Planning

Given

- an initial world description
- a description of available actions
- a goal

a plan is a sequence of actions that will achieve the goal.

Example Planning

If you want a plan to achieve Rob holding the key $k1$ and being at $o103$, you can issue the query

$$?carrying(rob, k1, S) \land at(rob, o103, S).$$

This has an answer

$$
\begin{aligned}
S = do(&move(rob, mail, o103),\\
&do(pickup(rob, k1),\\
&\quad do(move(rob, o103, mail),\\
&\quad\quad do(move(rob, o109, o103), init)))).
\end{aligned}
$$

Forward Planner

- Search in the state-space graph, where the nodes represent states and the arcs represent actions.
- Search from initial state to a state that satisfies the goal.
- A complete search strategy (e.g., A^* or iterative deepening) is guaranteed to find a solution.
- Branching factor is the number of legal actions. Path length is the number of actions to achieve the goal.
- You usually can't do backward planning in the state space, as the goal doesn't uniquely specify a state.

Planning as Resolution

- Idea: backward chain on the situation calculus rules or the situation calculus axiomatization of STRIPS.
- A complete search strategy (e.g., A^* or iterative deepening) is guaranteed to find a solution.
- When there is a solution to the query with situation $S = do(A, S_1)$, action A is the last action in the plan.
- You can virtually always use a frame axiom so that the search space is largely unconstrained by the goal.

Goal-directed searching

- Given a goal, you would like to consider only those actions that actually achieve it.
- Example:

$$?carrying(rob, parcel, S) \land in(rob, lab2, S).$$

the last action needed is irrelevant to the left subgoal.

STRIPS Planner

- Divide and conquer: to create a plan to achieve a conjunction of goals, create a plan to achieve one goal, and then create a plan to achieve the rest of the goals.
- To achieve a list of goals:
  - choose one of them to achieve.
  - If it is not already achieved
    - ⋆ choose an action that makes the goal true
    - ⋆ achieve the preconditions of the action
    - ⋆ carry out the action
  - achieve the rest of the goals.

STRIPS Planner Code

% $achieve\_all(Gs, W_1, W_2)$ is true if W_2 is the world resulting
% from achieving every element of the list Gs of goals from
% the world W_1.

$$
\begin{aligned}
& achieve\_all([\,], W_0, W_0). \\
& achieve\_all(Goals, W_0, W_2) \leftarrow \\
& \quad remove(G, Goals, Rem\_Gs) \wedge \\
& \quad achieve(G, W_0, W_1) \wedge \\
& \quad achieve\_all(Rem\_Gs, W_1, W_2).
\end{aligned}
$$

% $achieve(G, W_0, W_1)$ is true if W_1 is the resulting world
% after achieving goal G from the world W_0.

$$
\begin{aligned}
& achieve(G, W, W) \leftarrow \\
& \quad holds(G, W). \\
& achieve(G, W_0, W_1) \leftarrow \\
& \quad clause(G, B) \wedge \\
& \quad achieve\_all(B, W_0, W_1). \\
& achieve(G, W_0, do(Action, W_1)) \leftarrow \\
& \quad achieves(Action, G) \wedge \\
& \quad preconditions(Action, Pre) \wedge \\
& \quad achieve\_all(Pre, W_0, W_1).
\end{aligned}
$$

Undoing Achieved Goals

Example: consider trying to achieve

$$[carrying(rob, parcel), sitting\_at(rob, lab2)]$$

Example: consider trying to achieve

$$[sitting\_at(rob, lab2), carrying(rob, parcel)]$$

- The STRIPS algorithm, as presented, is unsound.
- Achieving one subgoal may undo already achieved subgoals.

Fixing the STRIPS Algorithm

Two ideas to make STRIPS sound:

- Protect subgoals so that, once achieved, until they are needed, they cannot be undone. Let *remove* return different choices.
- Reachieve subgoals that have been undone.
  - Protecting subgoals makes STRIPS incomplete.
  - Reachieving subgoals finds longer plans than necessary.

Does protecting always work?

- Example Suppose the robot can only carry one item at a time. Consider the goal:

$$sitting\_at(rob, lab2) \wedge carrying(rob, parcel)$$

- We cannot consider the subgoals in isolation!

Regression

- Idea: don't solve one subgoal by itself, but keep track of all subgoals that must be achieved.
- Given a set of goals:
  - If they all hold in the initial state, return the empty plan
  - Otherwise, choose an action A that achieves one of the subgoals. This will be the last action in the plan.
  - Determine what must be true immediately before A so that all of the goals will be true immediately after. Recursively solve these new goals.

Regression as Path Finding

- The nodes are sets of goals. Arcs correspond to actions.
- A node labeled with goal set G has a neighbor for each action A that achieves one of the goals in G.
- The neighbor corresponding to action A is the node with the goals G_A that must be true immediately before the action A so that all of the goals in G are true immediately after A. G_A is the **weakest precondition** for action A and goal set G.
- Search can stop when you have a node where all the goals are true in the initial state.

Weakest preconditions

$wp(A, GL, WP)$ is true if WP is the weakest precondition that must occur immediately before action A so every element of goal list GL is true immediately after A.

For the STRIPS representation (with all predicates primitive):

- $wp(A, GL, WP)$ is *false* if any element of GL is on delete list of action A.
- Otherwise WP is
  $$preconds(A) \cup \{G \in GL : G \notin add\_list(A)\}.$$
  where $preconds(A)$ is the list of preconditions of action A and $add\_list(A)$ is the add list of action A.

Weakest Precondition Example

The weakest precondition for

$[sitting\_at(rob, lab2), carrying(rob, parcel)]$

to be true after $move(rob, Pos, lab2)$ is that

$[autonomous(rob),$
$adjacent(Pos, lab2),$
$sitting\_at(rob, Pos),$
$carrying(rob, parcel)]$

is true immediately before the action.

A Regression Planner

% $solve(GL, W)$ is true if every element of goal list GL is true
% in world W.

$solve(GoalSet, init) \leftarrow$
$holdsall(GoalSet, init).$
$solve(GoalSet, do(Action, W)) \leftarrow$
$consistent(GoalSet) \wedge$
$choose\_goal(Goal, GoalSet) \wedge$
$choose\_action(Action, Goal) \wedge$
$wp(Action, GoalSet, NewGoalSet) \wedge$
$solve(NewGoalSet, W).$

Regression Search Space Example