# Today’s Topics

1. Example STRIPS domain
2. Logical encoding of actions & change

## 2 Example STRIPS domain

### 2.1 Terminology (last time)

Planning: the problem of producing intelligent behavior. An agent autonomously selects a sequence of actions in pursuit of goals it has been given. A representation of the agent’s environment encodes the effects of the actions on the environment’s state. The goal is a desired property of the environment’s state. Robots from SRI used a propositional representation of environments for a planner, “STRIPS”. The environments that can be represented by this system are STRIPS domains.

Example:

![Figure 1: Robot in room 0, box in room 1](image)

- **Attributes**
  - Robot’s location: \( at_{r,i} \equiv "\text{Robot is in room } i" \)
  - Box: \( at_{b,i} \equiv "\text{Box is in room } i" \)
  - Holding the box: \( holds_{r,b} \)

- **Actions**
  - "move to room \( i \) from \( i' \)”: set the current \( at_{r,i'} \) to 0, set \( at_{r,i} = 1 \)
  - "pick up in room \( i \)”: set \( holds_{r,b} = 1 \)
  - "put down”: set \( holds_{r,b} = 0 \)
• **Preconditions**
  must be in room $i'$: $at_{r, i'} = 1$
  robot must be in the same room as the box: $at_{r, i} = 1, at_{b, i} = 1$
  robot must be holding the box: $holds_{r, b} = 1$

Generally, STRIPS actions are given by a list of literals indicating which attributes are set to 
ture/false by the action.
Actions may have **preconditions**, a conjunction (list of literals) that must be satisfied for the agent
to take that action.
Actions may have **conditional effects**, effects that only occur if some given conjunction/list of
literals holds.

• **Conditional Effect**
  If the robot holds the box, the box also moves from $i'$ to $i$.

• **Precondition Effect**
  If $holds_{r, b} = 1$ is satisfied and the robot takes action "move to room $i$ from $i'$", then we set $at_{b, i'} = 0, at_{b, i} = 1$.

• **Goal**
  Ex. "Put box in room 2 and leave room 2": $at_{b, 2} \wedge \neg at_{r, 2}$
  STRIPS goals are given by conjunctions of the environment state attributes.

• **Plan**
  STRIPS plans are sequences of actions s.t. the preconditions of each action is satisfied.

• **Planning Problem**
  Planning problem given by:
  1) A STRIPS domain, describing the states, actions, and their effects
  2) A complete description of the initial settings of the state attributes
  3) A goal

• **Solution**
  A plan transforming the initial state attributes into one that satisfies the goal.
  Our example (Figure 1):
  In our robot state we start from the initial conditioned state described above. Suppose the goal
  we want to achieve is $at_{b, 2} \wedge at_{r, 2}$. Then the following is a solution:

  1. Move to Room 1 ($move_{0,1}$)
  2. Pick up the box ($pickup_1$)
  3. Move to Room 0 ($move_{1,0}$)
  4. Move to Room 2 ($move_{0,2}$)
  5. Put down the box ($putdown$)
  6. Move to Room 0 ($move_{2,0}$)

  The following table describe the attributes throughout this plan:
3 Logical encoding of actions & change

Standard approach to planning, represent the problem as a CNF, pass to a SAT-solver satisfying assignment $\leftrightarrow$ encoding of a plan solving the problem.

3.1 Small issue

- We don’t want to use the propositional state variables directly as variables in CNF, because they change. Attributes that change over time are called fluents.
- In general, for each time index $t = 0, 1, ...$, we include a copy of each fluent as a propositional variable.
- Another problem how many copies should we include? We can count the number of distinct possible states if $n$ attributes, there are less than or equal to $2^n$ states, we don’t need to consider plans of length larger than $2^n$, since then we must have repeated states, we can cut this "loop" out of the plan. In general, we can always assume some upper bound on the "time horizon", $T$. We’ll only generate copies out to the time horizon $T$ —- in general, much, much less than $2^n$, encoding has $n \cdot T$ variables —- poly. in the output length, description length of environment.

The following table describes the attributes throughout this plan:

<table>
<thead>
<tr>
<th>Step</th>
<th>$h_{olds_{r,b}}$</th>
<th>$at_{r,0}$</th>
<th>$at_{r,1}$</th>
<th>$at_{r,2}$</th>
<th>$at_{b,0}$</th>
<th>$at_{b,1}$</th>
<th>$at_{b,2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: We have $n$ fluents, the overall number of variables is $n \cdot T$.

We’ll assume we have a wait action that leaves the environment unchanged so that we don’t need to guess $T = \text{plan length}$ (can do binary search on $T$ to find shortest plan).

3.2 Pitfalls

How do we encode actions with preconditions?

- effects $\alpha_t$

  Actions $\alpha$ has effects $e_1, ..., e_k$,

  $\begin{align*}
  \alpha_t &\Rightarrow e_{1,t+1} \\
  \vdots \\
  \alpha_t &\Rightarrow e_{k,t+1}
  \end{align*}$


• Preconditions $\alpha_t$
  Preconditions $P_1, \ldots, P_r,$
  $\alpha_t \Rightarrow P_{1,t}$
  $\vdots$
  $\alpha_t \Rightarrow P_{r,t}$

Conditional effect $e'_1, \ldots, e'_k,$
  $[P_{1,t} \land \ldots \land P_{r,t} \land \alpha_t] \Rightarrow e'_1$
  $\vdots$
  $[P_{1,t} \land \ldots \land P_{r,t} \land \alpha_t] \Rightarrow e'_k$

What is missing?

1. Describe what changes, not what doesn’t change ("frame problem").
2. Does not enforce one action at a time (easy to address).

• "Unique" $\alpha_t$
  We add,
  $$\bigvee_{\alpha \in \text{Actions}} \alpha_t, \quad t = 0, \ldots, T - 1$$

it must take an action (maybe wait).

Mutual exclusion between actions,
  $$\neg[\alpha_t \land \alpha'_t] \equiv \neg\alpha_t \lor \neg\alpha'_t, \quad \forall \alpha, \alpha', \quad t = 0, \ldots, T - 1$$

taking action $\alpha@t$ prevents $\alpha'.$

For 1, it is easy with negation-as-failure, we can add,
  $$[\ell_t \land \text{not}(-\ell_{t+1})] \Rightarrow \ell_{t+1}$$

• Frame axioms $\alpha, \ell$
  $$[-\ell_t \land \ell_{t+1}] \Rightarrow \bigvee_{\alpha: \text{laneffect}} \alpha_t$$

for $t = 0, \ldots, T - 1$, "explanatory frame axiom".

**Proposition** For any time horizon $T,$ the formula,
  $$\bigwedge_{t=0}^{T-1} ([\text{unique } \alpha_t] \land [\text{preconditions}]_{\alpha_t} \land [\text{effects}]_{\alpha_t} \land [\text{frame axioms}]_{\alpha_t, \ell_t})$$
  $$\land (\ell_{1,0} \land \ldots \land \ell_{n,0})$$

where $(\ell_{1,0} \land \ldots \land \ell_{n,0})$ is conjunction encoding initial state, has for each $T$-step plan $\alpha_0, \ldots, \alpha_{T-1},$ a unique satisfying assignment in which, for each fluent $i$ and step $t,$ $x_{i,t} = 1$ iff $x_{i,t}$ is satisfied on the $t^{th}$ step of the plan from the initial state.

So, by taking this formula $\varphi$ and a goal can conjunction $G(x_{1,T}, \ldots, x_{n,T})$ and passing $\varphi \land G$ to a SAT-solver, satisfying assignment $\leftrightarrow$ plan achieving $G.$
**Proof sketch** By induction on $t$. There is a unique setting of the fluent and action variables for $\alpha_0, \ldots, \alpha_{t-1}$ from the initial state. Unique $\alpha_t$ formula, exactly one setting of $\alpha_{1,t}, \ldots, \alpha_{m,t}$ making $\alpha_t$ (action for step $t$) true.

Precondition constraints: we can only set $\alpha_{i,t} = 1$ if preconditions satisfied in this unique setting of $x_{1,t}, \ldots, x_{n,t}$.

Effect and frame axioms: uniquely determine $x_{1,t+1}, \ldots, x_{n,t+1}$ given $\alpha_t, x_{1,t}, \ldots, x_{n,t}$. ■

Note: easy to construct STRIPS domains encoding satisfiability e.g., we have actions that set variables $x_1, \ldots, x_n$ irrevocably, given a literal in a clause is satisfied, we have a conditional action letting us set the clause’s attribute to 1. Goal makes clause variables all 1.