- Values of the features are not (e.g. $Rloc = cs$, $rhc$, $¬rhc$)
- If $X$ is a feature, then $X'$ is the feature after an action is carried out

<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>

Example feature-based representation

Precondition of pick-up coffee ($puc$):

$RLoc = cs \land ¬rhc$

Rules for location is $cs$ (specifies $RLoc'$):

$RLoc' = cs \leftarrow RLoc = off \land Act = mcc$
$RLoc' = cs \leftarrow RLoc = mr \land Act = mc$
$RLoc' = cs \leftarrow RLoc = cs \land Act \neq mcc \land Act \neq mc$

Rules for “robot has coffee” (specifies $rhc'$):

(frame rule): $RHC' = true \leftarrow RCH = true \land Act \neq dc$
(causal rule): $RHC' = true \leftarrow Act = puc$

also write as:

$rhc' \leftarrow rhc \land Act \neq dc$
$rhc' \leftarrow Act = puc$

STRIPS Representation

- Previous representation was **feature-centric**: specify how each feature changes for each action that satisfies a precondition.
- **STRIPS is action-centric**: specify effects and preconditions for each action. For each action:
  - precondition that specifies when the action can be carried out.
  - effect a set of assignments of values to features that are made true by this action.

STRIPS:
STanford Research Institute Problem Solver used to program “Shakey” →

Frame assumption: all non-mentioned features stay the same.
Therefore, $V = v$ after $act$ if:

- if $V = v$ was on effect list of $act$ or
- if $V$ is not on the effect list of $act$, and $V = v$ immediately
Example STRIPS representation

Pick-up coffee (puc):
- **precondition:** [cs, ¬rhc]
- **effect:** [rhc]

Deliver coffee (dc):
- **precondition:** [off, rhc]
- **effect:** [¬rhc, ¬swc]

Planning

**Given:**
- A description of the effects and preconditions of the actions
- A description of the initial state
- A goal to achieve

**Find a sequence of actions** that is possible and will result in a state satisfying the goal.

Forward Planning

**Idea:** search in the state-space graph.
- The nodes represent the states
- The arcs correspond to the actions: The arcs from a state $s$ represent all of the actions that are legal in state $s$.
- A plan is a path from the state representing the initial state to a state that satisfies the goal.
- Can use any of the search techniques from Chap. 3
- **heuristics** important

A tutorial by Malte Helmert on Heuristics for Deterministic Planning:
https://ai.dmi.unibas.ch/misc/tutorial_aaaiai2015/

![Example state-space graph](image-url)
**Regression Planning**

Idea: search backwards from the goal description: nodes correspond to subgoals, and arcs to actions.

- Nodes are propositions: a formula made up of assignments of values to features
- Arcs correspond to actions that can achieve one of the goals
- Neighbors of a node $N$ associated with arc $A$ specify what must be true immediately before $A$ so that $N$ is true immediately after.
- The start node is the goal to be achieved.
- $\text{goal}(N)$ is true if $N$ is a proposition that is true of the initial state.

**Planning as a CSP**

- Search over planning horizons.
- For each planning horizon, create a CSP constraining possible actions and features
  - Choose a planning horizon $k$.
  - Create a variable for each state feature and each time from 0 to $k$.
  - Create a variable for each action feature for each time in the range 0 to $k - 1$.
  - Create constraints (next slide)

**Constraints**

- **state constraints**: between variables at the same time step.
- **precondition constraints**: between state variables at time $t$ and action variables at time $t$ that specify what actions are available from a state.
- **effect constraints**: between state variables at time $t$, action variables at time $t$ and state variables at time $t + 1$.
- **frame constraints**: between state variables at time $t$, action variables at time $t$ and state variables at time $t + 1$ specify that a variable does not change
- **initial state constraints** that are usually domain constraints on the initial state (at time 0).
- **goal constraints** that constrains the final state to be a state that satisfies the goals that are to be achieved.
CSP for Delivery Robot (horizon=2)

at time $i$:
- $RLoc_i$ — Rob’s location
- $RHC_i$ — Rob has coffee
- $SWC_i$ — Sam wants coffee
- $MW_i$ — Mail is waiting
- $RHM_i$ — Rob has mail

Action $i$ — Rob’s action

$SWC_0 = true$ — initial state

$RHC_0 = false$ — initial state

$SWC_0 = false$ — Goal

no solution possible:
(must use 3 actions at least and robot must start in cs)

Next:

- Supervised Learning (Poole & Mackworth (2nd ed.) Chapter 7.1-7.6)
- Uncertainty (Poole & Mackworth (2nd ed.) Chapter 8)