Planning

A planning agent will construct plans to achieve its goals, and then execute them.

Analyze a situation in which it finds itself and develop a strategy for achieving the agent’s goal.

Achieving a goal requires finding a sequence of actions that can be expected to have the desired outcome.

Problem Solving

Representation of actions – actions generate successor states

Representation of states – all state representations are complete

Representation of goals – contained in goal test and heuristic function

Representation of plans – unbroken sequence of actions leading from initial to goal state.
GOAL: Get a quart of milk and a bunch of bananas and a variable-speed cord-less drill.

Planning Versus Problem Solving

(1) Open up the representation of states, goals and actions.

- States and goals represented by sets of sentences –
  \( \text{Have(Milk)} \)

- Actions represented by rules whose consequent specifies an action and its effect: \( \text{Buy(Milk)} \rightarrow \text{Have(Milk)} \)

This allows the planner to make direct connections between states and actions.
Planning Versus Problem Solving

(2) Planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state.

- No connection between the order of planning and the order of execution.
- Representation of states as sets of logical sentences makes this freedom possible.

Slide CS472 – Planning 5

Planning Versus Problem Solving

(3) Most parts of the world are independent of most other parts.

- Can solve $\text{Have(Milk)} \land \text{Have(Bananas)} \land \text{Have(Drill)}$ using divide-and-conquer strategy.
- Can re-use subplans (go to supermarket)

Slide CS472 – Planning 6
Planning as a Logical Inference Problem

Initial situation

Goal situation

Axioms:
on(a,c), on(c,table), on(d,b), on(b,table), clear(a), clear(d)
plus rules for moving things around...
Prove: on (a,b) ∧ on(b,c)

Planning as Deduction: Situation Calculus

In first-order logic, once a statement is shown to be true, it remains true forever.

Situation calculus: way to describe change in first-order logic. Each assertion specifies the situation in which the assertion is true.

on(A, C)      on(A, C, S₀)
on(C, Table)  ∃s₁on(C, Table, s₁)

Actions: place-on-table
∀s∀x[¬on(x, Table, s) → on(x, Table, place(x, s))]
∀s∀y∀z[on(y, z, s) ∧ ¬equal(z, Table) → ¬on(y, Table, s)]
The Frame Problem and Its Relatives

Actions don’t specify what happens to objects not involved in the action, but the logic framework requires that information.

\[ \forall s \forall x [\neg on(x, \text{Table}, s) \rightarrow on(x, \text{Table}, place(x, s))] \]

Frame axioms: Inform the system about preserved relations.

\[ \forall s \forall x \forall y \forall z [on(x, y, s) \land \neg equal(x, z) \rightarrow on(x, y, place(z, s))] \]

representational frame problem: proliferation of frame axioms

inferential frame problem: have to carry each property through all intervening situations during problem-solving, even if the property remains unchanged throughout

qualification problem: difficult, in the real world, to define the circumstances under which a given action is guaranteed to work

ramification problem: proliferation of implicit consequences of actions.
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

Slide CS472 – Planning 11

The STRIPS Language

States and Goals: Conjunctions of function-free literals.
Have (Milk) \& Have (Bananas) \& Have (Drill)
\& At (Home)

Operators:

action description: name for action; command to environment.

preconditions: conjunction of atoms that must be true before the operator can be applied.

effects: add list and delete list
STRIPS Operators

Op 1: Move block $x$ from block $y$ to block $z$

preconds: $on(x, y) \land clear(x) \land clear(z)$

effects: Add: on($x$, $z$), clear($y$)

Delete: on($x$, $y$), clear($z$)

Op 2: Move block $x$ from block $y$ to Table

preconds: $on(x, y) \land clear(x)$

effects: Add: on($x$, Table), clear($y$)

Delete: on($x$, $y$)

Op 3: Move block $x$ from Table to block $z$

preconds: $on(x, \text{Table}) \land clear(x) \land clear(z)$

effects: Add: on($x$, $z$)

Delete: on($x$, Table), clear($z$)
Plan by Searching for a Satisfactory Sequence of Operators

**situation space planner** searches through space of possible situations

**progression planner** searches forward from the initial situation to the goal situation

**regression planner** search backwards from the goal state to the initial state

---

Searching Plan Space

Alternative is to search through the space of *plans* rather than the space of *situations*.

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 ∧ LeftShoeOn

Initial state: λ

Operators:

<table>
<thead>
<tr>
<th>Action</th>
<th>Preconds</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>RightShoe</td>
<td>RightSockOn</td>
<td>RightShoeOn</td>
</tr>
<tr>
<td>RightSock</td>
<td>λ</td>
<td>RightSockOn</td>
</tr>
<tr>
<td>LeftShoe</td>
<td>LeftSockOn</td>
<td>LeftShoeOn</td>
</tr>
<tr>
<td>LeftSock</td>
<td>λ</td>
<td>LeftSockOn</td>
</tr>
</tbody>
</table>

Slide CS472 – Planning 17

Partial Plans

Partial Plan: RightShoe LeftShoe

Principle of **Least Commitment** says to only make choices about things that you currently care about.

**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 P** is a totally ordered plan that is derived from a plan P by adding ordering constraints.

Slide CS472 – Planning 18
Defer Variable Binding

Planners must commit to bindings for variables

Example: Goal: Have(Milk) Action: Buy(item, store)

**Principle of Least Commitment:** Only make choices about things that you care about, leaving other details to be worked out later.

Buy(Milk, K-MART) versus Buy(Milk, store)

**Fully instantiated plan:** every variable is bound to a constant.

---

Definition of a Plan

- A set of plan steps (operators).
- A set of step ordering constraints of the form $S_i \prec S_j$
- A set of variable binding constraints
- A set of causal links, written as $S_i \xrightarrow{c} S_j$
Initial Plan for Shoes and Socks

Initial plan: $Start < Finish$

Slide CS472 – Planning 21

Partial Plan for Shoes and Socks

Partial Order Plan:

Total Order Plans:

Slide CS472 – Planning 22
Planner Output

A solution is a complete, consistent plan.

A complete plan: every precondition of every step is achieved by some other step.

A consistent plan: there are no contradictions in the ordering or binding constraints. Contradiction occurs when both \( S_i \prec S_j \) and \( S_j \prec S_i \).

<table>
<thead>
<tr>
<th>Action</th>
<th>PreCond</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go( (\text{there}) )</td>
<td>At( (\text{here}) )</td>
<td>( \text{At(there) } \land \neg\text{At(here)} )</td>
</tr>
<tr>
<td>Buy( (x) )</td>
<td>At( (\text{store}) ) \land Sells( (\text{store}, x) )</td>
<td>Have( (x) )</td>
</tr>
</tbody>
</table>

Slide CS472 – Planning 23

Slide CS472 – Planning 24
POP Example: Initial Plan

Start

At(Home) Sells(SM,Banana) Sells(SM,Milk) Sells(HWS,Drill)
Have(Drill) Have(Milk) Have(Banana) At(Home)

Finish

Slide CS472 – Planning 25

A Partial Plan I

Start

At(s), Sells(s,Drill) At(s), Sells(s,Milk) At(s), Sells(s,Banana)
Buy(Drill) Buy(Milk) Buy(Bananas)
Have(Drill), Have(Milk), Have(Banana) At(Home)

Finish

Start

At(HWS), Sells(HWS,Drill) At(SM), Sells(SM,Milk) At(SM), Sells(SM,Banana)
Buy(Drill) Buy(Milk) Buy(Bananas)
Have(Drill), Have(Milk), Have(Banana) At(Home)

Finish

Slide CS472 – Planning 26
Protecting Causal Links

(a) \( S_1 \rightarrow S_3 \rightarrow \neg c \rightarrow c \rightarrow S_2 \)

(b) \( S_1 \rightarrow S_3 \rightarrow \neg c \rightarrow c \rightarrow S_2 \)

(c) \( S_1 \rightarrow \neg c \rightarrow c \rightarrow S_2 \)

Slide CS472 – Planning 29

A Partial Plan III

Start

\( \text{At} (\text{HWS}) \)

\( \text{Go} (\text{HWS}) \)

\( \text{At} (\text{SM}) \)

\( \text{Go} (\text{SM}) \)

\( \text{At} (\text{HWS}) \)

\( \text{At} (\text{SM}) \)

Buy(Drill)

Buy(Milk)

Buy(Bananas)

Go(Home)

Finish

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

Slide CS472 – Planning 30
### Achieving At(Home)

<table>
<thead>
<tr>
<th>Candidate link</th>
<th>Threats</th>
</tr>
</thead>
<tbody>
<tr>
<td>At(x) to initial state</td>
<td>Go(HWS), Go(SM)</td>
</tr>
<tr>
<td>At(x) to Go(HWS)</td>
<td>Go(SM)</td>
</tr>
<tr>
<td>At(x) to Go(SM)</td>
<td>At(SM) preconds of Buy(Milk), Buy(Bananas)</td>
</tr>
</tbody>
</table>

**Solution:** Link At(x) to Go(SM), but order Go(Home) to come after Buy(Bananas) and Buy(Milk).

---

### A final Plan

![Diagram showing a final plan with nodes for At(Home), Go(HWS), Buy(Milk), etc., and arrows indicating transitions between states.]

---

**Slide CS472 – Planning 31**

**Slide CS472 – Planning 32**
function POP(initial, goal, operators) returns plan
    plan = MAKE-MINIMAL-PLAN(initial, goal)
    loop do
        if SOLUTION?(plan) then return plan
        \( S_{\text{need}} \leftarrow \text{SELECT-SUBGOAL}(plan) \)
        CHOOSE-OPERATOR(plan, operators, \( S_{\text{need}} \))
        RESOLVE-THREATS(plan)
    end

function SELECT-SUBGOAL(plan) returns \( S_{\text{need}}, c \)
    pick a plan step \( S_{\text{need}} \) from STEPS(plan)
    with a precondition \( c \) that has not been achieved
    return \( S_{\text{need}}, c \)

procedure CHOOSE-OPERATOR(plan, operators, \( S_{\text{need}}, c \))
    choose a step \( S_{\text{add}} \) from operators or STEPS(plan) that has \( c \) as an effect
    if there is no such step then fail
    add the causal link \( S_{\text{add}} \rightarrow S_{\text{need}} \) to LINKS(plan)
    add the ordering constraint \( S_{\text{add}} < S_{\text{need}} \) to ORDERINGS(plan)
    if \( S_{\text{add}} \) is a newly added step from operators then
        add \( S_{\text{add}} \) to STEPS(plan)
        add Start \( S_{\text{add}} \leftarrow \) Finish to ORDERINGS(plan)

procedure RESOLVE-THREATS(plan)
    for each \( S_{\text{threat}} \) that threatens a link \( S_i \rightarrow S_j \) in LINKS(plan) do
        choose either
        Promotion: Add \( S_i < S_j \) to ORDERINGS(plan)
        Demotion: Add \( S_i < S_j \) to ORDERINGS(plan)
        if not CONSISTENT(plan) then fail
    end

Slide CS472 – Planning 33

Strengths of Partial-Order Planning Algorithm

- Takes a huge state space problem and solves in only a few steps.
- Least commitment strategy means that search only occurs in places where sub-plans interact.
- Causal links allow planner to recognize when to abandon a doomed plan without wasting time exploring irrelevant parts of the plan.
Practical Planners

STRIPS approach is insufficient for many practical planning problems. Can’t express:

resources: Operators should incorporate resource consumption and generation. Planners have to handle constraints on resources efficiently.

time: Real-world planners need a better model of time.

hierarchical plans: Need the ability to specify plans at varying levels of detail.

Also need to incorporate heuristics for guiding search.

Slide CS472 – Planning 35

Spacecraft Assembly, integration and verification (AIV)

- OPTIMUM-AIV used by the European Space Agency to AIV spacecraft.

- Generates plans and monitors their execution – ability to re-plan is the principle objective.

- Uses O-Plan architecture – like partial-order planner, but can represent time, resources and hierarchical plans. Accepts heuristics for guiding search and records its reasons for each choice.

Slide CS472 – Planning 36
Scheduling for Space Missions

- Planners have been used by the ground teams for the Hubble space telescope and for the Voyager, Uosat-II and ERS-1.

- Goal: coordinate the observational equipment, signal transmitters and altitude and velocity-control mechanism in order to maximize the value of the information gained from observations while obeying resource constraints on time and energy.

Planning in a Hierarchy of Abstraction Spaces

- Ignore the detail of the full problem.

- Ideally work done at one level is not disturbed by the next level.

- If this assumption is true and if level $i$’s plan creates subproblems of equal size in level $i + 1$, then exponential speedup ensues.
Hierarchical Decomposition

**hierarchical decomposition**: an _abstract operator_ can be decomposed into a group of steps that forms a plan that implements the operator.

Requires two extensions to existing planner:

1. Extend STRIPS language to allow for nonprimitive operators.

2. Modify algorithm to allow the replacement of a nonprimitive operator with its decomposition.
Hierarchical decomposition of *Build House* operator

Rules for Decomposition

A plan $p$ correctly implements an operator $o$ if it is a complete and consistent plan for the problem of achieving the effects of $o$ given the preconditions of $o$:

1. $p$ must be consistent.
2. Every effect of $o$ must be asserted by at least one step of $p$ (and not denied by some other, later step of $p$).
3. Every precondition of the steps in $p$ must be achieved by a step in $p$ or be one of the preconditions of $o$. 
Adapting Previously Generated Plans

1. Given a plan, how can you index it so you can retrieve it for future situations?

2. Given a problem to solve, how can you retrieve the most similar stored plan?

3. How can you adapt the retrieved plan for the new situation?

Slide CS472 – Planning 43