1. **Memoryless plans** map a state/an observation to an operator. 
   We use this definition of plans for **fully observable** problems only.

2. **Conditional plans** generalize memoryless plans. 
   They are needed for problems without full observability.
   - The *state of the execution* of a conditional plan depends on *observations* on earlier execution steps.
   - The state of the execution = a primitive form of memory.
   - The operator to be executed depends on the state of the execution.
Memoryless plans

Example
Memoryless plans

Example
Memoryless plans
Definition

Let $S$ be the set of all states.
A memoryless plan is a partial function $\pi : S \rightarrow O$.

Execution of a memoryless plan

1. Determine the current state $s$ (full observability!!!).
2. If $\pi(s)$ is not defined then terminate execution.
   If the objective is to reach a goal state, then $\pi(s)$ is not defined if $s$ is a goal state so that the execution terminates.
3. Execute action $\pi(s)$.
The **image** of a set $T$ of states with respect to an operator $o$ is the set of those states that can be reached by executing $o$ in a state in $T$. 
Images
Formal definition

Definition (Image of a state)

\[ \text{img}_o(s) = \{ s' \in S | sos' \} \]

Definition (Image of a set of states)

\[ \text{img}_o(T) = \bigcup_{s \in T} \text{img}_o(s) \]
Weak preimage

The **preimage** of a set $T$ of states with respect to an operator $o$ is the set of those states from which a state in $T$ can be reached by executing $o$. 

$$\text{preimg}_o(T)$$
Definition (Weak preimage of a state)

\[ \text{preimg}_o(s') = \{ s \in S \mid sos' \} \]

Definition (Weak preimage of a set of states)

\[ \text{preimg}_o(T) = \bigcup_{s \in T} \text{preimg}_o(s). \]
The strong preimage of a set $T$ of states with respect to an operator $o$ is the set of those states from which a state in $T$ is always reached when executing $o$. 

$$spreimg_o(T)$$
Strong preimages
Formal definition

Definition (Strong preimage of a set of states)

\[ \text{spreimg}_o(T) = \{ s \in S | s' \in T, sos', \text{img}_o(s) \subseteq T \} \]
Algorithms for fully observable problems

1 Heuristic search (forward)
Nondeterministic planning can be viewed as AND-OR search.

   OR nodes: Choice between operators
   AND nodes: Nondeterministically reached state

Heuristic AND-OR search algorithms: AO*, ...

2 Dynamic programming (backward)
Idea Compute operator/distance/value for a state based on the operators/distances/values of its all successor states.

   1 0 actions needed for goal states.
   2 If states with \( i \) actions to goals are known, states with \( \leq i + 1 \) actions to goals can be easily identified.

Automatic reuse of already found plan suffixes.
AND-OR search
AND-OR search
Dynamic programming

Planning by dynamic programming

If for all successors of state $s$ with respect to operator $o$ a plan exists, assign operator $o$ to $s$.

Base case $i = 0$: In goal states there is nothing to do.

Inductive case $i \geq 1$: If there is $o \in O$ such that for all $s' \in img_o(s)$ $s'$ is a goal state or $\pi(s')$ was assigned on iteration $i - 1$, then assign $\pi(s) = o$.

Connection to distances

If $s$ is assigned a value on iteration $i \geq 1$, then the backward distance of $s$ is $i$.

The dynamic programming algorithm essentially computes the backward distances of states.
Backward distances

Example

![Graph showing backward distances to G]

- Distance to $G$: $\infty$
- Levels: 3, 2, 1, 0

- Blue and red arrows indicating transitions and distances.
Backward distances
Definition of distance sets

Definition

Let $G$ be a set of states and $O$ a set of operators. Define the backward distance sets $D_{bwd}^i$ for $G, O$ that consist of those states for which there is a guarantee of reaching a state in $G$ with at most $i$ operator applications.

\[
\begin{align*}
D_{bwd}^0 &= G \\
D_{bwd}^i &= D_{bwd}^{i-1} \cup \bigcup_{o \in O} \text{spreimg}_o(D_{bwd}^{i-1}) \text{ for all } i \geq 1
\end{align*}
\]
Backward distances

Definition

Let $\mathcal{G}$ be a set of states and $\mathcal{O}$ a set of operators, and let $\mathcal{D}_0^{bwd}, \mathcal{D}_1^{bwd}, \ldots$ be the backward distance sets for $\mathcal{G}$ and $\mathcal{O}$. Then the backward distance from a state $s$ to $\mathcal{G}$ is

$$\delta_{\mathcal{G}}^{bwd}(s) = \begin{cases} 0 & \text{if } s \in \mathcal{G} \\ i & \text{if } s \in \mathcal{D}_i^{bwd} \setminus \mathcal{D}_{i-1}^{bwd} \end{cases}$$

If $s \not\in \mathcal{D}_i^{bwd}$ for all $i \geq 0$ then $\delta_{\mathcal{G}}^{bwd}(s) = \infty$. 
Construction of a plan based on distances

Extraction of a plan from distance sets

1. Let $S' \subseteq S$ be those states having a finite backward distance.

2. Let $s$ be a state with distance $i = \delta^\text{bwd}_G(s) \geq 1$.

3. Assign to $\pi(s)$ any operator $o \in O$ such that $\text{img}_o(s) \subseteq D^\text{bwd}_{i-1}$. Hence $o$ decreases the backward distance by at least one.

The plan $\pi$ solves the planning problem for $\langle S, I, O, G, P \rangle$ iff $I \subseteq S'$.
Making the algorithm a logic-based algorithm

- An algorithm that represents the states explicitly is feasible for transition systems with at most $10^6$ or $10^7$ states.
- For planning with bigger transition systems structural properties of the transition system have to be taken advantage of.
- Representing state sets as propositional formulae often allow taking advantage of the structural properties: a formula that represents a set of states or a transition relation that has certain regularities may be very small in comparison to the set or relation.
Making the algorithm a logic-based algorithm

- We use a formula $\phi$ as a **data structure** for representing the set $\{s \in S | s \models \phi\}$.
- We show that regression $\text{regr}_{nd}^{o}(\phi)$ for nondeterministic operators is one way of computing strong preimages.
- We present general techniques for computing images, preimages and strong preimages of sets of states represented as formulae.
- Many of the algorithms presented later in the lecture can be **lifted** to use a logic-based representation, thereby expanding their range of applicability to much bigger transition systems.
We can easily generalize our regression operation for deterministic operators to \textit{regression for nondeterministic operators} of a restricted syntactic form.

**Definition (Regression for nondeterministic operators)**

Let $\phi$ be a propositional formula and $o = \langle c, e_1 | \cdots | e_n \rangle$ an operator where $e_1, \ldots, e_n$ are deterministic. Define

$$\text{regr}_{o}^{nd}(\phi) = \text{regr}_{\langle c, e_1 \rangle}(\phi) \land \cdots \land \text{regr}_{\langle c, e_n \rangle}(\phi).$$
Regression for nondeterministic operators

Illustration

\[
\text{regr}_{c,(e_1|e_2)}(\phi) = \text{regr}_{c,e_1}(\phi) \land \text{regr}_{c,e_2}(\phi)
\]
Theorem

Let $\phi$ be a formula over $A$, $o$ an operator over $A$, and $S$ the set of all states over $A$. Then

$$\{ s \in S | s \models \text{regr}^{nd}_o(\phi) \} = \text{spreimg}_o(\{ s \in S | s \models \phi \}).$$

Proof.

Let $o = \langle c, (e_1| \cdots |e_n) \rangle$.

$$\{ s \in S | s \models \text{regr}^{nd}_o(\phi) \}$$

$$= \{ s \in S | s \models \text{regr}_{\langle c, e_1 \rangle}(\phi) \land \cdots \land \text{regr}_{\langle c, e_n \rangle}(\phi) \}$$

$$= \{ s \in S | s \models \text{regr}_{\langle c, e_1 \rangle}(\phi), \ldots, s \models \text{regr}_{\langle c, e_n \rangle}(\phi) \}$$

$$= \{ s \in S | \text{app}_{\langle c, e_1 \rangle}(s) \models \phi, \ldots, \text{app}_{\langle c, e_n \rangle}(s) \models \phi \}$$

$$= \{ s \in S | s' \models \phi \text{ for all } s' \in \text{img}_o(s), \text{ there is } s' \models \phi \text{ with } sos' \}$$

$$= \text{spreimg}_o(\{ s \in S | s \models \phi \}).$$

3rd = is by properties of deterministic regression.

4th = is by $\text{img}_o(s) = \{ \text{app}_{\langle c, e_1 \rangle}(s), \ldots, \text{app}_{\langle c, e_n \rangle}(s) \}$. 
Regression for nondeterministic operators

Example

Let \( o = \langle d, (b|\neg c) \rangle \). Then

\[
\text{regr}^n d_o (b \leftrightarrow c) = \text{regr}_{d,b} (b \leftrightarrow c) \land \text{regr}_{d,\neg c} (b \leftrightarrow c) \\
= (d \land (\top \leftrightarrow c)) \land (d \land (b \leftrightarrow \bot)) \\
\equiv d \land c \land \neg b.
\]
By using regression we can compute formulas that represent backward distance sets.

**Definition**

Let $G$ be a formula and $O$ a set of operators. The backward distance sets $D_{i}^{bwd}$ for $G, O$ are represented by the following formulae.

$$D_{0}^{bwd} = G$$
$$D_{i}^{bwd} = D_{i-1}^{bwd} \lor \bigvee_{o \in O} \text{regr}_{o}^{nd}(D_{i-1}^{bwd}) \text{ for all } i \geq 1$$
Definition

Let $G$ be a formula and $O$ a set of operators, and let $D_0^{bwd}, D_1^{bwd}, \ldots$ be the formulae representing the backward distance sets for $G$ and $O$. Then the backward distance from a state $s$ to $G$ is

$$\delta_G^{bwd}(s) = \begin{cases} 0 & \text{if } s \models G \\ i & \text{if } s \models D_i^{bwd} \land \neg D_{i-1}^{bwd} \end{cases}$$

If $s \not\models D_i^{bwd}$ for all $i \geq 0$ then $\delta_G^{bwd}(s) = \infty$. 
The definition of regression covers only a subclass of nondeterministic operators.

How to define strong preimages for all operators, and images and preimages?

Now we apply a general idea:

1. View operators/actions as binary relations.
2. Represent these binary relations as formulae.
3. Define relational operations for relations represented as formulae.
General images and preimages with formulas

**Definition**

Define the set of state variables possibly changed by $e$ as

$$changes(a) = \{a\}$$

$$changes(\neg a) = \{a\}$$

$$changes(c \triangleright e) = changes(e)$$

$$changes(e_1 \land \cdots \land e_n) = changes(e_1) \cup \cdots \cup changes(e_n)$$

$$changes(e_1| \cdots |e_n) = changes(e_1) \cup \cdots \cup changes(e_n)$$

**Assumption**

Let $e_1 \land \cdots \land e_n$ occur in the effect of an operator. If $e_1, \ldots, e_n$ are not all deterministic then $a$ and $\neg a$ may occur as an atomic effect in at most one of $e_1, \ldots, e_n$.

This assumption rules out effects like $(a|b) \land (\neg a|c)$ that may make $a$ simultaneously true and false.
General images and preimages with formulas

In nondeterministic choices $e_1 \mid \cdots \mid e_n$ the formula for each $e_i$ has to express the changes for exactly the same set $B$ of state variables.

**Definition**

\[
\tau^{nd}_B(e) = \tau_B(e) \quad \text{when } e \text{ is deterministic}
\]

\[
\tau^{nd}_B(e_1 \mid \cdots \mid e_n) = \tau^{nd}_B(e_1) \lor \cdots \lor \tau^{nd}_B(e_n)
\]

\[
\tau^{nd}_B(e_1 \land \cdots \land e_n) = \tau^{nd}_{B \setminus (B_2 \cup \cdots \cup B_n)}(e_1) \land \tau^{nd}_{B_2}(e_2) \land \cdots \land \tau^{nd}_{B_n}(e_n)
\]

where $B_i = \text{changes}(e_i)$ for $i \in \{2, \ldots, n\}$
General images and preimages with formulas

Example

We translate the effect

$$e = (a \uparrow (d \triangleright a)) \land (c \downarrow d)$$

into a propositional formula. The set of state variables is

$$A = \{a, b, c, d\}.$$ 

$$\tau^{\text{nd}}_{\{a, b, c, d\}}(e) = \tau^{\text{nd}}_{\{a, b\}}(a \uparrow (d \triangleright a)) \land \tau^{\text{nd}}_{\{c, d\}}(c \downarrow d)$$

$$= (\tau^{\text{nd}}_{\{a, b\}}(a) \lor \tau^{\text{nd}}_{\{a, b\}}(d \triangleright a)) \land (\tau^{\text{nd}}_{\{c, d\}}(c) \lor \tau^{\text{nd}}_{\{c, d\}}(d))$$

$$= (((a' \land (b \leftrightarrow b'))) \lor (((a \lor d) \leftrightarrow a') \land (b \leftrightarrow b'))) \land$$

$$(((c' \land (d \leftrightarrow d'))) \lor (((c \leftrightarrow c') \land d')))$$
Definition

Let $A$ be a set of state variables. Let $o = \langle c, e \rangle$ be an operator over $A$ in normal form. Define $\tau_A^{nd}(o) = c \land \tau_A^{nd}(e)$.

Lemma

Let $o$ be an operator. Then

$$\{v | v \text{ is a valuation of } A \cup A', v \models \tau_A^{nd}(o)\}$$

$$= \{s \cup s'[A'/A] | s, s' \in S, s' \in \text{img}_o(s)\}.$$
The most important operations performed on transition relations represented as propositional formulae are based on existential abstraction and universal abstraction.

**Definition**

**Existential abstraction** of a formula $\phi$ with respect to $a \in A$:

$$\exists a.\phi = \phi[\top /a] \lor \phi[\bot /a].$$

Universal abstraction is defined analogously by using conjunction instead of disjunction.

**Definition**

**Universal abstraction** of a formula $\phi$ with respect to $a \in A$:

$$\forall a.\phi = \phi[\top /a] \land \phi[\bot /a].$$
Example

\[ \exists b. ((a \rightarrow b) \land (b \rightarrow c)) \]
\[ = ((a \rightarrow \top) \land (\top \rightarrow c)) \lor ((a \rightarrow \bot) \land (\bot \rightarrow c)) \]
\[ \equiv c \lor \neg a \]
\[ \equiv a \rightarrow c \]

\[ \exists ab. (a \lor b) = \exists b. (\top \lor b) \lor (\bot \lor b) \]
\[ = ((\top \lor \top) \lor (\bot \lor \top)) \lor ((\top \lor \bot) \lor (\bot \lor \bot)) \]
\[ = (\top \lor \top) \lor (\top \lor \bot) = \top \]

Example

\[ \exists \text{-abstraction is also known as forgetting:} \]
\[ \exists \text{mon}\exists \text{tue}((\text{mon} \lor \text{tue}) \land (\text{mon} \rightarrow \text{work}) \land (\text{tue} \rightarrow \text{work})) \]
\[ \equiv \exists \text{tue}((\text{work} \land (\text{tue} \rightarrow \text{work})) \lor (\text{tue} \land (\text{tue} \rightarrow \text{work}))) \equiv \text{work} \]
∀a and ∃a correspond to combining pairs of lines with the same valuation for variables other than a.

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a ∨ (b ∧ c)</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```
∀a b | ∃c.(a ∨ (b ∧ c)) ≡ a ∨ b ∀c.(a ∨ (b ∧ c)) ≡ a
```

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a ∨ (b ∧ c)</th>
<th>∃c.(a ∨ (b ∧ c))</th>
<th>∀c.(a ∨ (b ∧ c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
∀a b | ∃c.(a ∨ (b ∧ c)) ≡ a ∨ b ∀c.(a ∨ (b ∧ c)) ≡ a
```

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a ∨ (b ∧ c)</th>
<th>∃c.(a ∨ (b ∧ c))</th>
<th>∀c.(a ∨ (b ∧ c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Properties of abstraction operations

**Definition**

Existential and universal abstraction of $\phi$ with respect to a set of atomic propositions $B = \{b_1, \ldots, b_n\}$ are

$$\exists B. \phi = \exists b_1.(\exists b_2.(\ldots \exists b_n. \phi \ldots))$$

$$\forall B. \phi = \forall b_1.(\forall b_2.(\ldots \forall b_n. \phi \ldots)).$$
Let \( \phi \) be a formula over \( A \). Then \( \exists A.\phi \) and \( \forall A.\phi \) are formulae that consist of the constants \( \top \) and \( \bot \) and the logical connectives only.

The truth-values of these formulae are independent of the valuation of \( A \), that is, their values are the same for all valuations.

1. \( \exists A.\phi \equiv \top \) if and only if \( \phi \) is satisfiable.
2. \( \forall A.\phi \equiv \top \) if and only if \( \phi \) is valid.
Lemma

If φ is a formula over $A \cup A'$ and $v$ a valuation of $A$ then

1. $v \models \exists A'. \phi$ iff $v \cup v' \models \phi$ for some valuation $v'$ of $A'$.
2. $v \models \forall A'. \phi$ iff $v \cup v' \models \phi$ for all valuations $v'$ of $A'$. 
Abstracting one variable takes polynomial time in the size of the formula.

Abstracting one variable may double the formula size.

Abstracting \( n \) variables may increase size by factor \( 2^n \).

For making abstraction practical the formulae must be simplified, for example with equivalences like

\[
\top \land \phi \equiv \phi, \quad \bot \land \phi \equiv \bot, \quad \top \lor \phi \equiv \top, \quad \bot \lor \phi \equiv \phi, \quad \lnot \bot \equiv \top, \quad \text{and} \quad \lnot \top \equiv \bot.
\]
Images by $\exists$-abstraction

Let

- $A = \{a_1, \ldots, a_n\}$,
- $A' = \{a'_1, \ldots, a'_n\}$,
- $\phi_1$ be a formula on $A$ representing a row vector $V_{1 \times 2^n}$ (equivalently, a set of valuations of $A$), and
- $\phi_2$ a formula on $A \cup A'$ representing a matrix $M_{2^n \times 2^n}$ (equivalently, a binary relation on valuations of $A$).

The product matrix $V M$ of size $1 \times 2^n$ is represented by

$$\exists A.(\phi_1 \land \phi_2)$$

which is a formula on $A'$.

To obtain a formula over $A$ we have to rename the variables.

$$(\exists A.(\phi_1 \land \phi_2))[A/A']$$
Images by $\exists$-abstraction

**Example**

Let $A = \{a, b\}$ be the state variables.

$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$

represents the image of $\{00, 10\}$ with respect to a relation.

$\exists a.\exists b. (\neg b \land (b \leftrightarrow \neg b'))$

$\equiv \exists b. (\neg b \land (b \leftrightarrow \neg b'))$

$\equiv (\neg \top \land (\top \leftrightarrow \neg b')) \lor (\neg \bot \land (\bot \leftrightarrow \neg b'))$

$\equiv b'$

The formula $b'$ represents $\{01, 11\}$. 
Matrix multiplication by $\exists$-abstraction

Let
- $A = \{a_1, \ldots, a_n\}$,
- $A' = \{a_1', \ldots, a_n'\}$,
- $A'' = \{a_1'', \ldots, a_n''\}$,
- $\phi_1$ be a formula on $A \cup A'$ representing matrix $M_1$ and
- $\phi_2$ a formula on $A' \cup A''$ representing matrix $M_2$.

The matrices $M_1$ and $M_2$ have size $2^n \times 2^n$.

The product matrix $M_1 M_2$ is represented by

$$\exists A'. (\phi_1 \land \phi_2)$$

which is a formula on $A \cup A''$. 

$\exists$-abstraction
Matrix multiplication by \( \exists \)-abstraction

Example

Let \( \phi_1 = a \leftrightarrow \neg a' \) and \( \phi_2 = a' \leftrightarrow a'' \) represent two actions, reversing the truth-value of \( a \) and doing nothing. The sequential composition of these actions is

\[
\exists a'. \phi_1 \land \phi_2 = ((a \leftrightarrow \neg T) \land (T \leftrightarrow a'')) \lor ((a \leftrightarrow \neg \perp) \land (\perp \leftrightarrow a'')) \\
\equiv ((a \leftrightarrow \perp) \land (T \leftrightarrow a'')) \lor ((a \leftrightarrow T) \land (\perp \leftrightarrow a'')) \\
\equiv (\neg a \land a'') \lor (a \land \neg a'') \\
\equiv a \leftrightarrow \neg a''.
\]
Matrix multiplication

Multiply \((-a \leftrightarrow a') \land (-b \leftrightarrow b')\) and \((a' \leftrightarrow b'') \land (b' \leftrightarrow a'')\):

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

This is

\[
\exists a'. \exists b'.((-a \leftrightarrow a') \land (-b \leftrightarrow b') \land (a' \leftrightarrow b'') \land (b' \leftrightarrow a''))
\equiv (-a \leftrightarrow b'') \land (-b \leftrightarrow a'').
\]
Images and preimages by formula manipulation

Define $s[A'/A] = \{ \langle a', s(a) \rangle | a \in A \}$.

**Lemma**

*Let $\phi$ be a formula on $A$ and $\nu$ a valuation of $A$. Then $\nu \models \phi$ iff $\nu[A'/A] \models \phi[A'/A]$.*

**Definition**

Let $o$ be an operator and $\phi$ a formula. Define

\[
\text{img}_o(\phi) = (\exists A. (\phi \land \tau^n_d(o)))[A/A']
\]

\[
\text{preimg}_o(\phi) = \exists A'. (\tau^n_d(o) \land \phi[A'/A])
\]

\[
\text{spreimg}_o(\phi) = \forall A'. (\tau^n_d(o) \rightarrow \phi[A'/A]) \land \exists A'. \tau^n_d(o).
\]
**Theorem**

Let $T = \{ s \in S | s \models \phi \}$. Then $\{ s \in S | s \models \text{img}_o(\phi) \} = \{ s \in S | s \models (\exists A. (\phi \land \tau_A^{nd}(o)))[A/A'] \} = \text{img}_o(T)$.

**Proof.**

$s' \models (\exists A. (\phi \land \tau_A^{nd}(o)))[A/A']$

iff $s'[A'/A] \models \exists A. (\phi \land \tau_A^{nd}(o))$

iff there is valuation $s$ of $A$ s.t. $(s \cup s'[A'/A]) \models \phi \land \tau_A^{nd}(o)$

iff there is valuation $s$ of $A$ s.t. $s \models \phi$ and $(s \cup s'[A'/A]) \models \tau_A^{nd}(o)$

iff there is $s \in T$ s.t. $(s \cup s'[A'/A]) \models \tau_A^{nd}(o)$

iff there is $s \in T$ s.t. $s' \in \text{img}_o(s)$

iff $s' \in \text{img}_o(T)$. 
Preimages by formula manipulation

**Theorem**

\[ T = \{ s \in S | s \models \phi \} \]

Then \( \{ s \in S | s \models \text{preimg}_o(\phi) \} = \{ s \in S | s \models \exists A'.(\tau_A^{nd} \land \phi[A'/A]) \} = \text{preimg}_o(T). \]

**Proof.**

\[ s \models \exists A'.(\tau_A^{nd} \land \phi[A'/A]) \]

iff there is \( s'_0 : A' \to \{0, 1\} \) s.t. \((s \cup s'_0) \models \tau_A^{nd} \land \phi[A'/A]\)

iff there is \( s'_0 : A' \to \{0, 1\} \) s.t. \( s'_0 \models \phi[A'/A] \) and \((s \cup s'_0) \models \tau_A^{nd} \)

iff there is \( s' : A \to \{0, 1\} \) s.t. \( s' \models \phi \) and \((s \cup s'_0) \models \tau_A^{nd} \)

iff there is \( s' \in T \) s.t. \((s \cup s'[A'/A]) \models \tau_A^{nd} \)

iff there is \( s' \in T \) s.t. \( s' \in \text{img}_o(s) \)

iff there is \( s' \in T \) s.t. \( s \in \text{preimg}_o(s') \)

iff \( s \in \text{preimg}_o(T) \).

Above we define \( s' = s'_0[A/A'] \) (and hence \( s'_0 = s'[A'/A] \).)
Strong preimages by formula manipulation

**Theorem**

Let \( T = \{ s \in S | s \models \phi \} \). Then

\[
\{ s \in S | s \models \text{spreimg}_o(\phi) \} = \{ s \in S | s \models \forall A'.(\tau^{nd}_A(o) \rightarrow \\
\phi[A'/A]) \land \exists A'.\tau^{nd}_A(o) \} = \text{spreimg}_o(T).
\]

**Proof.**

See the lecture notes.
Corollary

Let $o = \langle c, (e_1| \cdots | e_n) \rangle$ be an operator such that all $e_i$ are deterministic. The formula $\text{spreimg}_o(\phi)$ is logically equivalent to $\text{regr}^n_o(\phi)$.

Proof.

$$\{ s \in S | s \models \text{regr}_o(\phi) \} = \text{spreimg}_o(\{ s \in S | s \models \phi \}) = \{ s \in S | s \models \text{spreimg}_o(\phi) \}.$$
## Summary of matrix/logic/relational operations

<table>
<thead>
<tr>
<th>Matrices</th>
<th>Formulas</th>
<th>State Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector $V_{1 \times n}$</td>
<td>Formula on $A$</td>
<td>Set union $\text{img}_o(T)$</td>
</tr>
<tr>
<td>Matrix $M_{n \times n}$</td>
<td>Formula on $A \cup A'$</td>
<td>Relation $\text{preimg}_o(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} + V'_{1 \times n}$</td>
<td>$\phi_1 \lor \phi_2$</td>
<td>Intersection $\text{spreimg}_o(T)$</td>
</tr>
<tr>
<td>$V_{1 \times n} \times M_{n \times n}$</td>
<td>$\phi_1 \land \phi_2$</td>
<td></td>
</tr>
<tr>
<td>$M_{n \times n} \times V_{n \times 1}$</td>
<td>$(\exists A. (\phi \land \tau_A^{nd}(o)))[A/A']$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$(\exists A'. (\tau_A^{nd}(o) \land \phi[A'/A]))$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\forall A'. (\tau_A^{nd}(o) \rightarrow \phi[A'/A]) \land \exists A'. \tau_A^{nd}(o)$</td>
<td></td>
</tr>
</tbody>
</table>
The union of images of $\phi$ with respect to all operators $o \in O$ is

$$\bigvee_{o \in O} \text{img}_o(\phi).$$

This can be computed more directly by using the disjunction $\bigvee_{o \in O} \tau_A(o)$ of the transition formulae:

$$\exists A. (\phi \land (\bigvee_{o \in O} \tau_A(o)))[A/A'].$$

Same works for preimages.
We tested plan existence by testing satisfiability of

$$
i^0 \land R_1(A^0, A^1) \land \cdots \land R_1(A^{t-1}, A^t) \land G^t$$

where $R_1(A, A') = \bigvee_{o \in O} \tau_A(o)$.

$\exists$-abstracting $A^0 \cup \cdots \cup A^{t-1}$ yields

$$\exists A^{t-1}.(\cdots \exists A^0.(i^0 \land R_1(A^0, A^1)) \land \cdots \land R_1(A^{t-1}, A^t) \land G^t).$$

This is equivalent to conjoining the $t$-fold image of $i$

$$\bigvee_{o \in O} \bigvee_{o \in O} \text{img}_o(\cdots \bigvee_{o \in O} \text{img}_o(i) \cdots)$$

with $G$ to test goal reachability in $t$ steps.

We can do the same with preimages starting from $G$. 

Image computation vs. planning by satisfiability
We tested plan existence by testing satisfiability of

\[ \iota^0 \land R_1(A^0, A^1) \land \cdots \land R_1(A^{t-1}, A^t) \land G^t \]

where \( R_1(A, A') = \bigvee_{o \in O} \tau_A(o) \).

\( \exists \)-abstracting \( A^0 \cup \cdots \cup A^{t-1} \) yields

\[ \exists A^{t-1}.(\cdots \exists A^0.((\iota^0 \land R_1(A^0, A^1)) \land \cdots \land R_1(A^{t-1}, A^t) \land G^t)). \]

This is equivalent to conjoining the \( t \)-fold image of \( \iota \)

\[ \bigvee_{o \in O} \bigvee_{o \in O} \text{img}_o(\iota) \]

with \( G \) to test goal reachability in \( t \) steps.

We can do the same with preimages starting from \( G \).
Shannon expansion

**Definition**

3-place connective if-then-else is defined by

\[
\text{ite}(a, \phi_1, \phi_2) = (a \land \phi_1) \lor (\neg a \land \phi_2)
\]

where \(a\) is a proposition.

**Definition**

Shannon expansion of a formula \(\phi\) with respect to \(a \in A\) is

\[
\phi \equiv (a \land \phi[\top/a]) \lor (\neg a \land \phi[\bot/a]) = \text{ite}(a, \phi[\top/a], \phi[\bot/a])
\]
By repeated application of Shannon expansion any propositional formula can be transformed to an equivalent formula containing no other connectives than $ite$ and propositional variables only in the first position of $ite$.

Example

\[
(a \lor b) \land (b \lor c) \\
\equiv ite(a, (\top \lor b) \land (b \lor c), (\bot \lor b) \land (b \lor c)) \\
\equiv ite(a, b \lor c, b) \\
\equiv ite(a, ite(b, \top \lor c, \bot \lor c), ite(b, \top, \bot)) \\
\equiv ite(a, ite(b, \top, ite(c, \top, \bot)), ite(b, \top, \bot)) \\
\]

### Transformation to ordered BDDs

1. Fix an ordering $a_1, \ldots, a_n$ on all propositional variables.
2. Apply Shannon expansion to all variables in this order.
3. Represent the resulting formulae as directed acyclic graphs (DAG) so that shared subformulae occur only once.

### Theorem

Let $\phi_1$ and $\phi_2$ be two ordered BDDs obtained by using the same variable ordering. Then $\phi_1 \equiv \phi_2$ if and only if $\phi_1$ and $\phi_2$ are isomorphic (the same DAG.)
Binary decision diagrams: example

\[(a \lor b) \land (b \lor c)\]
Binary decision diagrams: example

\[
(\top \lor b) \land (b \lor c) \quad \text{and} \quad (\bot \lor b) \land (b \lor c)
\]
Binary decision diagrams: example

\[ b \vee c \rightarrow a \rightarrow b \]

1 0
Binary decision diagrams: example

\[ a \rightarrow b \]

\[ \top \lor c \rightarrow 1 \]

\[ \perp \lor c \rightarrow 0 \]
Binary decision diagrams: example
Binary decision diagrams: example
### Satisfiability algorithms vs. BDDs

#### Comparison: formula size, runtime

<table>
<thead>
<tr>
<th>technique</th>
<th>size of $R_1(P, P')$</th>
<th>runtime for plan length $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfiability</td>
<td>not a problem</td>
<td>exponential in $n$</td>
</tr>
<tr>
<td>BDDs</td>
<td>major problem</td>
<td>less dependent on $n$</td>
</tr>
</tbody>
</table>

#### Comparison: resource consumption

<table>
<thead>
<tr>
<th>technique</th>
<th>critical resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfiability</td>
<td>runtime</td>
</tr>
<tr>
<td>BDDs</td>
<td>memory</td>
</tr>
</tbody>
</table>

#### Comparison: application domain

<table>
<thead>
<tr>
<th>technique</th>
<th>types of problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>satisfiability</td>
<td>lots of state variables, short plans</td>
</tr>
<tr>
<td>BDDs</td>
<td>few state variables, long plans</td>
</tr>
</tbody>
</table>
Properties of CPC normal forms

Trade-offs between different CPC normal forms

Normal forms that allow faster reasoning are more expensive to construct from an arbitrary propositional formula and may be much bigger.

Properties of different normal forms

<table>
<thead>
<tr>
<th></th>
<th>∨</th>
<th>∧</th>
<th>¬</th>
<th>φ ∈ TAUT?</th>
<th>φ ∈ SAT?</th>
<th>φ ≡ φ'?</th>
</tr>
</thead>
<tbody>
<tr>
<td>circuits formulae</td>
<td>poly</td>
<td>poly</td>
<td>poly</td>
<td>co-NP-hard</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>DNF</td>
<td>poly</td>
<td>exp</td>
<td>exp</td>
<td>co-NP-hard</td>
<td>in P</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>CNF</td>
<td>exp</td>
<td>poly</td>
<td>exp</td>
<td>in P</td>
<td>NP-hard</td>
<td>co-NP-hard</td>
</tr>
<tr>
<td>BDD</td>
<td>exp</td>
<td>exp</td>
<td>poly</td>
<td>in P</td>
<td>in P</td>
<td>in P</td>
</tr>
</tbody>
</table>

For BDDs one ∨/∧ is polynomial time/size (size is doubled) but repeated ∨/∧ lead to exponential size.