

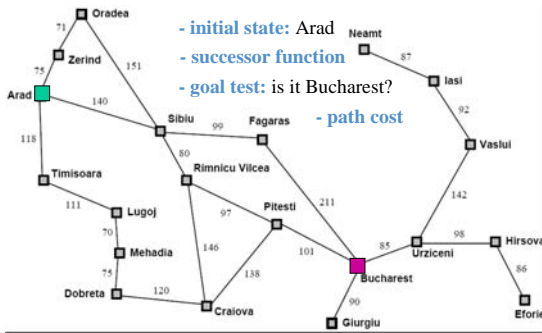
Planning

Chapter 11
Sections 11.1-11.4

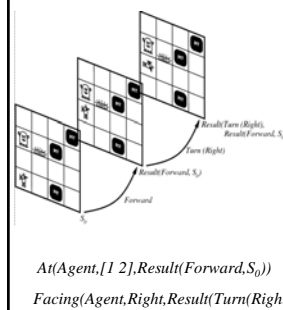
Planning

- Coming up with a sequence of actions that will achieve a goal

Search-based planning (ch. 3)



Logical planning (Ch. 10)



- Initial condition in KB:
 - $At(Agent, [1, 1], S_0)$
 - $At(Gold, [1, 2], S_0)$
 - plus, all the axiomatic rules
- Query: in what situation will I be holding the gold?
 - $ASK(KB, \exists s. Holding(Gold, s))$
- Answer: substitution
 - $\{s\}$
 - $Result(Grab, Result(Forward, S_0))$
 - go forward, then grab the gold

Today: scaling up

- Complex planning problems
- Combining search and logic for planning
- STRIPS language and extension
- Planning with state-space search
- Partial-order planning
- Blocks world

Planning problem

- Fully observable, deterministic, finite, discrete, static environments (classical planning)
 - more realistic planning for a final project? (ch. 12)

Planning problem example

- Buy four books *A, B, C,* and *D* online
 - *A* = ISBN0137903952 (AIMA)
 - 10 billion actions (one per ISBN) – branching factor
 - 10^{40} possible plans for 4 books
- What's a good heuristic function?
 - number of books yet to buy
 - but state is a black box to a search agent

Oct 15, 2008

COMP 131 Lecture 10

9

Domain-independent representation

- Goal state: a conjunction of subgoals
 - $Have(A) \wedge Have(B) \wedge Have(C) \wedge Have(D)$
- Heuristic: number of unsatisfied conjuncts (subgoals)
 - in state where only $Have(A) \wedge Have(B)$ true, $h=2$
- Bonus: problem decomposition
 - satisfy subgoals independently
 - problems may be **nearly decomposable**
 - Planner may work on subgoals independently but some work needed to combine sub-solutions

Oct 15, 2008

COMP 131 Lecture 10

10

More examples

- Mobile robot to navigate to a goal
- Robotic arm to reach a goal position in a workspace with obstacles
- Automated assembly of parts into a configuration
- Travel route planning (e.g., cargo shipping)
- “Todo” list order optimization

Oct 15, 2008

COMP 131 Lecture 10

11

Language of planning

- States:
 - decompose into logical conditions
 - represent as *conjunction of positive literals*
 - $Poor \wedge Unknown$ (PL)
 - $At(Plane1, Melbourne) \wedge At(Plane2, Sydney)$ (FOL)
 - literals must be *ground* and *function-free*: $At(x,y)$ or $At(Father(Fred), Sydney)$ not allowed
- Goal:
 - a partially specified state
 - represent as conjunction of positive literals
 - $Rich \wedge Famous$ (PL), $At(Plane2, Tahiti)$ (FOL)
- In PL, a state *s* **satisfies** a goal *g* if *s* contains all the literals in *g* (and maybe some others)
 - $Rich \wedge Famous \wedge Miserable$ satisfies $Rich \wedge Famous$

Oct 15, 2008

COMP 131 Lecture 10

12

Language of planning (cont.)

- Action schema:
 - preconditions that must hold before it can be executed
 - effects of action
- name + parameters** **positive ground literals only**
- $Action(Fly(p,from,to),$
 PRECOND: $At(p,from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
 EFFECT: $\neg At(p,from) \wedge At(p,to)$
- positive literals asserted to be true,**
negative literals asserted to be false
- Actions are instances of action schemas derived by assigning variables to constants

Oct 15, 2008

COMP 131 Lecture 10

13

Semantics of planning rep.

- Describe how actions affect states
 - Action **applicable** in any state that satisfies the preconditions (otherwise, no effect)
 - for a FOL action schema, establishing applicability involves a *substitution* for the variables in the preconditions
- $Action(Fly(p,from,to),$
 PRECOND: $At(p,from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
 EFFECT: $\neg At(p,from) \wedge At(p,to)$
- STATE: $At(P1,BOS) \wedge At(P2,SFO) \wedge Airport(BOS) \wedge Airport(SFO) \wedge Plane(P1) \wedge Plane(P2)$
- satisfies precondition with substitution: $\{p/P1,from/BOS,to/SFO\}$
 (other substitutions possible)

Oct 15, 2008

COMP 131 Lecture 10

14

Semantics of planning rep. (cont.)

- Result of action a in state s is a new state s' same as s except:
 - any positive literal P in the effect of a is added to s no duplications
 - any negative literal $\neg P$ in the effect of a is removed from s

$Action(Fly(p,from,to),$
 PRECOND: $At(p,from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
 EFFECT: $\neg At(p,from) \wedge At(p,to)$)

$s: At(P1,BOS) \wedge At(P2,SFO) \wedge Airport(BOS) \wedge Airport(SFO)$
 $\wedge Plane(P1) \wedge Plane(P2)$

Applying $Fly(P1,BOS,SFO)$ to state s gives:

$s': At(P1,SFO) \wedge At(P2,SFO) \wedge Airport(BOS) \wedge Airport(SFO)$
 $\wedge Plane(P1) \wedge Plane(P2)$

Oct 15, 2008

COMP 131 Lecture 10

15

STRIPS representation

- STanford Research Institute Problem Solver
- STRIPS assumption: every literal not mentioned in the effects of action remains unchanged
 - avoiding the frame problem

Oct 15, 2008

COMP 131 Lecture 10

16

Air cargo transport in STRIPS

$Init(At(C1,SFO) \wedge At(C2,BOS) \wedge At(P1,SFO) \wedge At(P2,BOS)$
 $\wedge Cargo(C1) \wedge Cargo(C2) \wedge Plane(P1) \wedge Plane(P2)$
 $\wedge Airport(SFO) \wedge Airport(BOS))$

$Goal(At(C1,BOS) \wedge At(C2,SFO))$

$Action(Load(c,p,a),$

PRECOND: $At(c,a) \wedge At(p,a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
 EFFECT: $\neg At(c,a) \wedge In(c,p)$)

$Action(Unload(c,p,a),$

PRECOND: $In(c,p) \wedge At(p,a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
 EFFECT: $At(c,a) \wedge \neg In(c,p)$)

$Action(Fly(p,from,to),$

PRECOND: $At(p,from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
 EFFECT: $\neg At(p,from) \wedge At(p,to)$)

Oct 15, 2008

COMP 131 Lecture 10

17

Solution of a planning problem

- ... is a sequence of actions that, when executed in the initial state, results in a state that satisfies the goal.

Oct 15, 2008

COMP 131 Lecture 10

18

Expressiveness of STRIPS

- Important restriction: literals are *function-free*
 - every action schema can be converted to PL
 - $Fly(p,from,to)$ with 10 planes, 5 airports gives 250 PL actions
 - we will see planners that work directly with PL descriptions; allowing functions would result in infinitely many states and actions
- STRIPS too restrictive for some real domains

Oct 15, 2008

COMP 131 Lecture 10

19

Extension to STRIPS: ADL

STRIPS	Action Description Language (ADL)
Only positive literals in states	Positive and negative ok
Closed world assumption: unmentioned literals are false	Open world assumption: unmentioned literals are unknown
Effect $P \wedge \neg Q$ means add P and delete Q	Effect $P \wedge \neg Q$ means add P and $\neg Q$ and delete $\neg P$ and Q
Only ground literals in goals: $At(P1,BOS) \wedge At(P2,BOS)$	Quantified variables in goals: $\exists x. At(P1,x) \wedge At(P2,x)$ is the goal of having $P1$ and $P2$ in the same place
Goals are conjunctions only: $Rich \wedge Famous$	Goals allow disjunctions too: $\neg Poor \wedge (Famous \vee Happy)$
Effects are conjunctions only.	Conditional effects allowed: when $P: E$ (effect E only if P satisfied)
No equality	Equality predicate ($x=y$)
No types	Variables can have types ($p: Plane$)

Oct 15, 2008

COMP 131 Lecture 10

20

Fly action in ADL

Action($Fly(p: Plane, from: Airport, to: Airport)$,
 PRECOND: $At(p, from) \wedge \neg(from=to)$
 EFFECT: $\neg At(p, from) \wedge At(p, to)$

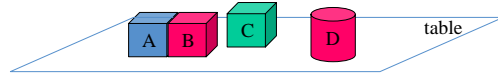
- A flight cannot be made from an airport to itself, but we couldn't express that with STRIPS
- Warning: still don't represent the *ramification* problem in a natural way
 - do all the people and packages inside p also get flown to to ?
- Warning: still don't address the *qualification* problem
 - what unrepresented circumstances could cause an action to fail?

Oct 15, 2008

COMP 131 Lecture 10

21

Example: blocks world



$On(b, x)$ – block b is on x (another block or table)
 $Move(b, x, y)$ – action of moving block b from x to y
 $Clear(x)$ – there is nothing on x

Action($Move(b, x, y)$,
 PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y)$,
 EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)$)

what about moving a block to the table?

Oct 15, 2008

COMP 131 Lecture 10

22

Planning in blocks world

Init($On(A, Table) \wedge On(B, Table) \wedge On(C, Table)$
 $\wedge Block(A) \wedge Block(B) \wedge Block(C)$
 $\wedge Clear(A) \wedge Clear(B) \wedge Clear(C)$)

Goal($On(A, B) \wedge On(B, C)$)

Action($Move(b, x, y)$,
 PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b)$
 $\wedge \neg(b=x) \wedge \neg(b=y) \wedge \neg(x=y)$,
 EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)$)

Action($MoveToTable(b, x)$,
 PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge \neg(b=x)$,
 EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x)$)

Solution sequence: [$Move(B, Table, C), Move(A, Table, B)$]

Oct 15, 2008

COMP 131 Lecture 10

23

Planning with state-space search

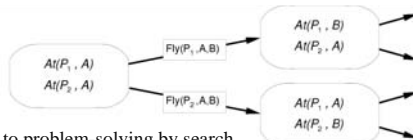
- Forward search (from initial state)
- Backward search (from goal)

Oct 15, 2008

COMP 131 Lecture 10

24

Forward search (progression)



- Similar to problem-solving by search
- Initial state of search = initial state of world
- Successor-states: generated by effects of applicable actions
- Goal test: check if planning goal satisfied
- Step cost: typically 1 for each action
- No functions = finite state space; graph search algorithms complete (e.g., A^*)

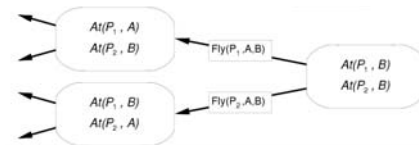
Oct 15, 2008

COMP 131 Lecture 10

25

Backward search (regression)

- STRIPS makes it easy to generate **predecessor** states
- Based on action **relevance**
 - an action is relevant to a goal if it achieves one of the goal's conjuncts (subgoals)



Oct 15, 2008

COMP 131 Lecture 10

26

Backward search example

- 10-airport cargo problem with 20 pieces of cargo
 - goal: all 20 pieces in airport B
 - $At(C1,B) \wedge At(C2,B) \wedge \dots \wedge At(C20,B)$
- which action has this effect? $Unload(C1,p,B)$, where plane p unspecified
 - only one relevant action – lower branching factor
- Predecessor-state generation:
 - $Unload(C1,p,B)$ has preconditions $In(C1,p) \wedge At(p,B)$
 - subgoal $At(C1,B)$ should not be true in predecessor state
 - therefore, predecessor state description is: $In(C1,p) \wedge At(p,B) \wedge At(C2,B) \wedge \dots \wedge At(C20,B)$

Oct 15, 2008

COMP 131 Lecture 10

27

Predecessor generation cont.

- Applicable actions should achieve desired literals (subgoals), but also *not undo* any desired literals
 - consider **consistent** actions only
 - in cargo example, action $Load(C2,p)$ is not consistent with the goal because its effect negates the literal $At(C2,B)$
- Let A be an action that is relevant and consistent. Construct the predecessor for current goal G :
 - delete any positive effects of A that appear in G
 - add each precondition literal of A , unless it already appears
- Terminate when predecessor = initial state

Oct 15, 2008

COMP 131 Lecture 10

28

Heuristics for state-space search

- Forward and backward search inefficient as is
 - Heuristic function = estimate of distance to goal
 - In planning, true distance $h^* =$ number of actions
 - finding exact number is NP-hard
 - How to construct heuristic?
 - divide-and-conquer
 - (assume subgoal independence)
 - derive a **relaxed problem** (easy to solve and generates h)
- can be optimistic
or pessimistic (not admissible!)
if subplans generate
redundant actions...

Oct 15, 2008

COMP 131 Lecture 10

29

Relaxed problems

- A problem where there are *no preconditions* to all actions
 - every subgoal is achieved with one step, if there's an action that makes it true
 - Combine with independent subgoals assumption gives $h =$ number of unsatisfied goals so far
 - Set cover heuristic (considering subgoal interaction):
 - Relax problem further by *removing negative effects*
 - Count minimum number of actions required to satisfy goal
- $Goal(A \wedge B \wedge C)$
 $Action(X, EFFECT: A \wedge P)$, minimal set cover for goal is given by actions $\{X, Y\}$
 $Action(Y, EFFECT: B \wedge C \wedge Q)$, cover heuristic returns $h=2$
 $Action(Z, EFFECT: B \wedge P \wedge Q)$

Oct 15, 2008

COMP 131 Lecture 10

30

Partial-order planning

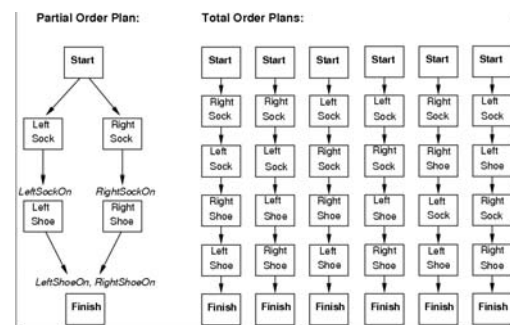
- Forward and backward search give **totally ordered plans**
 - strictly linear sequences of actions
 - cannot take advantage of problem decomposition
- Instead, work on independent subgoals independently
 - focus on important subgoals first
 - Can place two actions into a plan without specifying which one comes first (**partial-order**)

Oct 15, 2008

COMP 131 Lecture 10

31

Example



Oct 15, 2008

linearization results in totally ordered plans

POP (partial order planning)

- Search the space of partial plans
 - Start with empty plan {}
 - Actions make up plan steps. {} has only *Start* and *Finish* for actions
- Ordering constraints:
 - in the form of $A < B$ read as “A before B”
 - not necessarily immediately before
 - cycles are contradictions: no $A < B$ and $B < A$
- Causal links between actions:
 - A achieves p for B : $A \dashv\dashv B$
 - means p is the effect of action A and a precondition for action B and must remain true from the time of action A to the time of action B
 - plan may not be extended to an action that conflicts with the causal link (makes p untrue)
- Set of **open preconditions** – those not achieved by some action in the plan
 - planner works to reduce this set

Oct 15, 2008

COMP 131 Lecture 10

33

POP cont.

- A consistent plan (no cycles in ordering and no conflicts with causal links) with no open preconditions left is a **solution**
- Every linearization of a partial-order solution is a total-order solution

Oct 15, 2008

COMP 131 Lecture 10

34

POP algorithm

1. Initial plan contains *Start* and *Finish*, ordering constraint $Start < Finish$, no causal links, all preconditions open
2. Successor function arbitrarily picks an open precondition p on action B , generates successor plan for every consistent way of choosing A that achieves p for B
 1. Add causal link “A achieves p for B ” and ordering constraint “ $A < B$ ” to the plan. Add action A to the plan, as well as $Start < A$ and $A < Finish$, if A new action
 2. Resolve any conflicts between A , B and C by making C occur outside the protection interval (either add $B < C$ or add $C < A$)
 3. Add successor states for either or both if they result in consistent paths (no cycles, no conflicts)
3. Goal test checks if a plan is a solution to the original problem (i.e., no open preconditions)

Oct 15, 2008

COMP 131 Lecture 10

35

Heuristics for partial-order planning

- Advantage: decompose problems into subproblems
- Disadvantage: states not represented directly, harder to estimate cost
- H: number of distinct open preconditions left
 - will overestimate cost when actions achieve multiple subgoals
- Which precondition to choose? **Most-constrained-variable (CSP)**
 - select precondition satisfied in the fewest number of ways
 - early detection of impossibility (no ways to satisfy precondition)

Oct 15, 2008

COMP 131 Lecture 10

36

Summary

- Planners are problem-solvers operating on explicit logical representations of states and actions
- STRIPS language describes actions in terms of preconditions and effects, and start and goal states as conjunctions of positive literals. ADL is a relaxed version.
- State-space search can operate in the forward (progression) or backward (regression) direction.
- Effective heuristics make a subgoal independence assumption (but watch out for pessimism!) and relax the problem
- Partial-order planning (POP) algorithms work back from goal without committing to a totally ordered action sequence.

Oct 15, 2008

COMP 131 Lecture 10

37

Announcements

- A4 due Friday
- A5 out today, due Wed, Oct 22nd
- Next week: reasoning with uncertainty

Oct 15, 2008

COMP 131 Lecture 10

38