Planning

Failing to plan is planning to fail!

Introduction to Planning

- Plan: a sequence of steps to achieve a goal.
- Problem solving agent knows: actions, states, goals and plans.
- Planning is a special case of problem solving: reach a state satisfying the requirements from the current state using available actions.

Major Agent Types

Agents with Goals

A Planning Agent has the ability to generate a goal, to achieve and construct a plan from the current state. It then executes the plan until the plan is finished and then begins again with a new goal.

From Problem Solving to Planning

A Simple Problem:
“Get a quart of milk and a bunch of bananas and a variable speed cordless drill”

For a problem solving exercise we need to specify:
- Initial State: the agent is at home without any objects that it wanted.
- Operator Set: everything the agent can do.
- Heuristic function: the # of things that have not yet been acquired.

From Problem Solving to Planning

“For many problems the notion of planning is the same as a search for a solution.”
- Example: 8-puzzle
- Other problem domains make the distinction between search and planning clear:
  - those in which the universe is not predictable
  - when steps cannot be undone (can’t backtrack).
Situation Space and Plan Space

A path through the shopping world from the initial state to the goal state constitutes a plan for the shopping problem.

- Start at the initial state and apply operators one at a time until we reach a state that includes all the literals in the goal.
- Use search methods, which would be called a situation space planner because it searches through the space of possible situations. It would also be called a progression planner because it searches forward from the initial to the goal situation.

But the problem with this is the high branching factor and therefore the huge size of the search space.

Assumptions of the “Standard” AI Planning Paradigm

- There is a single causal agent and this agent is the planner.
- The planner is given a well-defined goal which remains fixed over the course of planning.
- The planner is assumed to have functionally complete and accurate knowledge of the starting situation.
- The planner is assumed to possess the knowledge required to accurately model the world.
- The planner is assumed to possess the resources (time and memory) required to use this model to reason about the possible worlds associated with different courses of action that might be pursued.

STRIPS - Linear Planner

- First planner developed by SRI, stands for STanford Research Institute Problem Solver.
- In STRIPS notation, a model of the world is just a list of variables free atomic propositions that hold in the world.
- Operators involving variables are called operator schemas.
- The following expresses an initial state in the block world: `<on(a,t), on(b,a), clear(b), on(c,t), clear(c)>`
- It is assumed that anything not mentioned in the description of the initial state of the world is false.

STRIPS

- The description of the goal state is again a list of atomic proposition where all variables are interpreted existentially.
- The goal state of plan, for example, will be given by such a description (if we want an apple we usually do not refer to a particular apple). An example goal state in the block world is: `<on(X,c), on(c,l)>` This means that some block should be on c, which is itself directly on the table.

Examples of Planning Systems

- Spacecraft assembly, integration and verification
- Job shop scheduling
- Space mission scheduling
- Building construction
- Operations on a flight deck of an aircraft carrier
- For demos: blocks world
**Blocks World**

- **PickUp**(X)
  - X on table, hand empty, X free
- **PutDown**(X)
  - X in hand
- **Stack**(X,Y)
  - X in hand, y free
- **Unstack**(X,Y)
  - X free, X on Y, hand free

---

**STRIPS Example - shopping**

1. **Initial State**: a logical sentence of a situation S_i.
   \[ \text{At(Home,}S_i\text{)} \land \neg \text{Have(Milk,}S_i\text{)} \land \neg \text{Have(Bananas,}S_i\text{)} \land \neg \text{Have(Drill,}S_i\text{)} \]
2. **Goal State**: a logical query asking for a suitable situation.
   \[ \text{At(Home,}S_o\text{)} \land \text{Have(Milk,}S_o\text{)} \land \text{Have(Bananas,}S_o\text{)} \land \text{Have(Drill,}S_o\text{)} \]
3. **Operators**: a set of descriptions of actions.
   \[
   \text{Buy(Milk)} \quad \forall \ a, s \quad \text{Have(Milk, Result(a, s))} \Leftrightarrow [(a = \text{Buy(Milk)} \land \text{At(supermarket,}s\text{)}} \lor \text{Have(Milk,}s\text{)} \land \neg a = \text{Drop(Milk)}] \]

---

**STRIPS Example**

**Example of STRIPS Planning**

- Item R1: a logical sentence of a situation S_i.
  \[
  \text{R1: Have(Milk)} \land \text{Have(Bananas)} \land \text{Have(Drill)} \land \text{At(Home)}
  \]
- Item R2: a logical sentence of a situation S_o.
  \[
  \text{R2: Have(Milk)} \land \text{Have(Bananas)} \land \text{Have(Drill)} \land \text{At(Home)}
  \]

---

**STRIPS Example - shopping**

- **Tidily arranged actions descriptions, restricted language**
  - **ACTION**: \text{Buy}(x)
  - **Precondition**: \( \text{At}(p) \land \text{Sells}(p, z) \)
  - **Effect**: \( \text{Have}(x) \)
  - [Note: this abstracts away many important details!]

- **Restricted language**: efficient algorithm
  - Precondition: conjunction of positive literals
  - Effect: conjunction of literals

- \[
  \text{ASeq: Sell(p, x)}
  \]

- \[
  \text{Buy(x)}
  \]

- \[
  \text{Have(x)}
  \]
State Space Searching

Progression Planners
- Search **top-down** from initial state to the goal state.
- This algorithm will build a path from the initial state to the goal.
- The algorithm also **keep a record** of the plan it has built at any stage to the current state.

Regression Planners
- Search **bottom-up** from the goal state to the initial state.
- This algorithm builds a path from the goal to the initial state.
- The algorithm also **keep a record** of the plan it has built at any stage to the current state.

Plan Space Searching
- An alternative way of viewing the planning problem is to see it as a search through possible plans.
- The main motivation for plan space searching is to avoid back-tracking by looking at the goals in an order different from execution order.
- Search space consisting of states of the world are linked by actions.

Plan Space Searching (cont)
- Plan space is a collection of partially specified plans linked by operators that refine a plan into a more detailed one.
- The initial plan, is some unspecified actions that takes the initial state to the goal state.
- The goal will be fully specified plan (or plans) that performs the desired function.

Partial Ordered Planning
Introduction
- A partially ordered plan is a general representation of plans.
- Idea:
  - Working parallel on several sub-goals.
  - Ordering of goals based on interactions.
- Underlying assumption:
  - Not many interactions.
  - Partially ordered plan = directed graph (AND).

Partial Ordered Planning (cont)
An Example

---

Spring 2002
Partial ordered planners

A plan is complete iff every precondition is achieved.

A precondition is achieved iff it is the effect of an earlier step and no possibly intervening step undoes it.

Partial Order Plan

Plan: a data structure consisting of four components:

- A set of plan steps. Each step is one of the operators for the problem.
- A set of step ordering constraints, each in the form $S_i < S_j$ ($S_i$ must occur before $S_j$, but not necessarily immediately before).
- A set of variable binding constraints, each of the form $v = x$, where $v$ is a variable in some step, and $x$ is either a constant or another variable.
- A set of causal links, written as $S_i \rightarrow S_j$ and read as $S_i$ achieves $c$ for $S_j$. They serve to record the purposes of steps in the plan: here a purpose of $S_i$ is to achieve the precondition $c$ of $S_j$.

Partial Order Plan – shopping planner

1. Initial State: a logical sentence of a situation $S_0$.
   \[ \text{At(Home,}S_0\text{)} /\neg\text{Have(Milk,}S_0\text{)} /\neg\text{Have(Bananas,}S_0\text{)} /\neg\text{Have(Drill,}S_0\text{)} \]

2. Goal State: a logical query asking for a suitable situation.
   \[ \text{At(Home,}S_S\text{)} /\text{Have(Milk,}S_S\text{)} /\text{Have(Bananas,}S_S\text{)} /\text{Have(Drill,}S_S\text{)} \]

   \[ \text{Buy(Milk) action} \]
   \[ \forall a,s \text{ Have(Milk,Result(a,s))} \leftrightarrow [(a = \text{Buy(Milk)} /\text{At(supermarket,}s\text{)} /\neg\text{Have(Milk,}s\text{)} /a \neq \text{Drop(Milk)})] \]
Partial Order Plan – shopping planner

Causal link between Buy(Drill) and Finish means it was added to achieve the Finish precondition Have(Drill).

If a new step comes that deletes Have(Drill) condition the planner will make sure it won’t go between Buy(Drill) and Finish.

A clashing is a potentially interesting step that destroys the condition achieved by a causal link. E.g., Go(Home) clashes At(Home):

- **Definition:** put before Go(HWS)
- **Precondition:** put after Buy(Drill)
Shortcomings of AI Planning in General

- Not every actions can be described with STRIPS-like operators:
  - money transfer: new balance is a function of the old
  - alternative post-conditions
    - PAINTBLACK(x)
      - precondition: x is white
      - (Why not blue? to know what to delete!)

- No complete knowledge about the world.

Shortcomings of AI Planning in General (cont)

- The world is not stable
  - Re-planning must be supported
- Goals are not clearly defined
- Nobody plans the solution of everyday tasks
- Humans learn