

STRIPS PLANNER

Notation: Let a be an action and S a state. By $a(S)$ we denote the state that results from S by applying (executing) a . In STRIPS,

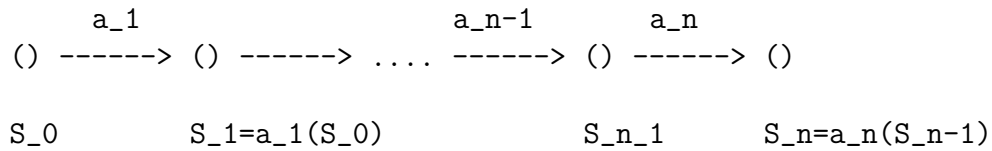
$$a(S) = (S - Eff^-) \cup Eff^+.$$

Definition: Let A be a set of actions, let S_0 be a state, and let $G = [g_1, \dots, g_k]$ be a list of atomic formulas (list of *goals*). A plan for G from S_0 is a sequence

$$a_1, \dots, a_n$$

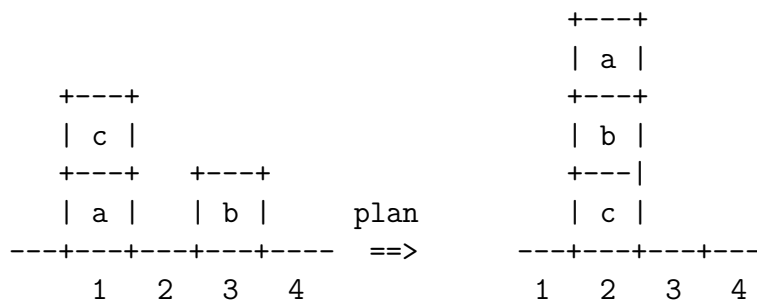
of actions from A such that every goal g_i from G is true in state

$$a_n(a_{n-1}(\dots(a_1(S_0))\dots))$$



Every goal in G is in S_n

Consider the following planning problem



S_final

$$S_0 = [on(a, 1), on(c, a), on(b, 3), clear(c), clear(2), clear(b), clear(4)]$$
$$Goal = [on(c, 2), on(b, c), on(a, b)].$$

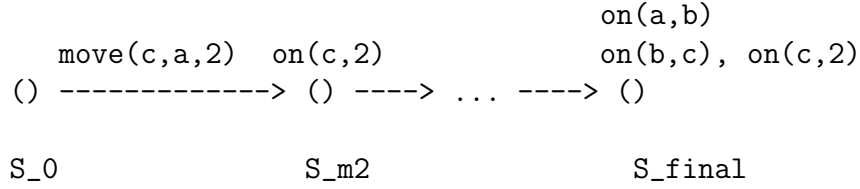
```

preconditions: [on(X, From), clear(X), clear(To)],

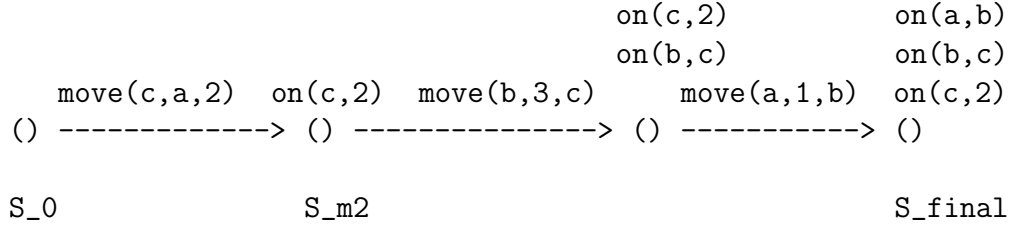
```

$$Eff^-: [on(X, From), clear(To)].$$
$$\begin{array}{ccccccc}
 & & & & & \text{on}(a,b) & \\
 & & & & & \text{on}(b,c), \text{on}(c,2) & \\
 & & \text{on}(c,2) & & & & \\
 () \text{ ----} \rightarrow & \dots \text{ ----} \rightarrow & () \text{ ----} \rightarrow & \dots \text{ ----} \rightarrow & () & & \\
 S_0 & & S_m2 & & S_final & &
 \end{array}$$

1. the only way $on(c, 2)$ can be made true is by executing $move(c, From, 2)$, since Eff^+ will add $on(c, 2)$ to the new state;
2. to execute $move(c, From, 2)$, the preconditions $[on(c, From), clear(c), clear(2)]$ have to be true: $clear(c), clear(2)$ are true since they are in S_0 ; $on(c, From)$ is true in S_0 if $From = a$;
3. executing $move(c, a, 2)$ produces the state $[on(a, 1), on(c, 2), on(b, 3), clear(c), clear(a), clear(b), clear(4)]$; this is our S_{m2} :

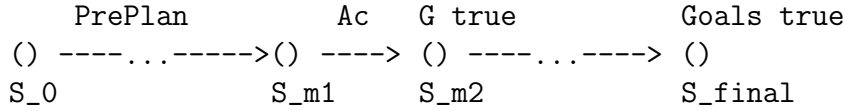


Clearly, by executing $move(b, 3, c)$ we make $on(b, c)$ true and, finally, by executing $move(a, 1, b)$ we make $on(a, b)$ true.



In general: assume that S_0 is some initial state and $Goals$ is a list of goals. Then:

- (1) select a still unsolved goal G from $Goals$,
- (2) find an action Ac that adds G to the current state,
- (3) make Ac executable by deriving a plan (called *PrePlan*, to satisfy all the preconditions of Ac ; let S_{m1} be the state obtained from S_0 by executing *PrePlan*;
- (4) execute Ac in S_{m1} giving the state S_{m2} ;
- (5) solve the remaining goals in $Goals$ by deriving a plan, called *PostPlan*, from S_{m2} .



PROLOG IMPLEMENTATION of MEANS-ENDS PLANNER

Representation of Actions

preconditions: $can(action, Prec_List)$
 $can(move(X, From, To), [on(X, From), clear(X), clear(To)]),$

Eff^+ : $add(action, Eff^+),$
 $add(move(X, From, To), [on(X, To), clear(From)]),$

Eff^- : $del(action, Eff^-),$
 $del(move(X, From, To), [on(X, From), clear(To)]).$

```

plan(State,Goals,[],State):- satisfied(State,Goals).
plan(State,Goals,Plan,FinalState):-
    conc(PrePlan,[Action|PostPlan],Plan),
    selectG(State,Goals,Goal),
    achieves(Action,Goal),
    can(Action,Condition),
    plan(State,Condition,PrePlan,MidState1),
    apply(MidState1,Action,MidState2),
    plan(MidState2,Goals,PostPlan,FinalState).

satisfied(State,[]).
satisfied(State,[Goal|Goals]):-
    member(Goal,State), satisfied(State,Goals).

selectG(State,Goals,Goal):-
    member(Goal,Goals), not(member(Goal,State)).
achieves(Action,Goal):-
    add(Action,Goals),
    member(Goal,Goals).

apply(State,Action,NewState):-
    del(Action,Dellist), deleteS(State,Dellist,State1),!,
    add(Action,AddList),
    conc(AddList,State1,NewState).

```