Foundations of Artificial Intelligence

Planning

CS472 – Fall 2007
Thorsten Joachims

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

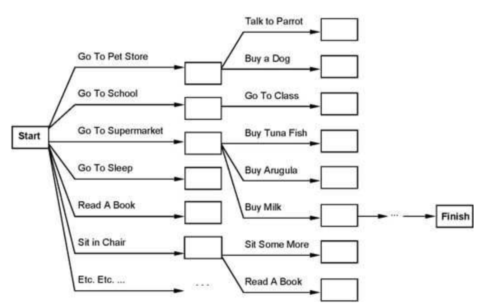
Representation of actions
actions generate successor states

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

Planning Example



GOAL: Get a quart of milk and a bunch of bananas and a variable-speed cord-less drill.

Planning vs. Problem Solving

1. **Open up the representation of states, goals and actions.**
   - States and goals represented by sets of sentences – *Have* (*Milk*)
   - Actions represented by rules that represent their preconditions and effects: *Buy*(*x*) achieves *Have*(*x*) and leaves everything else unchanged

$\Rightarrow$ **This allows the planner to make direct connections between states and actions.**

Planning vs. Problem Solving

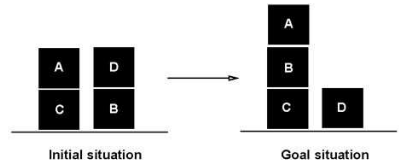
2. **Most parts of the world are independent of most other parts.**
   - Can solve $Have(Milk) \wedge Have(Bananas) \wedge Have(Drill)$ using divide-and-conquer strategy.
   - Can re-use sub-plans (go to supermarket)

Planning vs. Problem Solving

3. 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.

Planning as a Logical Inference Problem



Axioms:

On(A,C) On(C,Table), On(D,B), On(B,Table), Clear(A), Clear(D)

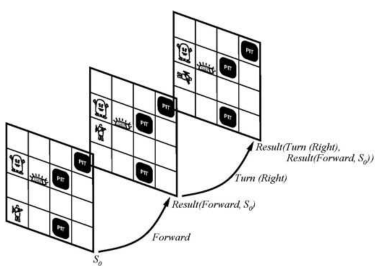
Plus rules for moving things around…

Prove: On (A,B) Æ 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.



Situation Calculus

Fluents: functions and predicates that vary from one situation to the next

on(*A,C*)	*on*(*A,C,*S_0)
at(*agent*,[1,1])	*at*(*agent*, [1,1], S_0)

Atemporal functions and predicates: true in any situation

block(A)

gold(G_1)

Situation Calculus: Actions

Actions are described by stating their effects.

Possibility Axiom: *preconditions* $\Rightarrow$ *Poss(a,s)*.

$$\forall s \forall x \neg On(x, Table, s) \land Clear(x,s) \Rightarrow Poss(PlaceOnTable(x),s)]$$

Effect Axiom: *Poss(a,s)* $\Rightarrow$ *Changes that result from action.*

$$\forall s \forall x Poss(PlaceOnTable(x), s) \Rightarrow$$
$$On(x, Table, Result(PlaceOnTable(x), s))$$

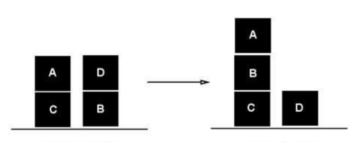
Situation Calculus: Action Sequences

We'd like to be able to prove:

$$\exists\, seq\ \ On(A,B,Result(seq,S_0)) \land On(B,C,Result(seq,S_0))$$

Which would produce, for example, the following:

$$On(A,B,Result(Put(A,B),Result(Put(B,C),Result(PoT(D),Result(PoT(A),S_0)))))$$
$$\land$$
$$On(B,C,Result(Put(A,B),Result(Put(B,C),Result(PoT(D),Result(PoT(A),S_0)))))$$



Situation Calculus: Problem

Axioms:

On(A,C,S_0) On(C,Table,S_0), On(D,B,S_0), On(B,Table,S_0), Clear(A,S_0), Clear(D,S_0)

$\forall s \forall x \neg On(x, Table, s) \wedge Clear(x, s) \Rightarrow Poss(PlaceOnTable(x), s)]$

$\forall s \forall x Poss(PlaceOnTable(x), s) \Rightarrow$
$On(x, Table, Result(PlaceOnTable(x), s))$

Prove:

1. On(A,Table,Result(PoT(A),S0))
2. On(D,B,Result(PoT(A),S0))

The Frame Problem

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

$$\forall s \forall x Poss(PoT(x), s)) \Rightarrow On(x, Table, Result(PoT(x), s))$$

Frame Axioms: Inform the system about preserved relations.

$$\forall s \forall x \forall y \forall z [(On(x, y, s) \wedge (x \neq z)) \Rightarrow On(x, y, Result(PoT(z), s))]$$

… and Its Relatives

Representational Frame Problem: proliferation of frame axioms.

Solution: use successor-state axioms

Action is possible ⇒ (Fluent is true in result state ⇔ (Action's effect made it true ∨ It was true before and action left it alone)).

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

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.

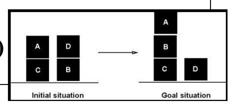
Solution for the planning problem:
An action sequence that, when executed in the initial state, results in a state that satisfies the goal.

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: □

Operators:

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

Partial Plans

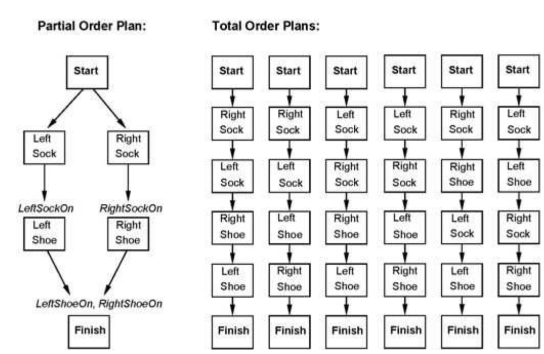
Partial Plan: RightShoe LeftShoe

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



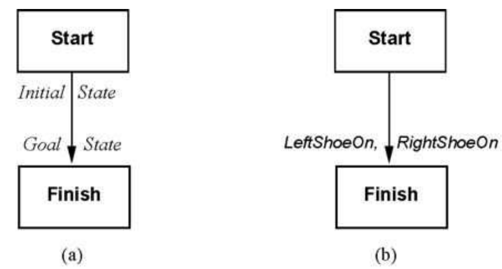
Definition of a Partially-Ordered Plan

- **A set of plan steps (actions).**
- **A set of step ordering constraints of the form** $S_i \prec S_j$ **written as** $S_i \longrightarrow S_j$
- **A set of variable binding constraints**
- **A set of causal links, written as**

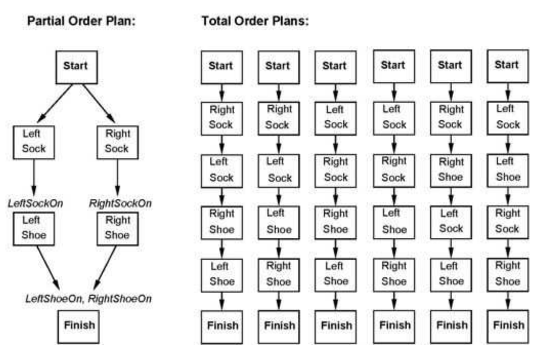
$$S_i \xrightarrow{c} S_j$$

Initial Plan for Shoes and Socks

Initial plan: $Start \prec Finish$



Partial Plan for Shoes and Socks



Planner Output

A solution is a complete, consistent plan.

1. **A complete plan:** every precondition of every step is achieved by some other step.
2. **A consistent plan:** there are no contradictions in the ordering or causal constraints. Contradiction occurs when both $S_i \prec S_j$ and $S_j \prec S_i$, or when there is a conflict between two causal links.
   - A **conflict** exists when two causal links for some literal and its negation are not strictly ordered.

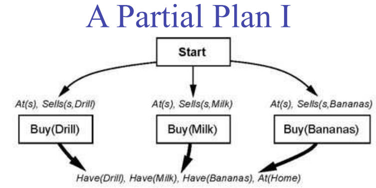
POP Example

Actions:

Action	PreCond	Effect
Go(*there*)	At(*here*)	At(*there*) ∧ ¬At(*here*)
Buy(*x*)	At(*store*) ∧ Sells(*store*, *x*)	Have(*x*)

Initial Plan:

A Partial Plan I

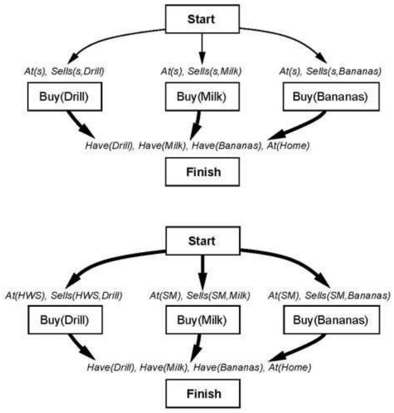


Planners must commit to bindings for variables
Example: Goal: Have(Milk) Action: Buy(item,store)

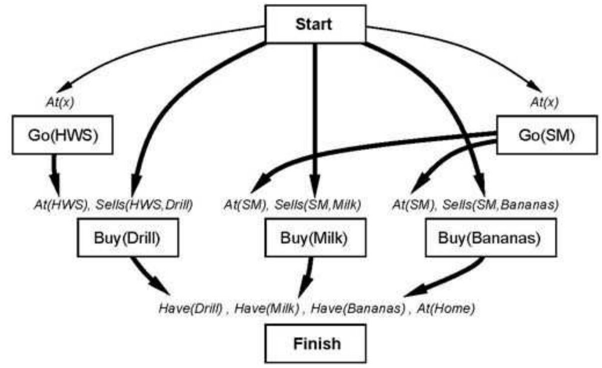
Principle of Least Commitment: Only make choices about things that you care about, leaving other details to be worked out later.
Buy(Milk,K-MART) versus Buy(Milk,store)

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

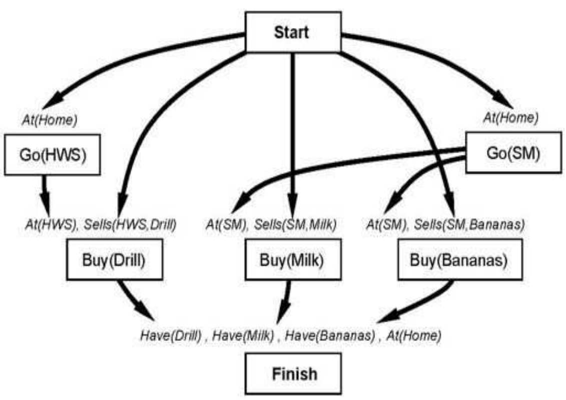
A Partial Plan II



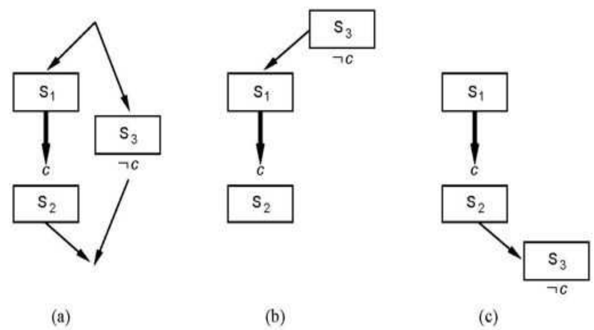
A Partial Plan III



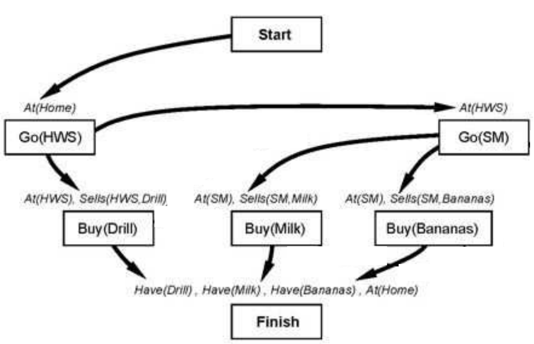
A Partial Plan IV



Protecting Causal Links



A Partial Plan IV'

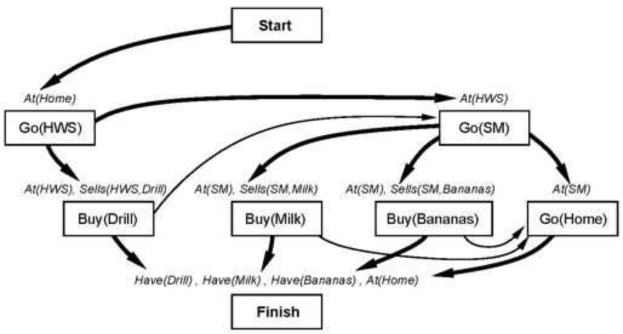


Achieving At(Home)

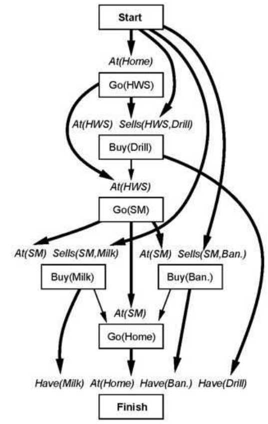
Candidate link	Threats
At(x) to initial state	Go(HWS), Go(SM)
At(x) to Go(HWS)	Go(SM)
At(x) to Go(SM)	At(SM) preconds of Buy(Milk), Buy(Bananas)

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

A Partial Plan V



A Final Plan




```
function POP(initial, goal, operators) returns plan

  plan ← MAKE-MINIMAL-PLAN(initial, goal)
  loop do
      if SOLUTION?(plan) then return plan
      S_need, c ← SELECT-SUBGOAL(plan)
      CHOOSE-OPERATOR(plan, operators, S_need, c)
      RESOLVE-THREATS(plan)
  end

function SELECT-SUBGOAL(plan) returns S_need, c

  pick a plan step S_need from STEPS(plan)
      with a precondition c that has not been achieved
  return S_need, c

procedure CHOOSE-OPERATOR(plan, operators, S_need, c)

  choose a step S_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_add --c--> S_need to LINKS(plan)
  add the ordering constraint S_add ≺ S_need to ORDERINGS(plan)
  if S_add is a newly added step from operators then
      add S_add to STEPS(plan)
      add Start ≺ S_add ≺ Finish to ORDERINGS(plan)

procedure RESOLVE-THREATS(plan)

  for each S_threat that threatens a link S_i --c--> S_j in LINKS(plan) do
      choose either
          Promotion: Add S_threat ≺ S_i to ORDERINGS(plan)
          Demotion: Add S_j ≺ S_threat to ORDERINGS(plan)
      if not CONSISTENT(plan) then fail
  end
```

Strengths of Partial-Order Planning Algorithms

- **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:

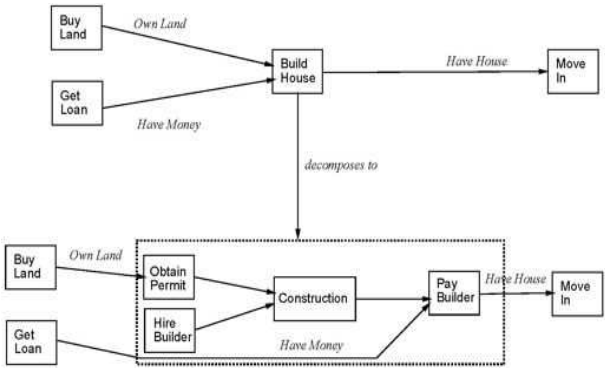
- **Resources:** Operators should incorporate resource consumption and generation. Planners have to handle constraints on resources efficiently.
- **Time:** Real-world planners need a better model of time.
- **Hierarchical plans:** need the ability to specify plans at varying levels of details.

Also need to incorporate heuristics for guiding search.

Planning Graphs

- Data structure (graphs) that represent plans, and can be efficiently constructed, and that allows for better heuristic estimates.
- **Graphplan:** algorithm that processes the planning graph, using backward search, to extract a plan.
- **SATPlan:** algorithm that translates a planning problem into propositional axioms and applies a CSP algorithm to find a valid plan.
- **Take CS672 / CS475 to learn more!!**

Hierarchical Planning



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.**

Scheduling for Space Missions

- **Planners have been used by 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.**