# Observability and sensing (June 20, 2005)

## Motivation

- 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.

## Formalization

A succinct transition system is a 5-tuple \( \langle A, I, O, G, V \rangle \) consisting of:

1. A set \( A \) of state variables,
2. A propositional formula \( I \) over \( A \),
3. A set \( O \) of operators,
4. A propositional formula \( G \) over \( A \), and
5. A set \( V \subseteq A \) of observable state variables.

## 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

- **Observability:** \( S \) 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 \( \{i, j\} \subseteq \{1, \ldots, n\} \) 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.

## Planning without observability

- 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.
Let $B$ be a belief state (a set of states).

Operator $o$ is executable in $B$ if it is executable in every $s \in B$.

Observations correspond to one of the observational class, $S_j$.

New belief state is $B^o = \text{img}_o(B) \cap S_j$.

Example: after WWW

Example: after WWWWWWW

Example: after WWWWNNNN

Example: after WWWWWWWNNNNE
The belief space
Sorting networks

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.

\[
\begin{align*}
A &= \{a_0, a_1, a_2\} \\
I &= \mathbb{T} \\
O &= \{a_{01}, a_{02}, a_{12}\} \\
G &= \langle a_0, \neg a_1 \rangle \land \langle \neg a_0, a_1 \rangle \\
o_{a_1} &= \langle \langle a_0 \land \neg a_1 \rangle \lor \langle \neg a_0 \land a_1 \rangle \rangle \\
o_{a_2} &= \langle \langle a_0 \land \neg a_2 \rangle \lor \langle \neg a_0 \land a_2 \rangle \rangle \\
o_{a_{12}} &= \langle \langle a_1 \land \neg a_2 \rangle \lor \langle \neg a_1 \land a_2 \rangle \rangle
\end{align*}
\]

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 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.

\[
\begin{align*}
000, 001, 010, 011, 100, 101, 110, 111 & \text{ initially} \\
000, 001, 010, 011, 100, 101, 110, 111 & \text{ after } o_{12} \\
000, 001, 010, 011, 100, 101, 110, 111 & \text{ after } o_{a_2} \\
000, 001, 010, 011, 100, 101, 110, 111 & \text{ after } o_{a_1}
\end{align*}
\]

Algorithms for unobservable problems

1. Find an operator sequence \( o_1, \ldots, o_n \), that reaches a state satisfying \( G \) starting from any state satisfying \( I \).

2. \( o_1 \) must be applicable in all states \( B_0 = \{ s \in S | s \models I \} \) satisfying \( I \).

\( o_2 \) must be applicable in all states in \( B_1 = \text{img}_{o_1}(B_0) \).

\( o_3 \) must be applicable in all states in \( B_i = \text{img}_{o_1}(B_{i-1}) \) for all \( i \in \{1, \ldots, n\} \).

Terminal states must be goal states: \( B_n \subseteq \{ s \in S | s \models G \} \).

Algorithms for unobservable problems

Heuristic search

Heuristic search

Heuristic search

Heuristic search

Cardinality heuristics

Algorithms for unobservable problems

Cardinality heuristics

Use backward distances of states as a heuristic:

\[
D_{i+1} = G \cup \bigcup_{i \geq 1} \text{spreimg}_{o_1}(D_i) \quad \text{for all } i \geq 1
\]

A lower bound on plan length for belief state \( B \) is \( j \) if \( B \subseteq D_j \) and \( B \not\subseteq D_{j-1} \) for \( j \geq 1 \).

This is an admissible heuristic (does not overestimate the distance).
Translation into quantified Boolean formulas (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.

Example
The formulae $\forall x \exists y (x \rightarrow y)$ and $\exists x \forall y (x \land y)$ are true.
The formulae $\exists x \forall y (x \rightarrow y)$ and $\forall x \exists y (x \lor y)$ are false.

Nondeterministic operators in CPC

- We replace nondeterministic choice by dependence of the effects
  on values of "hidden" state variables $x$.
- Nondeterministic effect $e_1[x_1] \cdots e_n$ roughly corresponds to a
  number of conditional effects:

$$\Phi_1 \triangleright e_1 \land \Phi_2 \triangleright e_2 \land \cdots \land \Phi_m \triangleright e_m.$$  

Formulas $\Phi_i$ refer to valuations of a some unknown "hidden" state
variables $x_1, \ldots, x_m$ (different at every time point).
For $n$ choices we have $m = \log_2 n$ variables $x_j$.

Nondeterministic operators in CPC

- 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 $x_{11}$.

Nondeterministic operators in CPC

Let $e$ be an effect and $\sigma$ a sequence of integers. Sequences $\sigma$ identify
nondeterministic choice inside an operator.
Define $EPC_{\sigma}^e(e, \sigma)$ as follows.

$$EPC_{\sigma_1}^e(e_1, \sigma_1) = EPC(e) \text{ if } e \text{ is deterministic}$$
$$EPC_{\sigma_1}^e(e_1[x_2], \sigma_1) = (x_2 \land EPC_{\sigma_1}^d(e_1, \sigma_1))$$
$$EPC_{\sigma_1}^e(e_1 \cdots \land e_m, \sigma) = EPC_{\sigma_1}^e(e_1, \sigma_1) \lor \cdots \lor EPC_{\sigma_1}^e(e_m, \sigma_m)$$

Nondeterministic operators in CPC

- The translation $\Phi^D_{\sigma_1}(\sigma)$ of individual operators and the formulae
  $\sigma_1^D(\sigma)$ 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.

Nondeterministic operators in CPC

- If $\phi$ is a propositional formula and $\sigma$ is a sequence of $\exists y$ and $\forall p$,
  one for every $p \in A$, then $\phi[\sigma]$ is a QBF.
- A formula $\exists \phi$ 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 $\forall \phi$ 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.
**Example**

\[ EPC_n^{ab}((a|b)|(c|d), 1) = (x_1 \land EPC_n^{ab}((a|b), 1)) \lor (x_2 \land EPC_n^{ab}((c|d), 1)) \]
\[ \lor (x_3 \land EPC_n^{ab}((a|b), 1)) \]
\[ \lor (x_4 \land (x_5 \land EPC_n^{ab}((a, 1)))) \]
\[ \lor (x_6 \land (x_7 \land EPC_n^{ab}((c, 1)))) \]
\[ \lor (x_8 \land (x_9 \land EPC_n^{ab}((d, 1)))) \]
\[ \lor (x_{10} \land (x_{11} \land EPC_n^{ab}((a, 1)))) \]
\[ \lor (x_{12} \land (x_{13} \land EPC_n^{ab}((c, 1)))) \]
\[ \lor (x_{14} \land (x_{15} \land EPC_n^{ab}((d, 1)))) \]

**Frame axioms**

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

\((-a \land a') \rightarrow (o_1 \land EPC_n^{ab}((e_1, 1)) \lor \cdots \lor (o_n \land EPC_n^{ab}((e_n, 1))))
\((-a \land a') \rightarrow (o_1 \land EPC_n^{ab}((e_1, 1)) \lor \cdots \lor (o_n \land EPC_n^{ab}((e_n, 1))))

**Precondition and effect axioms**

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

\[(a \land c) \land
\land_{i \in O}(a \land EPC_n^{ab}((e, i)) \rightarrow a') \land
\land_{i \in O}(a \land EPC_n^{ab}((e, i)) \rightarrow a').

**Example**

Consider the operators

\[
o_1 = (\neg a, b, (c \lor d) \land (\neg c \lor \neg d))
\]
\[
o_2 = (\neg b, (d \lor b) \land (c \lor \neg a))
\]

<table>
<thead>
<tr>
<th>var</th>
<th>made true by ( o_1 ) if</th>
<th>made true by ( o_2 ) if</th>
</tr>
</thead>
</table>
| \( a \) | \( x_1 \lor x_2 \lor x_3 \lor x_4 \) | \( x_2 \land x_3 \land d \)
| \( b \) | \( x_3 \land x_4 \) | \( x_2 \land x_3 \land d \)
| \( c \) | \( x_1 \lor x_2 \lor x_3 \lor x_4 \) | \( x_2 \land x_3 \land d \)
| \( d \) | \( x_1 \lor x_2 \lor x_3 \lor x_4 \) | \( x_2 \land x_3 \land d \)