Observability

- Robot can see and hear only the immediate surroundings.
- Poker player cannot see the opponents’ cards.
- Formalization: only a subset of the state variables are observable.
A succinct transition system is a 5-tuple $\langle A, I, O, G, V \rangle$ consisting of

- a set $A$ of state variables,
- a propositional formula $I$ over $A$,
- a set $O$ of operators,
- a propositional formula $G$ over $A$, and
- a set $V \subseteq A$ of observable state variables.
Restrictions on observability

Let \( \langle A, I, O, G, V \rangle \) be a problem instance in conditional planning.

1. If \( A = V \), the problem instance is **fully observable**.
2. If \( V = \emptyset \), the problem instance is **unobservable**.
3. If there are no restrictions on \( V \) then the problem instance is **partially observable**.
Observational classes

- When variables in $V = \{a_1, \ldots, a_m\}$ are observable and others are not then it is not possible to distinguish between states $s$ and $s'$ such that $s(a) = s'(a)$ for all $a \in V$.

- Let $S$ be the set of all states (valuations of $A$). Observability partitions $S$ to classes $S_1, S_2, \ldots, S_n$ of observationally indistinguishable states so that
  1. $S = S_1 \cup S_2 \cup \cdots \cup S_n$,
  2. $S_i \cap S_j = \emptyset$ for any $\{i, j\} \subset \{1, \ldots, n\}$ such that $i \neq j$. 
Observational classes: cardinality

Full observability: \( S = \{s_1, \ldots, s_n\} \) is partitioned to singleton classes
\( S_1 = \{s_1\}, S_2 = \{s_2\}, \ldots, S_n = \{s_n\} \).

Unobservability: The partition has only one class \( S_1 = S \) consisting of all the \( n \) states.

Partial observability: The number of partitions and the cardinalities of \( S_i \) may be anywhere between 1 and \( n \).
Belief states and the belief space

- Current state is not in general known during plan execution. Instead, the a set of possible current states is known.
- A set of possible current states is a belief state.
- The set of all belief states is the belief space.
- If there are $n$ states and none of them can be observationally distinguished from another, then there are $2^n - 1$ belief states.
First we consider the second special case of planning with partial observability: planning without observability.

Plans are sequences of actions because observations are not possible, actions cannot depend on the nondeterministic events, and hence the same actions have to be taken no matter what happens.

Techniques needed for planning without observability can often be generalized to the general partially observable case.
The belief space

1. Let $B$ be a belief state (a set of states).
2. Operator $o$ is executable in $B$ if it is executable in every $s \in B$.
3. When $o$ is executed, possible next states are $T = img_o(B)$.
4. Observations correspond to one of the observational class, $S_j$.
5. New belief state is $B' = img_o(B) \cap S_j$. 
Example (Next slide)

Belief space generated by states over two Boolean state variables.

\( n = 2 \) state variables, \( 2^n = 4 \) states, \( 2^{2n} = 16 \) belief states

**red action**: complement the value of the first state variable

**blue action**: assign a random value to the second state variable
The belief space
Example
A robot without any sensors, anywhere in a room of size 7 × 8.

Actions: go North, South, East, West

Plan for getting out: 6 × West, 7 × North, 1 × East, 1 × North

On the next slides we depict one possible location of the robot (●) and the change in the belief state at every execution step.
Example: belief state initially
Example: belief state after W
Example: after WW

door
Example: after WWW
Example: after WWWWW
Example: after WWWWW

door

The diagram shows a grid with a black dot at the bottom, indicating the starting position, and a label 'door' to the left.
Example: after WWWWWW

![Diagram of a grid with a door at the top and a black dot at the bottom]

**door**

---

**AI Planning**

**Motivation**

**Formalization**

**Heuristic search**

**Planning by QBF**
Example: after WWWWWWN
Example: after WWWWWWWNN

![Diagram with a grid and a door marked]

**door**
Example: after WWWWWWNNN

door
Example: after WWWWWWWNNNNN

door
Example: after WWWWWWWNNNNN
Example: after WWWWWWWNWWNNNNN
Example: after WWWWWWWNNNNNNNNN

![Diagram of a grid with a door labeled as 'door']
Example: after WWWWWWNNNNNNNNNNE
Example: after WWWWWWNNNNNNNEN
Sorting networks consist of comparator-swapper elements that compare the values of two inputs and output them sorted: if first input is bigger than the second, then they are swapped, otherwise the outputs are the inputs. A sorting network for $n$ inputs should sort any input sequence.
The belief space
Sorting networks

Theorem

If a sorting network correctly sorts any sequence of binary digits 0 and 1, then it correctly sorts any input sequence.

3-input sorting networks can be formalized as a succinct transition system \( \langle A, I, O, G, V \rangle \) where

\[
A = \{a_0, a_1, a_2\} \\
I = \top \\
O = \{o_{01}, o_{02}, o_{12}\} \\
G = (a_0 \rightarrow a_1) \land (a_1 \rightarrow a_2) \\
o_{01} = \langle \top, (a_0 \land \neg a_1) \triangleright (\neg a_0 \land a_1) \rangle \\
o_{02} = \langle \top, (a_0 \land \neg a_2) \triangleright (\neg a_0 \land a_2) \rangle \\
o_{12} = \langle \top, (a_1 \land \neg a_2) \triangleright (\neg a_1 \land a_2) \rangle
\]
The belief space
Sorting networks

A plan for the 3-input sorting network is $o_{12}, o_{02}, o_{01}$.
The initial states are 000, 001, 010, 011, 100, 101, 110, 111.
The goal states are 000, 001, 011, 111.
The belief state evolves as follows.

\[
\begin{align*}
000, & 001, 010, 011, 100, 101, 110, 111 & & \text{initially} \\
000, & 001, 010, 011, 100, 101, 110, 111 & & \text{after } o_{12} \\
000, & 001, 010, 011, 100, 101, 110, 111 & & \text{after } o_{02} \\
000, & 001, 010, 011, 100, 101, 110, 111 & & \text{after } o_{01}
\end{align*}
\]
Find an operator sequence $o_1, \ldots, o_n$ that reaches a state satisfying $G$ starting from any state satisfying $I$.

1. $o_1$ must be applicable in all states $B_0 = \{ s \in S | s \models I \}$ satisfying $I$.
2. $o_2$ must be applicable in all states in $B_1 = \text{img}_{o_1}(B_0)$.
3. $o_i$ must be applicable in all states in $B_i = \text{img}_{o_i}(B_{i-1})$ for all $i \in \{1, \ldots, n\}$.

Terminal states must be goal states:

$B_n \subseteq \{ s \in S | s \models G \}$.
Algorithms for deterministic planning can be lifted to the level of belief states.

We can do forward search in the belief space with \( \text{img}_o(B) \), backward search with \( \text{spreimg}_o(B) \).

We have already introduced implementation techniques that allow representing belief states \( B \) as formulae \( \phi \) and computing images and preimages respectively as \( \text{img}_o(\phi) \) and \( \text{spreimg}_o(\phi) \).

Size of belief space is exponentially bigger than the size of the corresponding state space. For \( n \) states there are \( 2^n \) belief states.
Algorithms for unobservable problems

Heuristic search

progression/regression + heuristic search (A*, IDA*, simulated annealing, ...)

Heuristics:

- heuristic 1: backward distances (for forward search)
- heuristic 2: cardinality of belief state (for both forward and backward search)
Use **backward distances** of states as a heuristic:

\[
D_0 = G \\
D_{i+1} = D_i \cup \bigcup_{o \in O} spreimg_o(D_i) \text{ for all } i \geq 1
\]

A **lower bound** on plan length for belief state \( B \) is \( j \) if \( B \subseteq D_j \) and \( B \nsubseteq D_{j-1} \) for \( j \geq 1 \).

This is an **admissible heuristic** (does not overestimate the distance).
Backward search: Prefer operators that increase the size of the belief state, i.e. find a plan suffix that reaches a goal state from more starting states.

Forward search: Prefer operators that decrease the size of the belief state, i.e. reduce the uncertainty about the current state and make reaching goals easier. For the room navigation example this heuristic works very well until the size of the belief state is 1.

This heuristic is not admissible.

Computing the cardinality of a belief state from its BDD representation takes polynomial time. (Propositional logic in general: problem is NP-hard.)
Translation into quantified Boolean formulae (QBF)

Why not translation into propositional logic?

- We need to say that there is a plan such that ...
  This is like the satisfiability problem in CPC: there is a valuation...

- We need to say that for all executions ...
  This is like the validity problem in CPC: for all valuations...

- Consequence: the problem does not seem to be in NP nor in co-NP.
Quantified Boolean formulae

Definition

- If $\phi$ is a propositional formula and $\sigma$ is a sequence of $\exists p$ and $\forall p$, one for every $p \in A$, then $\sigma \phi$ is a QBF.
- A formula $\exists x \phi$ is true if and only if $\phi[\top/x] \lor \phi[\bot/x]$ is true. (Equivalently, $\phi[\top/x]$ is true or $\phi[\bot/x]$ is true.)
- A formula $\forall x \phi$ is true if and only if $\phi[\top/x] \land \phi[\bot/x]$ is true. (Equivalently, $\phi[\top/x]$ is true and $\phi[\bot/x]$ is true.)

The most important algorithms for evaluating QBF are based on AND/OR tree search, $\forall$-variables correspond to AND-nodes and $\exists$-variables to OR-nodes.
The evaluation problem of QBF generalizes both the satisfiability and validity/tautology problems of the propositional logic. The latter are respectively NP-complete and co-NP-complete whereas the former is PSPACE-complete.

**Example**

The formulae $\forall x \exists y (x \leftrightarrow y)$ and $\exists x \exists y (x \land y)$ are true. The formulae $\exists x \forall y (x \leftrightarrow y)$ and $\forall x \forall y (x \lor y)$ are false.
There is a sequence of operators so that for all initial states and nondeterministic choices there is an execution that reaches a goal state.

\[ \exists o_1 \cdots o_m \cdots o_{t-1} \cdots o_n \]
\[ \forall a_1 \cdots a_n x_1 \cdots x_k \cdots x_{t-1} \cdots x_k \]
\[ \exists a_1 \cdots a_n \cdots a_1 \cdots a_n \]
\[ I^0 \rightarrow (R_3(A^0, A^1, O^0, X^0) \land \cdots \land R_3(A^{t-1}, A^t, O^{t-1}, X^{t-1}) \land G^t) \]
We replace nondeterministic choice by dependence of the effects on values of “hidden” state variables $x$.

Nondeterministic effect $e_1 | e_2 | \cdots | e_n$ roughly corresponds to a number of conditional effects:

$$(\phi_1 \triangleright e_1) \land (\phi_2 \triangleright e_2) \land \cdots \land (\phi_n \triangleright e_n).$$

Formulae $\phi_i$ refer to valuations of some unknown “hidden” state variables $x_1, \ldots, x_m$ (different at every time point).

For $n$ choices we have $m = \lceil \log_2 n \rceil$ variables $x_j$. 
The translation $\tau_{A}^{nd}(o)$ of individual operators and the formulae $\tau_{a}^{nd}(o) \lor \cdots \lor \tau_{a}^{nd}(o)$ do not allow to distinguish between controllable and uncontrollable choices. Choice of operator is controllable, but the choice between nondeterministic alternatives is not.

We give a new translation that distinguishes between controllability and uncontrollability.

This translation also allows parallel operator application.
We consider **binary nondeterminism** only so that every nondeterministic choice corresponds to the values of one propositional variable. Effects $a|b|c|d$ can always be equivalently represented as $(a|b)|(c|d)$.

For $n$ nondeterministic choices we need $\lceil \log_2 n \rceil$ auxiliary variables.

For $(a|b)|(c|d)$ the variable $x_1$ chooses between $a|b$ and $c|d$.

After $a|b$ or $c|d$ has been chosen, the respective choices between $a$ and $b$, and $c$ and $d$ are represented by a second variable $x_{11}$. 
Let $e$ be an effect and $\sigma$ a sequence of integers. Sequences $\sigma$ identify nondeterministic choice inside an operator. Define $\text{EPC}_l^{nd}(e, \sigma)$ as follows.

\[
\begin{align*}
\text{EPC}_l^{nd}(e, \sigma) &= \text{EPC}_l(e) \text{ if } e \text{ is deterministic} \\
\text{EPC}_l^{nd}(e_1 | e_2, \sigma) &= (x_{\sigma} \land \text{EPC}_l^{nd}(e_1, \sigma_1)) \\
&\quad \lor (\neg x_{\sigma} \land \text{EPC}_l^{nd}(e_2, \sigma_1)) \\
\text{EPC}_l^{nd}(e_1 \land \cdots \land e_n, \sigma) &= \text{EPC}_l^{nd}(e_1, \sigma_1) \lor \cdots \lor \text{EPC}_l^{nd}(e_n, \sigma_n)
\end{align*}
\]
Nondeterministic operators in CPC

Example

\[ EPC_{a}^{nd}((\langle a| b\rangle|\langle c| d\rangle), 1) = (x_1 \land EPC_{a}^{nd}((\langle a| b\rangle), 1)) \]
\[ \lor (\neg x_1 \land EPC_{a}^{nd}((\langle c| d\rangle), 1)) \]
\[ \equiv (x_1 \land EPC_{a}^{nd}((\langle a| b\rangle), 1)) \]
\[ \equiv (x_1 \land (x_{11} \land EPC_{a}^{nd}(a, 1)))) \]
\[ \lor (\neg x_{11} \land EPC_{a}^{nd}(b, 1)) \]
\[ \equiv x_1 \land x_{11} \]

\[ EPC_{b}^{nd}((\langle a| b\rangle|\langle c| d\rangle), 1) = (x_1 \land EPC_{b}^{nd}((\langle a| b\rangle), 1)) \]
\[ \lor (\neg x_1 \land EPC_{b}^{nd}((\langle c| d\rangle), 1)) \]
\[ \equiv (x_1 \land EPC_{b}^{nd}((\langle a| b\rangle), 1)) \]
\[ \equiv (x_1 \land (x_{11} \land EPC_{b}^{nd}(a, 1)))) \]
\[ \lor (\neg x_{11} \land EPC_{b}^{nd}(b, 1)) \]
\[ \equiv x_1 \land \neg x_{11} \]
Frame axioms

Let $e_1, \ldots, e_n$ be the effects of $o_1, \ldots, o_n$ respectively.

$$(a \land \neg a') \rightarrow ((o_1 \land \text{EPC}^{nd}_{=a}(e_1, 1)) \lor \cdots \lor (o_n \land \text{EPC}^{nd}_{=a}(e_n, n)))$$

$$(\neg a \land a') \rightarrow ((o_1 \land \text{EPC}^{nd}_{a}(e_1, 1)) \lor \cdots \lor (o_n \land \text{EPC}^{nd}_{a}(e_n, n)))$$

Precondition and effect axioms

Let $i$ be the index of operator $o = \langle c, e \rangle \in O$. The formula that describes this operator are

$$(o \rightarrow c) \land 
\bigwedge_{a \in A} (o \land \text{EPC}^{nd}_{a}(e, i) \rightarrow a') \land 
\bigwedge_{a \in A} (o \land \text{EPC}^{nd}_{a}(e, i) \rightarrow \neg a').$$
Consider the operators

\[ o_1 = \langle \neg a, (\langle x_{11}, \neg x_{11} \rangle \mid (c \triangleright d)) \land (\langle x_{12}, \neg x_{12} \rangle \mid x_2) \rangle \]

\[ o_2 = \langle \neg b, (\langle d \triangleright b \rangle \mid (\neg x_{21}, c) \rangle \mid (\neg x_2, \neg b) \rangle \]

<table>
<thead>
<tr>
<th>var</th>
<th>made true by ( o_1 ) if</th>
<th>made true by ( o_2 ) if</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( x_{12} )</td>
<td>( \neg x_2 )</td>
</tr>
<tr>
<td>( b )</td>
<td>( x_{11} )</td>
<td>( x_2 \land x_{21} \land d )</td>
</tr>
<tr>
<td>( c )</td>
<td>( \neg x_{12} )</td>
<td>( x_2 \land \neg x_{21} )</td>
</tr>
<tr>
<td>( d )</td>
<td>( \neg x_{11} \land c )</td>
<td></td>
</tr>
</tbody>
</table>

Example
Now $R_3(\{a, b, c, d\}, \{a', b', c', d'\}, \{o_1, o_2\}, \{x_{11}, x_{12}, x_2, x_{21}\})$ is the conjunction of the following formulae.

\[
\neg(a \land \neg a') \quad (\neg a \land a') \rightarrow ((o_1 \land x_{12}) \lor (o_2 \land \neg x_2))
\]
\[
\neg(b \land \neg b') \quad (\neg b \land b') \rightarrow ((o_1 \land x_{11}) \lor (o_2 \land x_2 \land x_{21} \land d))
\]
\[
\neg(c \land \neg c') \quad (\neg c \land c') \rightarrow ((o_1 \land \neg x_{12}) \lor (o_2 \land x_2 \land \neg x_{21}))
\]
\[
\neg(d \land \neg d') \quad (\neg d \land d') \rightarrow (o_1 \land \neg x_{11} \land c)
\]
\[
 o_1 \rightarrow \neg a
\]
\[
 (o_1 \land x_{12}) \rightarrow a'
\]
\[
 (o_1 \land \neg x_{12}) \rightarrow c'
\]
\[
 o_2 \rightarrow \neg b
\]
\[
 (o_2 \land \neg x_2) \rightarrow a'
\]
\[
 (o_2 \land x_2 \land \neg x_{21}) \rightarrow c'
\]
\[
 (o_2 \land x_2 \land x_{21} \land d) \rightarrow b'
\]