Lecture 3

- Normal form for operators
- Definition of the deterministic planning problem
- Progression and regression (forward and backward search)
- Planning by heuristic search algorithms
Equivalences on effects

\[ c \triangleright (e_1 \land \cdots \land e_n) \equiv (c \triangleright e_1) \land \cdots \land (c \triangleright e_n) \quad (1) \]

\[ c \triangleright (c' \triangleright e) \equiv (c \land c') \triangleright e \quad (2) \]

\[ (c_1 \triangleright e) \land (c_2 \triangleright e) \equiv (c_1 \lor c_2) \triangleright e \quad (3) \]

\[ e \land (c \triangleright e) \equiv e \quad (4) \]

\[ e \equiv \top \triangleright e \quad (5) \]

\[ e \equiv \top \land e \quad (6) \]

\[ e \land e' \equiv e' \land e \quad (7) \]

\[ (e_1 \land e_2) \land e_3 \equiv e_1 \land (e_2 \land e_3) \quad (8) \]
Normal form for operators

DEFINITION: An operator \( \langle c, e \rangle \) is in normal form if for all occurrences of \( c' \triangleright e' \) in \( e \) the effect \( e' \) is either \( p \) or \( \neg p \) for some \( p \in P \), and \( e \) contains at most one of \( l \), \( c \triangleright l \) and \( c' \triangleright l \) for any \( l \), \( c \) and \( c' \).

THEOREM: For every operator there is an equivalent one in normal form.

PROOF: We can transform any operator into normal form by using equivalences 1, 2, 3 and 5.
Normal form for operators: example

\[(A \triangleright (B \land \\
(C \triangleright (\neg D \land E)))\)\land \\
(\neg B \triangleright E)\]

transformed to normal form is

\[(A \triangleright B)\land \\
((A \land C) \triangleright \neg D)\land \\
((\neg B \lor (A \land C)) \triangleright E)\]
Transition systems

- Model the dynamics of the world/system/application.

- Are formalized as \( \langle S, \{a_1, \ldots, a_n\} \rangle \) where
  - \( S \) is a finite set of states,
  - every \( a_i \subseteq S \times S \) is a binary relation on \( S \).

- First we restrict to \( a_i \) that are (partial) functions from \( S \) to \( S \): for every \( s \in S \), \( (s, s') \in a_i \) for at most one \( s' \in S \).
Sum matrix  $M_R + M_G + R_B$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

We use addition $0 + 0 = 0$ and $b + b' = 1$ if $b = 1$ or $b' = 1$. 
Sequential composition as matrix multiplication

\[
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\times
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 \\
\end{pmatrix}
= 
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\]

E is reachable from B in two steps, because F is reachable from B in one step and also E is reachable from F in one step.
Reachability

Let $M$ be the matrix that is the (Boolean) sum of the matrices of the individual actions. Define

$$
M_0 = \begin{bmatrix} 1 \end{bmatrix} \quad M_i = M_{i-1} + MM_{i-1} \text{ for all } i \geq 1
$$

where $n$ is the number of states, $I_{n \times n}$ is the unit matrix.

For some $i \in \{1, \ldots, n\}$ and all $j \geq i$, $M_i = M_j$.

Matrix $M_i = M^0 \cup M^1 \cup \cdots \cup M^i$ represents reachability by $i$ actions or less.
Reachability: example, $M_R$

$$
\begin{array}{ccccccc}
 & A & B & C & D & E & F \\
A & 0 & 1 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 0 & 0 & 0 & 1 \\
C & 0 & 0 & 1 & 0 & 0 & 0 \\
D & 0 & 0 & 1 & 0 & 0 & 0 \\
E & 0 & 1 & 0 & 0 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
$$
Reachability: example, $M_R + M_R^2$
Reachability: example, $M_R + M_R^2 + M_R^3$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Reachability: example**, 

\[ M_R + M_R^2 + M_R^3 + I_{6\times6} \]

```
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Reachability: row vectors are sets of states

Row vectors $S$ represent sets.

$SM$ is the set of states reachable from $S$ by $M$.

$$
\begin{pmatrix}
1 \\
0 \\
1 \\
0 \\
0
\end{pmatrix}
\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1
\end{pmatrix} =
\begin{pmatrix}
1 \\
1 \\
0
\end{pmatrix}
\begin{pmatrix}
1 \\
1 \\
1 \\
1
\end{pmatrix}
$$
Definition of planning with 1 initial state, deterministic actions

Let the 4-tuple $\langle P, I, O, G \rangle$ consist of a set $P$ of state variables, an initial state $I$ (a valuation of $P$), a set $O$ of operators, and a propositional formula $G$ over $P$.

The problem is to find a sequence $o_1, \ldots, o_n$ of operators so that $\text{app}_{o_n}(\text{app}_{o_{n-1}}(\cdots \text{app}_{o_1}(I) \cdots)) \models G$, that is, when applying the operators $o_1, \ldots, o_n$ in this order starting in the initial state, one of the goal states described by $G$ is reached.
A planning algorithm (not a very good one)

1. $n = 2^{|P|}$ is the number of states.

2. Generate the $n \times n$ matrices $M_o$ for every $o \in O$.

3. Compute the $n \times n$ matrix $M = \sum_{o \in O} M_o$.

4. Compute the $i$-step matrices $M_i$.

5. Represent the initial state as the row vector $I_{1 \times n}$.

6. Represent the set of goal states as the row vector $G_{1 \times n}$.
A planning algorithm, cont’d

Let $i$ be the least number such that $I_{1 \times n}^T M_i \cdot G_{1 \times n}^T \cdot G_{1 \times n} > 0$. (If there is no such $i$, there is no plan.)

Read the plan backwards from the matrices:

1. Start from a goal state $s$ in $I_{1 \times n}^T M_i$.
2. Output $o \in O$ such that $s$ is in $I_{1 \times n}^T M_i \cdot M_o$.
3. Set $s$ to a state from which $s$ can be reached with $o$ and that is in $I_{1 \times n}^T M_{i-1}$.
4. $i := i - 1$;
5. Continue from 2.
A planning algorithm: example

\[
\begin{array}{cccccc}
& A & B & C & D & E & F \\
A & 0 & 1 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 0 & 0 & 0 & 1 \\
C & 0 & 1 & 1 & 0 & 0 & 1 \\
D & 1 & 0 & 1 & 0 & 1 & 0 \\
E & 0 & 1 & 0 & 1 & 0 & 0 \\
F & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
\]
### A planning algorithm: example, cont’d

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$M_0 = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$M_1 = \begin{bmatrix}1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

$M_2 = \begin{bmatrix}1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$M_3 = \begin{bmatrix}1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

$M_4 = \begin{bmatrix}1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$M_5 = \begin{bmatrix}1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

Shortest plan for reaching C from A: A red B red F red E green D red C.
Progression

- Progression is computing the successor state $a_p(o)(s)$ of $s$ with respect to $o$.

- Used in *forward search* in a transition system: from the initial state toward the goal states.

- Efficient to implement.

- Only for deterministic planning: *nondeterministic operators* may produce a *set of states* from one state.
Search algorithms 1: Search with progression

depth-first search, breadth-first search, iterative deepening

More sophisticated search algorithms follow in a moment...
Regression

- The formula $\text{regr}_o(\phi)$ represents the set of states from which a state in $\phi$ is reached by operator $o$.

- Used in *backward search* in a transition system: from the goal states toward the initial states.

- Regression is powerful because it allows handling huge sets of states (progression: only one state at a time.)

- Handling formulae is more complicated than handling states: many questions about regression are NP-hard.
Regression: definition

Let $\phi$ be a propositional formula describing a set of states.

Let $\langle z, e \rangle$ be an operator in normal form.

First, unconditional effects $l$ are replaced by $\top \triangleright l$.

Let $c_1 \triangleright p_1, \overline{c}_1 \triangleright \neg p_1, \ldots, c_n \triangleright p_n, \overline{c}_n \triangleright \neg p_n$ be the conjuncts of $e$.

For literals $p_i$ and $\neg p_i$ that do not occur in $e$ as an effect, we define $c_i = \bot$ and $\overline{c}_i = \bot$, respectively.
Regression: definition, cont’d

Regression of $\phi$ w.r.t. $o = \langle z, e \rangle$ is $\text{regr}_o(\phi) = \phi_o \land z \land f$ where

- $\phi_o$ is like $\phi$ but every $p_i$ replaced by $(p_i \land \neg c_i) \lor c_i$, and

- $f$ is conjunction of formulae $\neg (c_i \land \bar{c}_i)$ for all $p_i$ that occur in $e$ both as $p_i$ and as $\neg p_i$.

$(p_i \land \neg \bar{c}_i) \lor c_i$ means: either $p_i$ was true before and did not become false (because of $\bar{c}_i \Rightarrow \neg p_i$) or $p_i$ became true (because of $c_i \Rightarrow p_i$).

$\neg (c_i \land \bar{c}_i)$ means: for $(c_i \Rightarrow p) \land (\bar{c}_i \Rightarrow \neg p)$ one of $c_i$ and $\bar{c}_i$ is false.
Regression: correctness

LEMMA: Let $\phi$ be a formula over $P$. Let $s$ and $s'$ be states over $P$ so that $s' = \text{app}_o(s)$. Then $s \models \text{regr}_o(\phi)$ if and only if $s' \models \phi$.

PROOF: This is a consequence of the fact that for every $p \in P$ the formula $(p \land \neg c) \lor c$ has in the predecessor state the same truth-value as $p$ in the successor state. Because $\text{regr}_o(\phi)$ is obtained from $\phi$ by replacing the latter formulae by the former, the truth-value of $\text{regr}_o(\phi)$ in $s$ equals the truth-value of $\phi$ in $s'$. This can be shown by a structural induction over $\phi$. Q.E.D.
Regression: examples

\[ \text{regr}_{\langle a,b \rangle}(b) = (((b \land \neg \bot) \lor \top) \land a) \equiv a \]

\[ \text{regr}_{\langle a,b \rangle}(b \land c \land d) = (((b \land \neg \bot) \lor \top) \land ((c \land \neg \bot) \lor \bot) \land ((d \land \neg \bot) \lor \bot) \land a) \equiv c \land d \land a \]

\[ \text{regr}_{\langle a,c \bowtie b \rangle}(b) = (((b \land \neg \bot) \lor c) \land a) \equiv (b \lor c) \land a \]

\[ \text{regr}_{\langle a,(c \bowtie b) \land (b \bowtie \neg b) \rangle}(b) = (((b \land \neg b) \lor c) \land a \land \neg (c \land b)) \equiv c \land a \land \neg b \]

\[ \text{regr}_{\langle a,(c \bowtie b) \land (d \bowtie \neg b) \rangle}(b) = (((b \land \neg d) \lor c) \land a \land \neg (c \land d)) \equiv (b \lor c) \land (\neg d \lor c) \land a \land (\neg c \lor d) \]
Regression: problems

1. \( \text{regr}_{\langle a, \neg p \rangle}(p) = a \land \bot \equiv \bot \): the new set of states is empty.
   Testing that a formula \( \text{regr}_o(\phi) \) does not represent the empty set (= search is in a blind alley) is NP-hard.

2. \( \text{regr}_{\langle b, c \rangle}(a) = a \land b \): the new set of states is properly smaller.
   Testing that a regression step does not make the set of states smaller (= more difficult to reach) is NP-hard.

These tests would be useful in pruning the search space.
Search algorithms: Planning by heuristic search algorithms

Heuristic search algorithms use a heuristic (= an estimate on the value of the current search state) to guide search in the search space.

Search states with progression: the initial state $s_I$, states

Search states with regression: the goal formula $\phi_G$, formulas
Search algorithms: Heuristic evaluation of states

In the next lecture we will describe how the distance (= number of operators) between a state and a set of states can be estimated. Both backward and forward search can use these estimates.

Forward: Distance between the current state and the goals.
Backward: Distance between the initial state and the current goal.
Search algorithms: systematic, local

Systematic algorithms:

- Keep track of all the states already visited.
- Memory consumption may be high.
- Always find a plan if one exists.
- depth-first, breadth-first, best-first, A*, IDA*, WA*, ...
Search algorithms: systematic, local

Local search algorithms:

- Keep track of only one search state at a time.
- Succeed with a high probability (given enough time).
- Cannot determine that no plans exist.
- Local minima may be a problem.
- hill-climbing, simulated annealing, tabu search, ...