**Data structure**

**DEFINITION** Let $\Pi = \langle C_1, \ldots, C_n \rangle$ be a partition of the set of all states. Then a **factored belief space** is $\langle G_1, \ldots, G_n \rangle$ where $s \subseteq s'$ for no $\{s, s'\} \subseteq G_i$ and $G_i \subseteq 2^{C_i}$ for all $i \in \{1, \ldots, n\}$.

*Intuitively, a factored belief space is a set of belief states, partitioned to subsets corresponding to the observational classes.*

The factored representation of one set $S$ of states is $\mathcal{F}(S) = \langle \{C_1 \cap S\}, \ldots, \{C_n \cap S\} \rangle$.

**Data structure: combination**

When we have two sets of belief states in the factored form, we may combine them and keep the result in the factored form.

**DEFINITION** Let $G = \langle G_1, \ldots, G_n \rangle$ and $H = \langle H_1, \ldots, H_n \rangle$ be factored belief spaces. Define $G \mathbin{\uplus} H$ as $\langle H_1 \mathbin{\cup} H_1, \ldots, G_n \mathbin{\cup} H_n \rangle$, where the operation $\mathbin{\cup}$ takes union of two sets of sets and eliminates sets that are not set-inclusion maximal. It is formally defined as $G \mathbin{\uplus} H = \{ R \subseteq G \mathbin{\cup} H \mid \text{no } K \subseteq G \mathbin{\cup} H \}$.

**Data structure: sets of states represented by**

A factored belief space $G = \langle G_1, \ldots, G_n \rangle$ can be viewed as representing the set of sets of states

$\text{flat}(G) = \{s_1 \cup \cdots \cup s_n \mid s_i \in G_i \text{ for all } i \in \{1, \ldots, n\}\}$.

The cardinality of $\text{flat}(G)$ is $|G_1| \cdot |G_2| \cdots |G_n|$.

**Data structure: inclusion**

**DEFINITION** A fbs $G$ is included in fbs $H$ ($G \subseteq H$) if for all $S \in \text{flat}(G)$ there is $S' \in \text{flat}(H)$ such that $S \subseteq S'$.

*Now $S \in \text{flat}(G)$ if and only if $\mathcal{F}(S) \subseteq G$.*

**THEOREM** Testing $G \subseteq H$ for factored belief spaces $G$ and $H$ is polynomial time.

**PROOF** Testing $\langle G_1, \ldots, G_n \rangle \subseteq \langle H_1, \ldots, H_n \rangle$ is simply by testing whether for all $i \in \{1, \ldots, n\}$ and all $s \in G_i$ there is $t \in H_i$ such that $s \subseteq t$. 
Finding new belief states

PROCEDURE findnew(o,A,F,H);
  IF F = {} AND spreimg(o) ∉ flat(H) THEN RETURN A;
  IF F = {} THEN RETURN ∅;
  F is \{f_1, \ldots, f_m\}, F_2, \ldots, F_k\} for k ≥ 1;
  FOR i := 1 TO m DO
    Z := findnew(o,A\[f\_{i}\],\(\bigcup\{\{\bigcup F_2, \ldots, F_k\}\}\),H);
    IF Z ≠ ∅ THEN RETURN Z;
  END;
  RETURN ∅;

Complexity of finding new belief states

THEOREM Testing whether \(G = \langle G_1, \ldots, G_n\rangle\) contains a set \(S\) such that \(\text{spreimg}_o(S)\) is not in \(G\) is NP-complete. This holds even for deterministic operators \(o\).

PROOF: Membership in NP is trivial: nondeterministically choose \(s_i \in G_i\) for every \(i \in \{1, \ldots, n\}\), compute the preimage \(r\) of \(s_1 \cup \cdots \cup s_n\), verify that \(r \cap C_i\) for some \(C_i\) is not in \(G_i\).

A planning algorithm: plan(I,O,G);

\(H := \mathcal{F}(G)\); progress := true;
WHILE progress and \(I \not\subseteq S\) for all \(S \in \text{flat}(H)\) DO
  progress := false;
  FOR EACH \(o \in O\) DO
    S := findnew(o,\(\emptyset\),H,H);
    IF S ≠ ∅ THEN
      BEGIN
        H := H \(\oplus\) \(\mathcal{F}(\text{spreimg}_o(S))\);
        progress := true;
      END; END; END;
  IF I \subseteq S for some S \in \text{flat}(H) THEN plan found;

A planning algorithm: plan(I,O,G);

\(H := \mathcal{F}(G)\); progress := true;
WHILE progress and \(I \not\subseteq S\) for all \(S \in \text{flat}(H)\) DO
  progress := false;
  FOR EACH \(o \in O\) DO
    S := findnew(o,\(\emptyset\),H,H);
    IF S ≠ ∅ THEN
      BEGIN
        H := H \(\oplus\) \(\mathcal{F}(\text{spreimg}_o(S))\);
        progress := true;
      END; END; END;
  IF I \subseteq S for some S \in \text{flat}(H) THEN plan found;
**EXPSPACE-completeness of UO planning**

THEOREM 1. The problem of testing the existence of a plan for problem instances without observability is EXPSPACE-hard.

Proof idea: Simulate deterministic Turing machines with an exponential space bound.

THEOREM 2. The problem of testing the existence of a plan for problem instances without observability is in EXPSPACE.

Proof idea: Adapt the proof PSPACE-membership proof of deterministic planning to work at the level of belief states (easy!)

**EXPSPACE-hardness of UO planning**

Problem: exponentially many tape cells!! Representing them all requires exponentially many state variables. Reduction not polynomial time!!

Solution: Randomized test whether the plan describes an execution of the TM.

1. For every execution randomly choose a watched tape cell.
2. Check that the plan correctly represents the watched tape cell. (Because of unobservability the plan cannot “see” what the watched tape cell is and it always has to simulate correctly.)

Let \((\Sigma, Q, \delta, q_0, g)\) be any deterministic Turing machine with an exponential space bound \(e(x)\).

Let \(\sigma\) be an input string of length \(n\). We denote the \(i\)th symbol of \(\sigma\) by \(\sigma_i\).

The Turing machine may use space \(e(n)\), and for encoding numbers from 0 to \(e(n) + 1\) corresponding to the tape cells we need \(m = \lceil \log_2(e(n) + 2) \rceil\) bits.

The set \(P\) of state variables in the problem instance consists of

1. \(q \in Q\) for denoting the internal states of the TM,
2. \(w_i\) for \(i \in \{0, \ldots, m-1\}\) for representing the watched tape cell \(j \in \{0, \ldots, e(n) + 1\}\),
3. \(s \in \Sigma \cup \{\square\}\) for the contents of the watched tape cell,
4. \(h_i\) for \(i \in \{0, \ldots, m-1\}\) for representing the position of the R/W head \(j \in \{0, \ldots, e(n) + 1\}\).
**EXPSPACE-hardness of UO planning**

In the initial state any tape cell could be the watched one. \( I \) is the conjunction of the following formulae.

1. \( q_0 \)
2. \( \neg q \) for all \( q \in Q \setminus \{q_0\} \).
3. Formulae for having the contents of the watched tape cell in state variables \( \Sigma \cup \{\|, \square\} \).

\[
\begin{align*}
| & \iff (w = 0) \\
\square & \iff (w > n) \\
s & \iff \bigvee_{i \in \{1, \ldots, n\}, \sigma_i = \epsilon} (w = i) \text{ for all } s \in \Sigma
\end{align*}
\]

4. \( h = 1 \) for the initial position of the R/W head.

\( w = i, w > i \) denote the formulae for testing integer equality and inequality of the numbers encoded by \( w_0, w_1, \ldots \).

Later we will use effects \( h := h + 1 \) and \( h := h - 1 \) that increment and decrement the number encoded by \( h_0, h_1, \ldots \).

---

**EXPSPACE-hardness of UO planning**

The goal is the following formula.

\[
G = \bigvee \{q \in Q | g(q) = \text{accept}\}
\]

---

**EXPSPACE-hardness of UO planning**

For all \( (s, q) \in (\Sigma \cup \{\|, \square\}) \times Q \) and \( (s', q', m) \in (\Sigma \cup \{\}) \times Q \times \{L, N, R\} \) define the effect \( \tau_{s,q}(s', q', m) \) as \( \alpha \land \kappa \land \theta \).

\( \alpha = \) effect for change in current tape cell

\( \kappa = \) effect for change in state of TM

\( \theta = \) effect for tape movement
\textbf{EXPSPACE-hardness of UO planning, }\alpha

\(\alpha\) describes what happens to the tape symbol under the R/W head.
\(\alpha = T\) if the tape cell does not change, i.e. \(s = s'\).
Otherwise,
\[\alpha = (h = w) \triangleright (\neg s \land s')\]
to change the watched tape cell.

\textbf{EXPSPACE-hardness of UO planning, }\kappa

\(\kappa\) describes the change to the internal state of the TM.
\[\kappa = \top \text{ if } q = q'\]
\[\kappa = \neg q \land q' \text{ if } q \neq q' \text{ and movement } \neq R\]
\[\kappa = \neg q \land ((h < e(n)) \triangleright q') \text{ movement is } R \text{ and } q \neq q'\]
The condition \(h < e(n)\) prevents reaching an accepting state if the space bound is violated.

\textbf{EXPSPACE-hardness of UO planning, }\theta

\(\theta\) describes the movement of the R/W head. Either there is movement to the left, no movement, or movement to the right. Hence
\[\theta = \begin{cases} h := h - 1 & \text{if } m = L \\ T & \text{if } m = N \\ h := h + 1 & \text{if } m = R \end{cases}\]

\textbf{EXPSPACE-hardness of UO planning}

Consider \((s, q) \in \Sigma \times Q\). If \(g(q) = \exists\) and \(\delta(s, q) = (s', q', m)\), then define
\[o_{s,q} = (((h \neq w) \lor s) \land q \land (h \leq e(n)), \tau_{s,q}(s', q', m))\]
The condition \((h \neq w) \lor s\) is the key to the EXPSPACE-hardness proof: If the plan tries to cheat here, then the operator is not applicable on some execution, and the plan is not a valid plan.
**EXPSPACE-hardness of UO planning**

Now the problem instance has a plan if and only if the Turing machine accepts without violating the space bound.

If the simulation of the Turing machine violates the space bound, then $h > e(n)$ and a goal state cannot be reached because no operator will be applicable.

**2-EXPTIME-hardness of PO planning**

- Branches in plans make it possible to simulate alternating TMs that have a computation tree with AND and OR nodes.

- With partial observability we can extend the EXPSPACE TM simulation to $\text{AEXPSPACE} = 2\text{-EXPTIME} = O(2^n^2)$ time simulation.

  **AND node** = nondeterministic operator corresponds to a set of transitions  
  **OR node** = a set of deterministic operators corresponds to a set of transitions (one operator is chosen)