Algorithm for constructing plans without loops

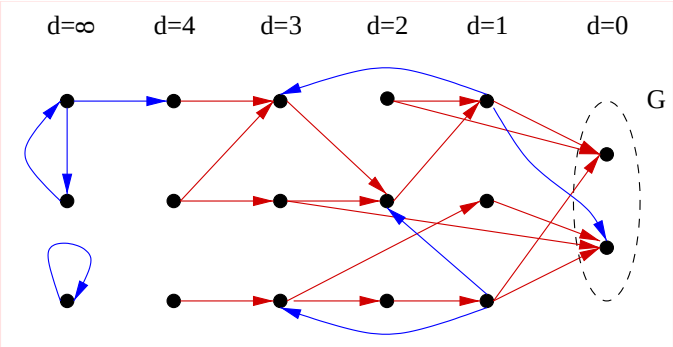
1. Compute distances to goals for all states by strong preimages.
2. Synthesize an acyclic plan in forward direction starting from the initial states.

   Operator selection: for each state, the operator has to decrease the remaining distance to goals by at least 1.

The algorithm: strong distances

PROCEDURE FOplan(I,O,G)
D_0 := G;
i := 0;
WHILE $I \not\subseteq D_i$ *AND* ($i = 0$ *OR* $D_{i-1} \neq D_i$) *DO*
  D_i := $D_{i-1} \cup \bigcup_{o \in O}$ *spreimg*$_o(D_{i-1})$;
  i := $i + 1$;
END

The algorithm: example of distances



The algorithm: plan construction

Initialize the edge labels and node set $N = \emptyset$, and call the recursive procedure FOplanconstruct.

N := $\{0\}$;
$l(j)$:= $\emptyset$ for all j;
cnt := 1;
FOplanconstruct(0,I);

The algorithm: plan construction

PROCEDURE FOplanconstruct(n,S)
FOR EACH $o \in O$
... **see next slide!**
IF $S \neq \emptyset$ *THEN*
  BEGIN (The remaining states are included in G.)
    cnt := cnt$+1$;
    $l(n)$:= $l(n) \cup \{\langle S$,cnt$-1\rangle\}$;
  END
END

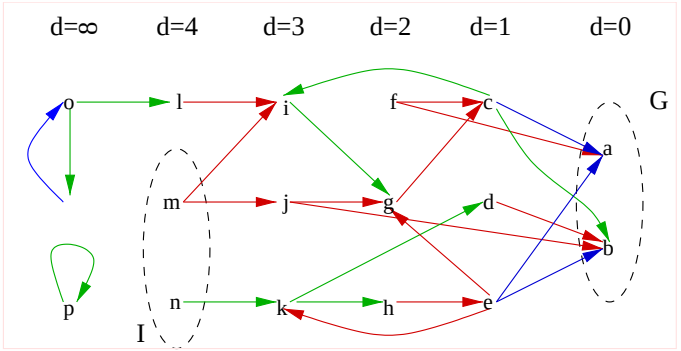
The algorithm: plan construction, cont'd

S' := the maximal subset of S such that progress(o, S');
IF $S' \neq \emptyset$ *THEN*
  BEGIN
    S := $S \backslash S'$;
    cnt := cnt$+2$;
    N := $N \cup \{$cnt-2,cnt-1$\}$;
    $l(n)$:= $l(n) \cup \{\langle S'$,cnt-2$\rangle\}$;
    l(cnt-2) := $\langle o$,cnt-1$\rangle$;
    FOplanconstruct(cnt-1,*img*$_o(S')$);
  END

The algorithm: a progress criterion

PROCEDURE progress(o, S)
FOR j := 1 *TO* i *DO*
  IF *img*$_o(S \cap D_j) \not\subseteq D_{j-1}$ *THEN RETURN* false;
END
RETURN true;

The algorithm: plan construction, example



The algorithm: cont'd

Plan is $\langle \{0, \ldots, 18\}, 0, l \rangle$ where

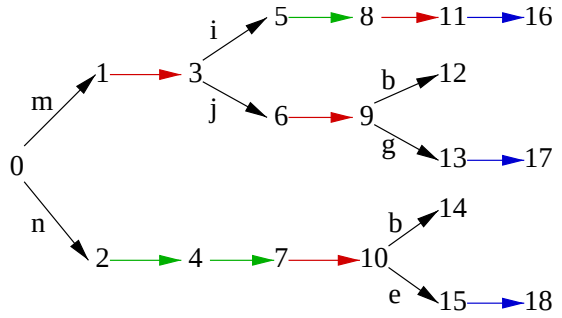
$$
\begin{array}{rclrcl}
l(0) & = & \{\langle m, 1\rangle, \langle n, 2\rangle\} & l(1) & = & \langle \mathsf{R}, 3\rangle \\
l(2) & = & \langle \mathsf{G}, 4\rangle & l(3) & = & \{\langle i, 5\rangle, \langle j, 6\rangle\} \\
l(4) & = & \langle \mathsf{G}, 7\rangle & l(5) & = & \langle \mathsf{G}, 8\rangle \\
l(6) & = & \langle \mathsf{R}, 9\rangle & l(7) & = & \langle \mathsf{R}, 10\rangle \\
l(8) & = & \langle \mathsf{R}, 11\rangle & l(9) & = & \{\langle b, 12\rangle, \langle g, 13\rangle\} \\
l(10) & = & \{\langle b, 14\rangle, \langle e, 15\rangle\} & l(11) & = & \langle \mathsf{B}, 16\rangle \\
l(12) & = & \emptyset & l(13) & = & \langle \mathsf{B}, 17\rangle \\
l(14) & = & \emptyset & l(15) & = & \langle \mathsf{B}, 18\rangle \\
l(16) & = & \emptyset, l(17) = \emptyset & l(18) & = & \emptyset
\end{array}
$$

Memoryless plans

When we have full observability, a simpler definition of plans suffices.

- Plan is a mapping from states to operators.
- In program form: loop inside which one case statement chooses an operator for every state (or chooses termination.)

The algorithm: plan construction, example



Memoryless plans: example

```
1:  CASE
      a,b:          GOTO 5
      d,f,g,h,j,m:  GOTO 2
      c,e:          GOTO 3
      i,k,n:        GOTO 4
2:  RED
    GOTO 1
3:  BLUE
    GOTO 1
4:  GREEN
    GOTO 1
5:
```