Planning

Burr H. Settles
CS-540, UW-Madison
www.cs.wisc.edu/~cs540-1
Summer 2003

Announcements (7/7)

- Lots of homework business!
  - HW#3 out (due Monday, 7/14)
  - HW#2 due today
  - HW#1 *almost* graded, back tomorrow
- Reminders:
  - Midterm review on Wednesday (7/9)
  - Midterm on Thursday, in class (7/10)
  - No class on Friday (7/11)

Announcements (7/8)

- Homework #1 is *not quite* graded!
  - Check my mailbox in CS 5th floor after 5pm today
  - Otherwise, collect them at the review tomorrow
  - There is a solution for HW#1 on the webpage
- Homework #3 due date extended (Wed. 7/16)
- About the exam:
  - *Closed book*, but you may bring a 1-sided handwritten 8½×11 sheet of notes and a calculator
  - I threw together a midterm study guide available on the course webpage under "exam" section

Planning

Problem:

∗ *Mechanically and efficiently find a sequence of actions that, when "executed," achieve a goal*

- Given:
  - Initial state, goal state, and actions
- Find:
  - A plan: a sequence of actions that when applied, beginning with the initial state, transforms the world into a goal state

Assumptions with Planning

- Goal is a conjunction of sub-goals:
  - To achieve a goal, you must achieve a set of sub-goals
- Actions are atomic
  - Are not divisible into sub-actions
- Actions are sequential
  - No two actions can be executed concurrently
- Actions are deterministic:
  - No uncertainty in performing an action

Assumptions with Planning

- The agent is the sole cause of change in the environment
- World is accessible (*i.e.* the agent knows all it need to know about the environment)
- Closed World Assumption:
  - State description lists all that is true
  - Anything else is assumed false

∗ *The planning task is very difficult, even with such a simplified framework!*

Classic Planning Problems

- Dressing
  - **Initial state:** socks, shoes, and pants off
  - **Goal state:** socks on, under shoes (on correct feet), under pants
  - **Actions:** PutOnPants, PutOnSock(*f*), PutOnShoe(*f*)
- Blocks World
  - **Initial state:** some configuration of blocks on a table
  - **Goal State:** another configuration (stacked?)
  - **Actions:** Pickup(*x*), Putdown(*x*), Stack(*x*,*y*), Unstack(*x*,*y*)
- Shopping
  - **Initial state:** at home, with no items
  - **Goal state:** at home, having a list of items
  - **Actions:** Go(*store*), Buy(*item*), etc…

Planning As Search

- State-space search:
  - State representation
  - Operators/actions
  - Start state
  - Goal test

∗ *Note: This is how we approached the "water jugs" problem back in lecture 3*

Planning As Search

- Doesn't allow for real *reasoning* about states and actions
  - Operators are just used to generate next state
  - Can't reason about operator definition or ordering
    - Causes exploration of dead-end successor states (possibly even illegal ones!)
  - Goal test is just used to determine if goal reached
  - Can't reason about goal definition
    - No knowledge of determining how to best achieve the goal
    - *Note: heuristic is simply the distance from goal*
- Weak representation
- Weak ability to reason about the world

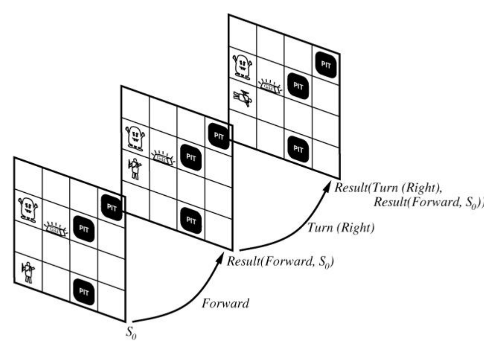
Planning Using Logic

- By using knowledge-based agents, we can capture reasonable information things about the agent's actions and their effect on the world
  - "If I move forward, I'm in the next room"
  - "If I pick up a gold brick, then I am holding it"
  - "If I am holding something, my hand is not empty"
- The problem here is dealing with time:
  - "If I move forward *again*, I'm in a *different* room"
  - The results of each action are now relative to the sequence of actions before…

Situation Calculus

- Situation calculus extends FOL to deal with such time-sensitive dilemmas for planning (Sec. 10.3)
  - Situations are states that are generated from applying an action to another situation
    - *Result*(*a*, *s*) is the function that returns the situation when applying action *a* to situation *s*
  - Fluents are predicates/functions that vary from one situation to the next, such as the location of the agent, or what it may be holding
  - Atemporals/eternals are predicates/functions that do not depend on a time stamp
    - *e.g.* Dog(*Lassie*) or LeftLegOf(*John*)

Situation Calculus



Situation Calculus

- There are two types of axioms (or rules) in situation calculus:
  - Possibility axioms: say when it is possible to perform a certain action
    - $\text{At}(Agent, x, s) \wedge \text{Adjacent}(x, y) \Rightarrow \text{Poss}(\text{Go}(x, y), s)$
    - $\text{Gold}(g) \wedge \text{At}(g, x, s) \wedge \text{At}(Agent, x, s) \Rightarrow \text{Poss}(\text{Pickup}(g), s)$
    - $\text{Holding}(g, s) \Rightarrow \text{Poss}(\text{Putdown}(g), s)$
  - Effect axioms: defines what happens in the environment when a possible action is executed
    - $\text{Poss}(\text{Go}(x, y), \text{s}) \Rightarrow \text{At}(Agent, y, \text{Result}(\text{Go}(x,y), s))$
    - $\text{Poss}(\text{Pickup}(g), s) \Rightarrow \text{Holding}(g, \text{Result}(\text{Pickup}(g), s))$
    - $\text{Poss}(\text{Putdown}(g), s) \Rightarrow \neg\text{Holding}(g, \text{Result}(\text{Putdown}(g), s))$

13

Situation Calculus

- Fortunately, situation calculus allows us to express what actions are reasonable as well as what will change when an action is taken
- Unfortunately, it doesn't say anything about what *stays the same*!
- Frame axioms specify what does not change when a certain action is applied
  - *e.g.* "If I go into a room that had gold in it during the last situation, then the gold is still there"
  - Many axioms are required (for each action even!)

14

Situation Calculus

- Situation calculus with frame axioms is a strong representation
  - However, the approach is not very modular… each new predicate requires axioms to be added for each of the possible actions

✱ *Inference procedures are very weak… the representation is too fine-grained*

15

Planning Solution

Combine the two approaches:

- Simplify the representation language
  - Allow reasoning about how to achieve the goal
  - Inference procedure is faster than resolution
- "Open up" the representation of states, operators, and goal test
  - Rather than blindly applying operators, try to reason about which ones are most important
  - Reduces the number of nodes that are considered

16

STRIPS Representation

STRIPS (STandard Research Institute Problem Solver):

- Facts: ground literals with variables
- Situations: conjunction of facts
- Goal: conjunction of positive literals
  - Variables allowed, assume all variables are existential
- Operators/Actions:
  - **Action name**
  - **Preconditions:** conjunction of positive literals that defines if action is legal/applicable
  - **Effects:** conjunction of positive literals (called the add list) and negative literals (called the delete list)
  - **Assumption:** everything stays the same unless explicitly on the delete list (avoids frame problem)

17

Representation for Planning

- Operator Examples:

– Action name:	Buy(x)
– Preconditions:	At(s), Sells(s,x)
– Effects:	Have(x)
– Action name:	Pickup(x)
– Preconditions:	OnTable(x), Clear(x), HandEmpty
– Effects:	Holding(x), ¬OnTable(x), ¬Clear(x), ¬HandEmpty

18

Planning as Seach

- Situation-space search:
  - **Search space:** all possible situations (*i.e.* states)
  - **Node:** situation (*i.e.* world state)
  - **Edges:** actions
  - **Start node:** initial situation
  - **Goal node:** situation where all of the sub-goals solved
  - **Plan:** sequence of actions in path from start to goal
- Plan-space search:
  - **Search space:** all possible plans
  - More later…

19

More on the Nature of Plans

- *A plan is complete if and only if every precondition is achieved*
- *A precondition is achieved if and only if it is the effect of an earlier step (and no intervening steps undo it)*

20

Situation-Space Planners

- Progression: Forward Chaining
  - Like state-space search except for representation
  - Inefficient due to large situation space to explore
- Regression: Backward Chaining (*e.g.* Prolog)
  - Start from the goal state and solve its sub-goals (preconditions)
  - More efficient and goal-directed than progression (fewer applicable operators)

21

Example

- Putting on pants, socks, and shoes

  Start:
  PantsOff, SockOff(L), SockOff(R), ShoeOff(L), ShoeOff(R)

  Goal:
  PantsOn, SockOn(L), SockOn(R), ShoeOn(L),ShoeOn(R)

  Operators:
  - **PutOnPants:** **Pre:** PantsOff, ShoeOff(L), ShoeOff(R)
    Eff: PantsOn, ¬PantsOff
  - **PutOnSock(*x*):** **Pre:** ShoeOff(*x*), SockOff(*x*),
    Eff: SockOn(*x*), ¬SockOff(*x*)
  - **PutOnShoe(*x*):** **Pre:** ShoeOff(*x*), SockOn(*x*)
    Eff: ShoeOn(*x*), ¬ShoeOff(*x*)

22

Goal-Stack Regression Planner

- Goal stack: what to do next
- Current situation: facts that are true
- Pick order of achieving (sub-)goals
  - Find operator that achieves the (sub-)goal
  - Push the operator onto stack
  - Push its preconditions (in some order) onto stack
  - When eventually get back to original goal, check that all of the preconditions that were needed to be satisfied are still satisfied

23

Key Assumption in STRIPS

∗ *Sub-goals are independent of each other*
  - Divide and conquer the problem without worrying about other parts of the problem
    - *e.g.* With putting on socks: the order doesn't matter; putting on left sock first doesn't preclude putting on the right
  - Whole plan is sum of all sub-plans
- Sussman anomaly
  - Sub-goals interfere with each other
    - *e.g.* Blocks world tower, can't fix with reordering
  - Thus, STRIPS is incomplete:
    (*i.e.* can't always find a plan even if one exists)

24

The Sussman Anomaly

Start State

Goal State

- Stacking A on top of B precludes us from stacking B on top of C
  - We cannot pick it up because it is no longer clear!
  - Imagine stacking 100 blocks…

Interleaving in Planning

- Non-interleaving planners
  - All of the steps for a sub-goal must occur "atomically"
  - Given two sub-goals G_1 and G_2, either all the steps for achieving G_1 occur before G_2, or vice-versa
  - STRIPS is non-interleaving because it uses a stack mechanism (solves one sub-goal at a time)
- Interleaving planners
  - Can intermix the order of sub-goal steps
  - This solves the Sussman anomaly

Partial-Order Plans (Sec. 11.3)

- Total-order planner (linear):
  - Maintains a partial solution as a "totally ordered" list of steps found so far
  - *e.g.* STRIPS
  - *e.g.* Situation-space progression/regression planners
- Partial-order planner (non-linear):
  - Only maintains partial order
  - Constraints on the ordering of steps in the plan

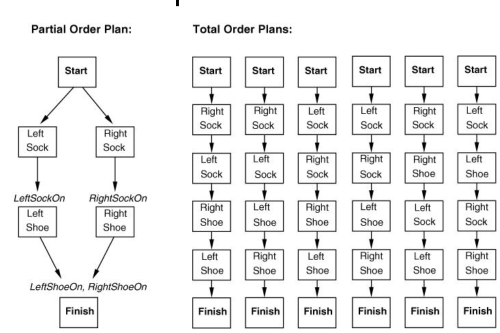
Principle of Least Commitment

- Principle of Least Commitment: don't make an ordering choice *unless required to do so*
  - Property of partial-order planners (POP)
  - Not a property of situation-space planners: they commit to an ordering when an operator is applied
- Keep the ordering choice as general as possible
- Reduces the amount of backtracking needed
  - Don't waste time undoing steps

Planning as Search: Revisited

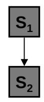
- Situation-space search:
  - **Search space:** all possible situations (*i.e.* states)
  - etc…
- Plan-space search:
  - **Search space:** all possible partial-order plans
  - **Node:** a partially-order plan
  - **Edges:** add/delete/modify steps of previous node's plan or add temporal and causal constraint between existing steps
  - **Start node:** initial partial-order plan, *start* → *finish* where *start*: pre = none, eff = positive literals defining start state and *finish*: pre = goal of conjunctive literals, eff = none
  - **Goal node:** a complete plan that solves all sub-goals

POP Example

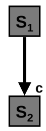


Types of "Links"

- Ordering constraints:
  - $S_1 < S_2$: S_1 before S_2
  - S_1 must occur before S_2 but not necessarily *immediately* before it
  - *Thin* links



- Causal constraints:
  - $S_1 \rightarrow_c S_2$: S_1 achieves c for S_2
  - S_1 has a literal c in its effect list that is needed to satisfy part of the precondition for S_2
  - Records the purpose of a step in the plan
  - *Thick* links

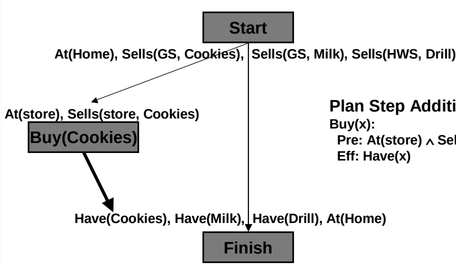


Solving Open Preconditions

- A open (*i.e.* unsatisfied) precondition is one that does not have a causal link to it
- How is an open precondition *p* for step *S* solved?
  - Step addition: add new plan step *R* that contains *p* in its Effects list
  - Simple establishment: find an *existing* plan step *R* prior to *S* that has *p* in its Effects list
  - Then add a causal and ordering links from *R* to *S*

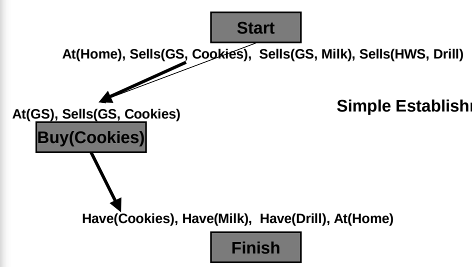
✱ *To keep the search focused, the planner only adds steps that achieve an open precondition*

Example: Shopping Problem



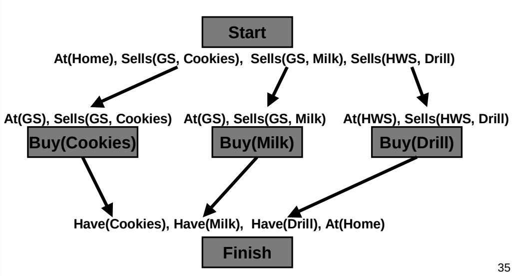
Plan Step Addition
Buy(x):
Pre: At(store) ∧ Sells(store,x)
Eff: Have(x)

Example: Shopping Problem



Simple Establishment

Example: Shopping Problem

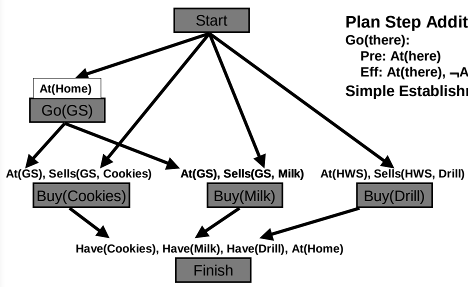


Finishing the Algorithm

- The algorithm is finished when every precondition in every step has a causal link
- The algorithm fails if a precondition cannot be satisfied *or* an ordering constraint cannot be met
  - *e.g.* $S_1 < S_2$ and $S_2 < S_1$

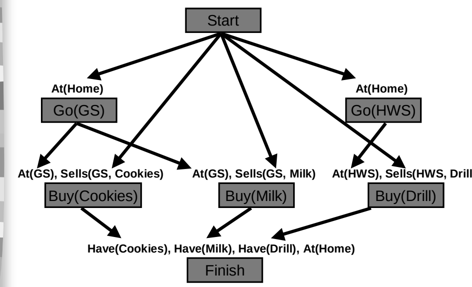
A Flawed Shopping Plan

Plan Step Addition
Go(there):
Pre: At(here)
Eff: At(there), ¬At(here)
Simple Establishment



37

A Flawed Shopping Plan



38

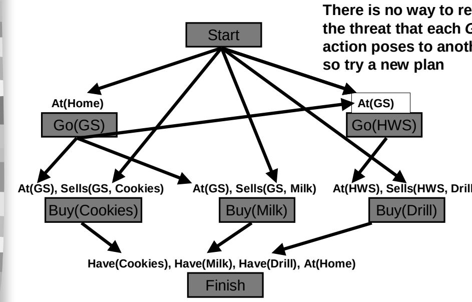
Threat Removal (Declobbering)

- Threat: step that deletes (clobbers) a needed effect
  - S_2 requires an effect of S_1
    (*i.e.* there is a causal link between S_1 and S_2),
    but the effect of S_3 is to undo the needed effect
- Thus S_3 can't occur between S_1 and S_2
  - It must occur either before S_1 (demotion)
    - Add link $S_3 < S_1$
  - Or after S_2 (promotion)
    - Add link $S_2 < S_3$

39

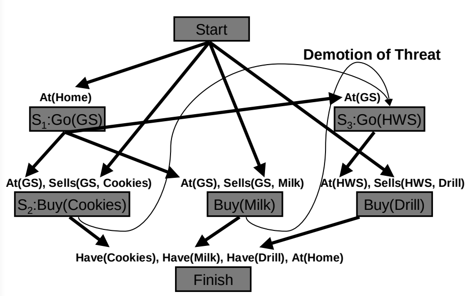
Threat Removal

There is no way to remove the threat that each *Go* action poses to another… so try a new plan



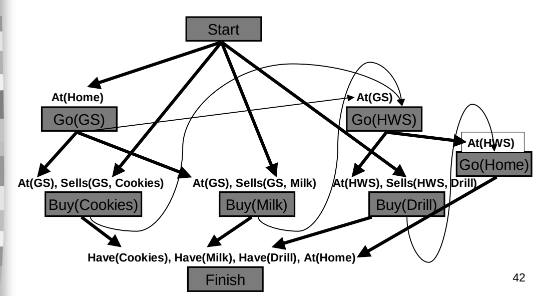
40

Threat Removal



41

Completing the Plan



42

Historical AI Planning

- State-space search (STRIPS) can be directed using logic, but is still incomplete
- Partially-ordered planners are complete, but are practically limited in the number of steps they can accurately plan

✱ *Planning was sort of a "dead" AI research area for a while*

43

Modern AI Planning

- Since 1992, there have been several new approaches to the planning task discovered (*e.g.* Graph-Plan and SAT-Plan) that can find plans up to thousands of steps long
  - CS-731 goes into these approaches in detail

🗎 D. Weld, "Recent advances in AI planning," *AI Magazine*,1999
  - Excellent coverage of these new approaches

44

Graph-Plan (Sec. 11.4)

🗎 A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis," *Artificial Intelligence*, 1997

- Propositionalize actions and situations
- Construct a planning graph
  - Levels (*e.g.* time steps) with potential action nodes
    - Include persistence actions (inactions) to deal with frame prob.
  - Link actions to situation nodes between each level
  - Indicate which situation descriptions are mutually exclusive with "mutex links"

45

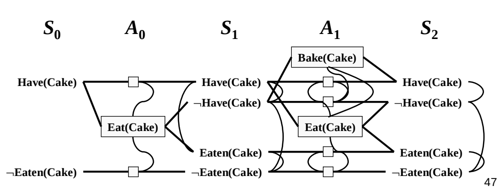
Graph-Plan

```
// basic graph-plan algorithm (p.399)
GRAPH = initial state graph
GOALS = problem goals
loop forever {
  if GOALS non-mutex in last level of graph then {
    SOL = extract_sol(GRAPH, GOALS, len(GRAPH))
    if SOL ≠ failure then return SOL
    else if no_sol(GRAPH) then return failure
  }
  GRAPH = expand_graph(GRAPH, problem definition)
}
// See textbook or paper for more details on computing
// mutex, algorithmically finding solutions, etc...
```

46

Graph-Plan

Start: Have(Cake) $\wedge$ $\neg$Eaten(Cake) **Goal:** Have(Cake) $\wedge$ Eaten(Cake)
Action: Eat(Cake)
Precond = Have(Cake) / Effect = $\neg$Have(Cake) $\wedge$ Eaten(Cake)
Action: Bake(Cake)
Precond = $\neg$Have(Cake) / Effect = Have(Cake)



47

SAT-Plan (Sec. 11.5)

🗎 H.A. Kautz and B. Selman, "Planning as Satisfiability," *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI)*, 1992

- Recall that a planning environment can be expressed in situation calculus
  - Axioms of the form $\alpha \Rightarrow \beta$ (rather $\neg\alpha \vee \beta$)
- Recall that plans are considered to be a conjunction of sub-goals:
  - Start state $\wedge$ axioms $\wedge$ goals

48

SAT-Plan

- The basic idea with SAT-Plan:
  - Describe the environment in situation calculus
  - Propositionalize all the axioms (disjunctions), enumerated for each of an arbitrary number of steps
  - Conjoin all instantiated rules with the initial state and goal descriptions

✱ *This provides us with a PL formula in CNF, which we can try to solve using HC, SA, Tabu, GAs, etc.*

49

SAT-Plan

Problem	**# of variables**	**# of clauses**	**SAT-Plan**
anomaly	94	933	0.1 sec
reverse	215	2,533	4 sec
medium	244	3,025	1.2 sec
hanoi	288	3,798	13 hours

- SAT-PLAN isn't necessarily complete
  - Using local search, can get stuck in local optima
  - Using exhaustive heuristic search (*e.g.* DPLL), it is complete but can take a long time

50

Summary

- Planning agents search to find a sequence of actions to achieve a goal using a flexible representation of states, operators, goals, plans
  - STRIPS language describes actions in terms of their preconditions and effects
- Not feasible to search through the entire space as was done with search agents
  - Regression planning focuses the search
  - STRIPS assumes sub-goals are independent
  - POP uses principle least commitment, declobbering

51

Summary

- Partial-Order Planning (POP) is a sound and complete planning algorithm, but can be limited by plan length
- Recent advances in AI planning reduce the planning environment to other problems (Graphs, SAT formulas) that can be solved using other methods

52

Next Lecture

- After the Midterm:

Machine Learning!

53