The Need for Special Purpose Algorithms

So…We have a formalism for expressing goals and plans and we can use resolution theorem proving to find plans.

Problems:

- Frame problem
- Time to find plan can be exponential
- Logical inference is semi-decidable
- Resulting plan could have many irrelevant steps

We'll need to:

- Restrict language
- Use a special purpose algorithm called a planner

The STRIPS Language

States and Goals: Conjunctions of positive, function-free literals. No variables (i.e. "ground").

Have (Milk) ∧ Have (Bananas) ∧ Have (Drill) ∧ At (Home)

Closed World Assumption: any conditions that are not mentioned in a state are assumed false.

Actions:

- **Preconditions:** conjunction of positive, function-free literals that must be true before the operator can be applied.
- **Effects:** conjunction of function-free literals; *add* list and *delete* list.

STRIPS Assumption

Assumption: Every literal not mentioned in the effect remains unchanged in the resulting state when the action is executed.

➔ Avoids the representational frame problem.

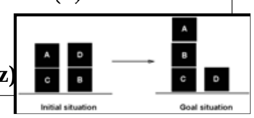
Solution for the planning problem:
An action sequence that, when executed in the initial state, results in a state that satisfies the goal.

STRIPS Actions

Move block *x* from block *y* to block *z* (Put(x,y,z))
Preconds: $On(x,y) \wedge Block(x) \wedge Block(z)$ $\wedge Clear(x) \wedge Clear(z)$
Effects: **Add: On(x,z), Clear(y)**
Delete: On(x,y), Clear(z)

Move block *x* from block *y* to Table (PoT(x,y))
Preconds: $On(x,y) \wedge Block(x) \wedge Block(y) \wedge Clear(x)$
Effects: **Add: On(x,Table), Clear(y)**
Delete: On(x,y)

Move block *x* from Table to block *z* (TtB(x,z))
Preconds: $On(x,Table) \wedge Block(x) \wedge Block(z)$ $\wedge Clear(x) \wedge Clear(z)$
Effects: **Add: On(x,z)**
Delete: On(x,Table), Clear(z)



Plan by Searching for a Satisfactory Sequence of Actions

Planning via State-Space Search

- **Progression planner** searches forward from the initial situation to the goal situation.
- **Regression planner** search backwards from the goal state to the initial state.
- **Heuristics:**
  - derive a relaxed problem
  - employ the subgoal independence assumption.

Searching Plan Space

Planning via Plan-Space Search:

- Alternative is to search through the space of *plans* rather than the original state space.
- Start with simple, incomplete partial plan; expand until complete.
- **Operators:** add a step, impose an ordering on existing steps, instantiate a previously unbound variable.
- **Refinement Operators** take a partial plan and add constraints
- **Modification Operators** are anything that is not a refinement operator; take an incorrect plan and debug it.

Representation for Plans

Goal: $RightShoeOn \wedge LeftShoeOn$

Initial state: λ

Operators:

Action	Preconds	Effect
RightShoe	RightSockOn	RightShoeOn
RightSock	λ	RightSockOn
LeftShoe	LeftSockOn	LeftShoeOn
LeftSock	λ	LeftSockOn

Partial Plans

Partial Plan: RightShoe LeftShoe

Partial order planner – can represent plans in which some steps are ordered and others are not.

Total order planner considers a plan a simple list of steps

A linearization of a plan P is a totally ordered plan that is derived from a plan P by adding ordering constraints.

Partial Plan for Shoes and Socks