What is planning?

- “A plan is any hierarchical process in the organism that can control the order in which a sequence of operations is to be performed.” [Miller, Galanter and Pribram 1960]
- “Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans.” [Russell and Norvig]

Example: Blocksworld

Moving blocks around a table

Example: Robot Tasks

Negotiating space and performing actions to achieve goals

Background: Situation Calculus

First order logic for describing state of the world over time as a sequence of situations

Properties: Statements dependent on time include an argument to indicate their situation.

\[ \text{At(Agent,}[1,1],S0) \land \text{At(Agent,}[1,2],S1) \]

means that the agent was at position [1,1] in situation S0 and at position [1,2] at situation S1

Successor-State Axioms combine

Effect Axioms indicate how the world changes
Frame Axioms describe how the world stays the same.
Issues in Planning I

The Frame Problem
How do we represent the many things that stay the same over time?
- Representation
- Inference

(Graphics from "How to Make Computer Animation"
http://www.combat-fishing.com/animationspace/animationhowto.html)

Issues in Planning II

The Qualification Problem
How do we figure out which actions will succeed and why?
The Ramification Problem
How do we represent all reasonable changes made when an action is taken in some environment?

Approaches to Planning

- Classical/Deliberative
  - Graph-based
  - Heuristic State Space Search
  - Propositional logic, Planning as satisfiability
  - Partial Order (Non linear, Least Commitment)
  - Hierarchical Decomposition (HTN)
- Partially Observable Markov Decision Processes (POMDP)
- Reactive

Classical Planning Assumptions

Problem Description
- An initial state,
- A goal state,
- Knowledge about actions and their effects

Complete Knowledge
Planner includes all knowledge necessary to reason about its world (about itself, actions and the state of the world).
Classical Planning Assumptions II

Deterministic Effects
- Nothing is left to chance.

Single Agent of Change

Atomic and Discrete Time
- World works in lockstep.
- No interruptions to actions.
- One action executed at a time.

Dimensions of Approaches:
Search Space Representation

State Space represents how the world changes as actions are executed (each node is a state in the world at some time).

Plan Space represents how the plan changes as actions are added or moved in the plan (each node is a partial plan).

State Space for Blocks World

Initial State in Plan Space

Plan
Steps:
S1: Op (Action: Start),
S2: Op (Action: Finish,
Precond: On(A,B) ^ On(B,Table)}
Orderings: { S1 < S2 }
Bindings: {},
Links: {}
Dimensions of Approaches: Linearity and Total Order

**Linearity Assumption:** solutions to subgoals can be connected together

**Total Order Planner:** creates plan by adding actions in the sequence in which they will execute

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal independence</td>
<td>Unrealistic</td>
</tr>
<tr>
<td>Fast constrained search space</td>
<td>May miss or find suboptimal solutions</td>
</tr>
</tbody>
</table>

Infamous Problem: Sussman’s Anomaly

Planning Example – Towers of Hanoi

- Why will exhaustive search choke on this?
- Why will random search choke on this?
- Why is planning for this domain exponential despite only n discs and one move operator?
- It’s exponential, so let’s just give up…

Planning Domain Description Language (PDDL)

Domain – Towers of Hanoi

```pddl
(define (domain hanoi)
  (:requirements :strips)
  (:predicates (clear ?x)
               (on ?x ?y)
               (smaller ?x ?y))
  (:action move
   :parameters (?disc ?from ?to)
   :precondition (and (smaller ?to ?disc)
                      (on ?disc ?from)
                      (clear ?disc)
                      (clear ?to))
   :effect (and (clear ?from)
                (on ?disc ?to)
                (not (on ?disc ?from))
                (not (clear ?to))))
```
PDDL Problem – Towers of Hanoi

(define (problem hanoi-3)
  (:domain hanoi)
  (:objects peg1 peg2 peg3 d1 d2 d3 )
  (:init
    (smaller peg1 d1)
    (smaller peg1 d2)
    (smaller peg1 d3)
    (smaller peg2 d1)
    (smaller peg2 d2)
    (smaller peg2 d3)
    (smaller peg3 d1)
    (smaller peg3 d2)
    (smaller peg3 d3)
    (smaller d1 d1)
    (smaller d2 d2)
    (smaller d3 d3)

  (:init continued
    (clear peg2)
    (clear peg3)
    (clear d1)
    (on d3 peg1)
    (on d2 d3)
    (on d1 d2)

  (:goal (and
    (on d3 peg3)
    (on d2 d3)
    (on d1 d2) )))

Approach I:
Planning through Graph Analysis

State-space planning by considering all possible states that can be derived from the initial state.

Graphplan plans in two phases:
- **Graph Expansion**: created a planning graph of relationships between states that can hold at each step in time.
- **Solution extraction**: backward chains over the planning graph to look for a plan that satisfies the goals.

Restricted representation: *conjunction of literals*

Identifying Mutually Exclusive Actions

Planning graphs represent parallel actions. Yet not all actions can be executed together.

Two actions are *mutually exclusive* if no valid plan could possibly contain both.

Rules for identifying that actions $a$ and $b$ are mutually exclusive:

- **Interference**: If either of the actions deletes a precondition or an add effect of the other.
- **Competing Needs**: If there is a precondition of action $a$ and a precondition of action $b$ that are marked as mutually exclusive of each other in the previous proposition level.
Mutexes in the Plan Graph


Solution Extraction

1. Add subgoals to agenda
2. For $i$ from $t$ to 1:
   1. For each literal $L$ in agenda:
      1. Choose an action $a$ at level $i-1$ that achieves $L$
         (Backtrack point, may use heuristics to choose)
      2. If $a$ is non-mutex with $actions[i-1]$, then add $a$ to $actions[i-1]$
      3. Otherwise backtrack
   2. If $actions[i-1]$ is not empty, then put preconditions of $actions[i-1]$ in agenda
   3. If preconditions of $action[0]$ are in initial conditions, then Plan Found!
   4. Otherwise try other actions at backtrack points

Approach 2: Heuristic Search

- Pure state-space search based on $A^*$
- Not necessarily optimal, sequential planning
- Needs heuristic ($h'$), preferably admissible
  - Strategy:
    - define relaxed (simplified) state space,
    - search to goal
    - count steps

Search Review

- Uninformed (“Blind Search”)
  - Random search
  - Stochastic Local Search
  - Exhaustive (Depth / Breadth First Search)
- Informed (“Heuristic Search”)
  - $A^*$ or Best First Search $f = g + h$
  - Hill Climbing Search
The Planning Graph as a Heuristic

- Provides reachability analysis
- Computing the graph for every state is as intractable as planning itself
- Compute the Relaxed Graphplan (RGP)
  - “Relax” by ignoring delete lists

Action Description using Add/Delete Effects

Action: \textit{Move}(b,x,y)
Precond: On(b,x) ^ Clear(b) ^ Clear(y)
Add Effects: On(b,y) ^ Clear(x)
Delete Effects: ^ ~Clear(y) ^ ~On(b,x)

Relaxed Plans (formally)

Let \( \Pi = (v, A, t, \pi) \) be a planning task. An action sequence \( \langle a_i \rangle \) is a relaxed plan for \( \Pi \) iff with \( s_i^* = \{(v,v') | s_i(v) = c\} \) and \( s_i^{+} = s_i^* \cup \text{eff}_{a_i} \), we have that \( s_i \subseteq s_i^* \) and that \( \text{pr} s_i \subseteq s_i^* \) for \( 1 \leq i \leq n \). The relaxed plan is optimal if \( n \) is minimal; the relaxed plan heuristic is then \( h^\star(\Pi) = n \).

- Assumes once something is true, it always is true
- Ignores all interactions
- Is admissible and consistent
- Works surprisingly well
- Is polynomial to compute, but still expensive for search
The First Planners based on State Reachability Relaxations

- **UnPOP** (McDermott, 1996): backchain from each goal in relaxed space and sum
- **HSP** (Bonet and Geffner, 1997): use simple forward chaining to compute distance from current state to a goal, then either:
  - Sum over all goals (additive costs, inadmissible), or
  - Maximum over all goals (max costs, admissible)

Fast Forward (FF): A Heuristic State Space Planner

- Phase 1: *Enforced Hillclimbing (EHC)*
  - variant of breadth-first search without state repetition
  - state value is heuristic estimate of summed distance over all goals (a la HSP)
  - Inadmissible but very effective in many problems
- Phase 2: *A*, Best First Search
  - Distance to goal estimated from the Relaxed Plan Graph
  - Helpful action pruning selects actions that are more likely to be useful (Hoffmann, 2000)

Approach 3: Compilation of Planning to SAT

SAT Plan or BlackBox by Kautz and Selman, 1996/1998

SAT Planning Issues

**Encoding**

- Propositions are fluents (with time step) in CNF, e.g., $\text{Fly}(P1, SFO, JFK, T0)$ from $\text{Fly}(P1, SFO, JFK)$
  - Precondition axioms to require precondition satisfaction
  - Action exclusion axioms to prevent illegal simultaneity
- Worst case solving time is exponential in the number of variables.
  - Parameterized Actions: e.g., in simple action splitting, each n-ary action is replaced by n unary fluents
  - Frame Axioms: need to track all fluents left unchanged from one time to next
Approach 4: Nonlinear, Least Commitment

- Plan space planners
- Add actions in any sequence
- Do not order actions until forced by problem constraints
- Enforce protection intervals (like critical sections on subgoals)

Partial Order Planners

- Plans include:
  - Actions
  - Ordering constraints
  - Variable binding constraints
  - Causal links
- Start with null plan. Gradually add to it (refine it).

Solutions

Complete: every precondition is achieved by some other step of the initial conditions.

Consistent: no contradictions in ordering (e.g., S1 < S2 ^ S2 < S1) or variable binding (e.g., x=A ^ x=B)

Blocksworld Example

```
*start*
On(c,a)  Clear(b)  Clear(c)  On(a,table)  On(b,table)  Clear(table)
On(a,b)  On(b,c)

*end*
Clear(b)  Clear(c)  On(b,table)  Clear(c)  On(c,a)  (move-c-from-a-to-table)  ~On(c,a)  Clear(a)
Clear(b)  Clear(b)  Clear(c)  On(b,table)  Clear(b)  On(b,c)  (move-b-from-table-to-c)
Clear(table)  ~On(b,table)  Clear(a)  On(a,table)  Clear(a)  On(a,b)  Clear(b)  On(a,b)
Clear(table)  ~On(a,table)  ~Clear(c)  On(b,c)  Clear(b)  Clear(a)
Clear(table)  Clear(c)  On(c,a)
~On(c,a)  Clear(a)
```

©CS540 - Spring 2013
Partial Order Planning Algorithm

Function $\text{POP}(\text{initial}, \text{goal}, \text{operators})$ returns $\text{plan}$

$\text{plan} \leftarrow \text{MAKE-MINIMAL-PLAN}(\text{initial}, \text{goal})$

loop do
    if $\text{SOLUTION?}(\text{plan})$ then return $\text{plan}$
    $S,c \leftarrow \text{SELECT-SUBGOAL}(\text{plan})$
    $\text{CHOOSE-OPERATOR}(\text{plan}, \text{operators}, S, c)$
    $\text{RESOLVE-THREATS}(\text{plan})$
end

Threats

Does Clear(A) threaten ~Clear(x)?
Possibly.

Resolutions:
- Select a non-threatening binding for x.
- Allow for inequality constraint to be carried along.
- Order: promotion or demotion
- Ignore for now.

Approach 5: Hierarchical Planning

- Plan space representation
- Decompose goal into increasingly specific subgoals and finally into actions
- Prioritize order of goal expansion
- Reason about interactions at higher level

Basics of Hierarchical Task Network (HTN) Planning

- Input:
  - Initial task network of goals to be accomplished, can include constraints on how variables can be bound, task ordering or resource use
  - Operators describing effects of actions
  - Methods describing how to achieve non-primitive tasks, represented as other task networks (aka abstract operators).
HTN Example

 HTN Planning

- Problem reduction
  - Tasks (activities) rather than goals
  - Methods to decompose tasks into subtasks
  - Enforce constraints
    - E.g., taxi not good for long distances
  - Backtrack if necessary

Basic HTN Algorithm

1. Until no non-primitive tasks remain
   1. Find a non-primitive task \( u \) in \( g \) (network of goals) and a method \( m \) that satisfies it
   2. Replace \( u \) with the task network \( d' \) from \( m \)
   2. Order \( d \) according to constraints
   3. Check for threats or conflicts (critics)

Analytical Results of HTN Planning

Decidability:
- Plan existence is semi-decidable, even if \( P \) is restricted to be propositional, to have at most 2 tasks in any network and to be totally ordered
- Plan existence is decidable if the methods are acyclic.

Expressivity:
- There exist HTN planning domains that cannot be represented by any finite number of STRIPS-style operators.

Planning Summary

- Many approaches suited to different types of problems.
- No unified theory as yet.
- State of the art:
  - simple problems can be solved with guarantees by classical planners and POMDP planners
  - Heuristic search can solve larger STRIPS and ADL problems.
  - Knowledge based planners solve real applications, but without guarantees.