Planning as a Search Problem

- Given a CW–KB representing the initial state, a set of STRIPS or ADL (Action Description Language) operators, and a goal condition we want to achieve (specified either as a conjunction of facts, or as a formula)
- The planning problem is determine a sequence of actions that when applied to the initial CW–KB yield an updated CW–KB which satisfies the goal.

Planning As Search

- This can be treated as a search problem.
  - The initial CW–KB is the initial state.
  - The actions are operators mapping a state (a CW–KB) to a new state (an updated CW–KB).
  - The goal is satisfied by any state (CW–KB) that satisfies the goal.

Example.

```
C A B
move(b,c)
C A
move(c,b)
A B
move(c,table)
A B C
move(a,b)
B A C
```
Problems

- Search tree is generally quite large
  - randomly reconfiguring 9 blocks takes thousands of CPU seconds.
- The representation suggests some structure.
  Each action only affects a small set of facts, actions depend on each other via their preconditions.
- Planning algorithms are designed to take advantage of the special nature of the representation.

Planning

- We will look at 1 technique
- Relaxed Plan heuristics used with heuristic search.

Reachability Analysis.

- The idea is to consider what happens if we ignore the delete lists of actions.
- This yields a “relaxed problem” that can produce a useful heuristic estimate.

Reachability Analysis

- In the relaxed problem actions add new facts, but never delete facts.
- Then we can do reachability analysis, which is much simpler than searching for a solution.
Reachability

- We start with the initial state $S_0$.
- We alternate between state and action layers.
- We find all actions whose preconditions are contained in $S_0$. These actions comprise the first action layer $A_0$.
- The next state layer consists of all of $S_0$ as well as the adds of all of the actions in $A_0$.
- In general
  - $A_i$ is the set of actions whose preconditions are contained in $S_i$.
  - $S_{i+1}$ is $S_i$ union the add lists of all of the actions in $A_i$.

STRIPS Blocks World Operators.

- pickup($X$)
  - Pre: $\{\text{clear}(X), \text{ontable}(X), \text{handempty}\}$
  - Add: $\{\text{holding}(X)\}$
  - Del: $\{\text{clear}(X), \text{ontable}(X), \text{handempty}\}$

- putdown($X$)
  - Pre: $\{\text{holding}(X)\}$
  - Add: $\{\text{clear}(X), \text{ontable}(X), \text{handempty}\}$
  - Del: $\{\text{holding}(X)\}$

- unstack($X,Y$)
  - Pre: $\{\text{clear}(X), \text{on}(X,Y), \text{handempty}\}$
  - Add: $\{\text{holding}(X), \text{clear}(Y)\}$
  - Del: $\{\text{clear}(X), \text{on}(X,Y), \text{handempty}\}$

- stack($X,Y$)
  - Pre: $\{\text{holding}(X), \text{clear}(Y)\}$
  - Add: $\{\text{on}(X,Y), \text{handempty}, \text{clear}(X)\}$
  - Del: $\{\text{holding}(X), \text{clear}(Y)\}$

Example

- $S_0$: $\text{on}(a,b), \text{on}(c,d), \text{ontable}(c), \text{ontable}(d), \text{clear}(a), \text{clear}(d), \text{handempty}$
- $A_0$: $\text{unstack}(a,b), \text{pickup}(d)$
- $S_1$: $\text{on}(a,b), \text{on}(a,c), \text{on}(b,c), \text{ontable}(c), \text{ontable}(d), \text{clear}(a), \text{clear}(d), \text{handempty}, \text{holding}(a), \text{holding}(d)$

This is not a state!
Example

- on(a, b),
- on(b, c),
- ontable(c),
- ontable(d),
- clear(a),
- clear(d),
- handempty,
- holding(a),
- clear(b),
- holding(d)

\[ S_1 \]

\[ A_1 \]

Reachability

- We continue until the goal G is contained in the state layer, or until the state layer no longer changes.
- Intuitively, the actions at level \( A_i \) are the actions that could be executed at the i-th step of some plan, and the facts in level \( S_i \) are the facts that could be made true after some i-1 step plan.
- Some of the actions/facts have this property. But not all!

Heuristics from Reachability Analysis

Grow the levels until the goal is contained in the final state level \( S[K] \).

- If the state level stops changing and the goal is not present. The goal is unachievable. (The goal is a set of positive facts, and in STRIPS all preconditions are positive facts).

- Now do the following
Heuristics from Reachability Analysis

CountActions(G,S₀):
/* Compute the number of actions contained in a relaxed plan achieving the goal. */

- Split G into facts in S₀⁻¹ and elements in S₀ only. These sets are the previously achieved and just achieved parts of G.
- Find a minimal set of actions A whose add-effects cover the just achieved part of G. (The set contains no redundant actions, but it might not be the minimum sized set.)
- Replace the just achieved part of G with the preconditions of A, call this updated G, NewG.
- Now return CountAction(NewG,S₀⁻¹) + number of actions needed to cover the just achieved part of G.

Example

CountActs(G,S₁)

G₀ = {f₁, f₂, f₃} //already in S₀
A₀ = {[f₁]a₁,[f₂]a₂,[f₃]}
S₁ = {f₁,f₂,f₃,f₄,f₅} //New in S₁
A₁ = {[f₂]a₂,[f₃]a₃,[f₄]a₃,[f₅]}
S₂ = {f₁,f₂,f₃,f₄,f₅,f₆}

G₁ = {f₅,f₁,f₂,f₄,f₅,f₆}

We split G into Gₚ and Gₙ:

So, in total CountActs(G,S₂)=1+2=3

Using the Heuristic

1. To use CountActions as a heuristic, we build a layered structure from a state S that reaches the goal.
2. Then we CountActions to see how many actions are required in a relaxed plan.
3. We use this count as our heuristic estimate of the distance of S to the goal.
4. This heuristic tends to work better as a best-first search, i.e., when the cost of getting to the current state is ignored.
Admissibility

- An optimal length plan in the relaxed problem (actions have no deletes) will be a lower bound on the optimal length of a plan in the real problem.
- However, CountActions does NOT compute the length of the optimal relaxed plan.
- The choice of which action set to use to achieve $G_p$ (“just achieved part of $G$”) is not necessarily optimal.
- In fact it is NP–Hard to compute the optimal length plan even in the relaxed plan space.
- So CountActions will not be admissible.

Empirically

- However, empirically refinements of CountActions performs very well on a number of sample planning domains.

CWA

- “Classical Planning”. No incomplete or uncertain knowledge.
- Use the “Closed World Assumption” in our knowledge representation and reasoning.
  - The Knowledge base used to represent a state of the world is a list of positive ground atomic facts.
  - CWA is the assumption that
    a) if a ground atomic fact is not in our list of “known” facts, its negation must be true.
    b) the constants mentioned in KB are all the domain objects.

State of the Art in Planning

- There are annual AI planning systems competitions (e.g. IPC)
- The Planning Domain Definition Language (PDDL) is a standard in the area; has sublanguages with varying expressiveness
- Several state of the art (classical) planning systems perform well on large real world problems
Current Research in Planning

- STRIPS/classical planning makes very strong assumptions: complete information, deterministic actions, static single-agent world
- Much recent work in planning addresses more general forms of planning that avoid such assumptions
- In such cases, a solution may be a branching plan (branching on observations/action outcomes), finite state automaton, or a policy
- Hierarchical planning/abstraction is useful to address large real world problems