Lecture 20 of 42

Introduction to Classical Planning: STRIPS & Partial-Order Planning (POP)

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:

Section 11.3, p. 387 – 394, Russell & Norvig 2nd edition
Partial plan: http://en.wikipedia.org/wiki/Partial_plan

Lecture Outline

- Reading for Next Class: Section 11.3 (p. 387 – 394), R&N 2e
- Last Class: Knowledge Representation Concluded; Midterm Review
  - Inheritance semantics
  - Midterm exam emphasis
    - Rational intelligent agents: reflex, reflex/state, goals, preferences
    - Search: heuristic, constraint, game tree
    - Knowledge representation and inference: logic, resolution; FC/BC, $L_{SAT}{}^{C}$
- Today: Classical Planning, Sections 11.1 – 11.2 (p. 375 – 386), R&N 2e
  - Planning problem defined
    - Initial conditions
    - Actions: preconditions, postconditions
    - Goal conditions / goal test
  - Limitations of situation calculus and FOL
  - STRIPS operators: represent actions with preconditions, ADD/DELETE lists
- Coming Week: Midterm; More Classical and Robust Planning

Planning in Situation Calculus

$PlanResult(p, s)$ is the situation resulting from executing p in s

$$PlanResult([], s) = s$$
$$PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Initial state $At(Home, S_0) \wedge \neg Have(Milk, S_0) \wedge \ldots$

Actions as Successor State axioms

$Have(Milk, Result(a, s)) \Leftrightarrow$
$[(a = Buy(Milk) \wedge At(Supermarket, s)) \vee (Have(Milk, s) \wedge a \neq \ldots)]$

Query

$s = PlanResult(p, S_0) \wedge At(Home, s) \wedge Have(Milk, s) \wedge \ldots$

Solution

$p = [Go(Supermarket), Buy(Milk), Buy(Bananas), Go(HWS), \ldots]$

Principal difficulty: unconstrained branching, hard to apply heuristics

Making Plans using FOL: Review

Initial condition in KB:

$At(Agent, [1,1], S_0)$
$At(Gold, [1,2], S_0)$

Query: $\text{Ask}(KB, \exists s \ \ Holding(Gold, s))$
i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$
i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Making Plans – Better Way: Review

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing p in s

Then the query $\text{Ask}(KB, \exists p\ \ Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$$\forall s\ \ PlanResult([\,], s) = s$$
$$\forall a, p, s\ \ PlanResult([a|p], s) = PlanResult(p, Result(a, s))$$

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

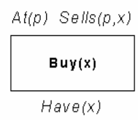
STRIPS Operators

Tidily arranged actions descriptions, restricted language

Action: $Buy(x)$
Precondition: $At(p), Sells(p, x)$
Effect: $Have(x)$

[Note: this abstracts away many important details!]

Restricted language $\Rightarrow$ efficient algorithm
Precondition: conjunction of positive literals
Effect: conjunction of literals



$Action(Fly(p, from, to),$
$\text{Precond:} At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
$\text{Effect:} \neg At(p, from) \wedge At(p, to))$

State Space *versus* Plan Space

Standard search: node = concrete world state
Planning search: node = partial plan

Defn: open condition is a precondition of a step not yet fulfilled

Operators on partial plans:

- add a link from an existing action to an open condition
- add a step to fulfill an open condition
- order one step wrt another

Gradually move from incomplete/vague plans to complete, correct plans

Air Cargo Transport Problem
STRIPS Specification

$Init(At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge At(P_2, JFK)$
$\quad \wedge\ Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2)$
$\quad \wedge\ Airport(JFK) \wedge Airport(SFO))$
$Goal(At(C_1, JFK) \wedge At(C_2, SFO))$
$Action(Load(c, p, a),$
$\quad$ PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
$\quad$ EFFECT: $\neg At(c, a) \wedge In(c, p))$
$Action(Unload(c, p, a),$
$\quad$ PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$
$\quad$ EFFECT: $At(c, a) \wedge \neg In(c, p))$
$Action(Fly(p, from, to),$
$\quad$ PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$
$\quad$ EFFECT: $\neg At(p, from) \wedge At(p, to))$

Figure 11.2
p. 380 R&N 2e

STRIPS and Its Limitations: Need for Richer Planning Language

- **What STRIPS *Can* Represent**
  - States
  - Goals
  - Actions (using action schema)
    - ⇨ <u>Preconditions</u>: must be true before action can be applied
    - ⇨ <u>Effects</u>: asserted afterwards

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

- **Real STRIPS: <u>ADD</u>, <u>DELETE</u> Lists for Operators**
- **<u>STRIPS Assumption</u>**
  - Representational frame problem solution
  - Default is that conditions remain unchanged unless mentioned in effect
- **What STRIPS *Cannot* Represent**
  - Negated preconditions
  - Inequality constraints
- **Richer Planning Language: <u>A</u>ction <u>D</u>escription <u>L</u>anguage (ADL)**

$Action(Fly(p : Plane, from : Airport, to : Airport),$
$\quad \text{PRECOND:} At(p, from) \wedge (from \neq to)$
$\quad \text{EFFECT:} \neg At(p, from) \wedge At(p, to))\ .$

STRIPS *vs.* <u>A</u>ction <u>D</u>escription <u>L</u>anguage (ADL)

STRIPS Language	ADL Language
Only positive literals in states: $Poor \wedge Unknown$	Positive and negative literals in states: $\neg Rich \wedge \neg Famous$
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: $Rich \wedge Famous$	Quantified variables in goals: $\exists x\, At(P_1, x) \wedge At(P_2, x)$ is the goal of having P_1 and P_2 in the same place.
Goals are conjunctions: $Rich \wedge Famous$	Goals allow conjunction and disjunction: $\neg Poor \wedge (Famous \vee Smart)$
Effects are conjunctions.	Conditional effects allowed: **when** P: E means E is an effect only if P is satisfied.
No support for equality.	Equality predicate $(x = y)$ is built in.
No support for types.	Variables can have types, as in $(p : Plane)$.

Figure 11.1
p. 379 R&N 2e

Simple Spare Tire Problem [1]: Illustrated Example

On(x) ~Flat(x)

START

FINISH

~Flat(Spare) Intact(Spare) Off(Spare)
On(Tire1) Flat(Tire1)

On(x)
Remove(x)
Off(x) ClearHub

Off(x) ClearHub
Puton(x)
On(x) ~ClearHub

Intact(x) Flat(x)
Inflate(x)
~Flat(x)

Simple Spare Tire Problem [2]: ADL Specification

$Init(At(Flat, Axle) \wedge At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(Spare, Trunk),$
  PRECOND: $At(Spare, Trunk)$
  EFFECT: $\neg At(Spare, Trunk) \wedge At(Spare, Ground))$
$Action(Remove(Flat, Axle),$
  PRECOND: $At(Flat, Axle)$
  EFFECT: $\neg At(Flat, Axle) \wedge At(Flat, Ground))$
$Action(PutOn(Spare, Axle),$
  PRECOND: $At(Spare, Ground) \wedge \neg At(Flat, Axle)$
  EFFECT: $\neg At(Spare, Ground) \wedge At(Spare, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk)$
  $\wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle))$

Figure 11.3
p. 381 R&N 2e

Blocks World: Three-Block Tower Problem

$Init(On(A, Table) \wedge On(B, Table) \wedge On(C, Table)$
$\wedge\ Block(A) \wedge Block(B) \wedge Block(C)$
$\wedge\ Clear(A) \wedge Clear(B) \wedge Clear(C))$
$Goal(On(A, B) \wedge On(B, C))$
$Action(Move(b, x, y),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge$
$(b \neq x) \wedge (b \neq y) \wedge (x \neq y),$
EFFECT: $On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$
$Action(MoveToTable(b, x),$
PRECOND: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$
EFFECT: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$

Figure 11.4
p. 383 R&N 2e

Forward (Progression) *vs.* Backward (Regression) State Space Search

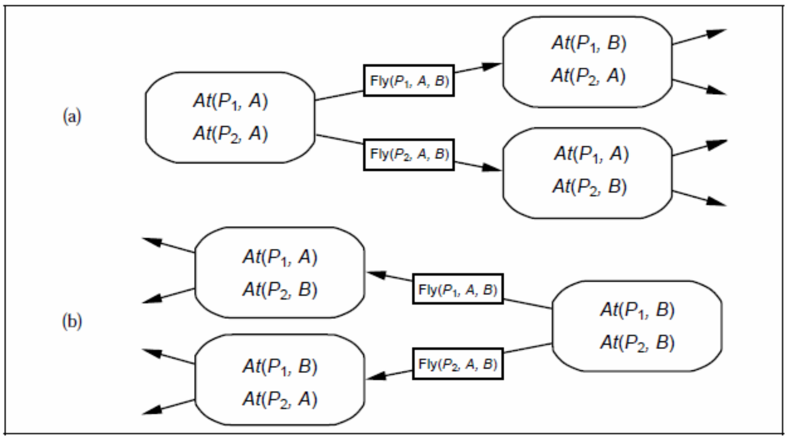
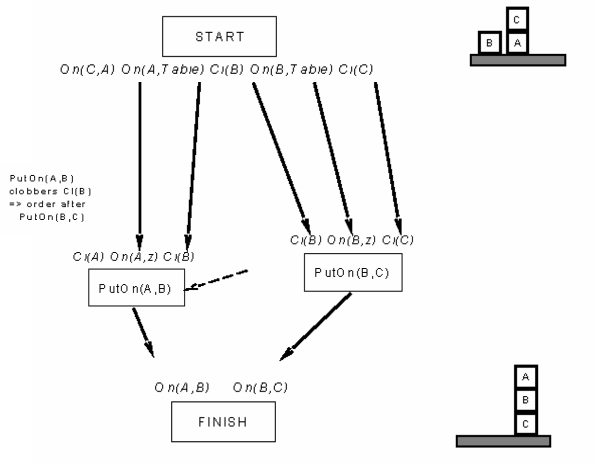


Figure 11.5
p. 383 R&N 2e

Failure of Non-Interleaved Planning: Sussman Anomaly in Blocks World

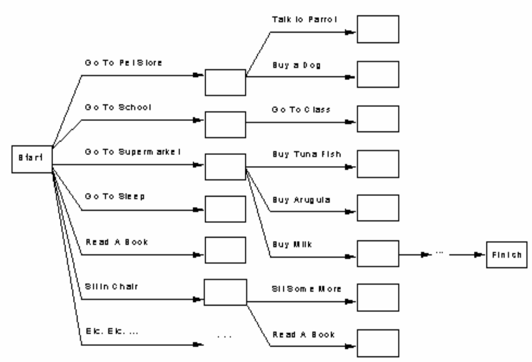


Adapted from slides © 2004 S. Russell & P. Norvig. Reused with permission.

Search *versus* Planning: State Space Search

Consider the task *get milk, bananas, and a cordless drill*

Standard search algorithms seem to fail miserably:



After-the-fact heuristic/goal test inadequate

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Partial Order Planning (POP) [1]: Total Order Plans & Interleavings

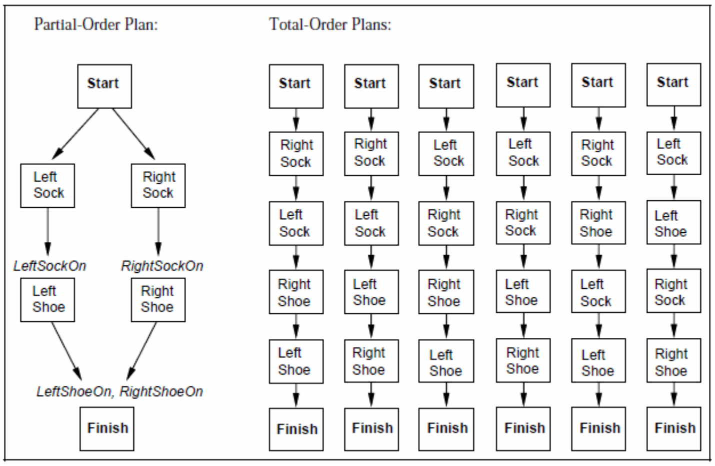
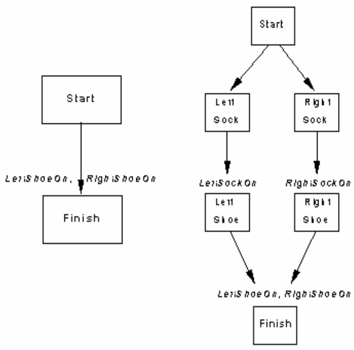


Figure 11.6
p. 389 R&N 2e

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Partial Order Planning (POP) [2]: Definition – Complete Plans



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

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

POP ALGORITHM [1]: TOP-LEVEL FUNCTIONS

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

POP ALGORITHM [2]: LOWER-LEVEL FUNCTIONS & PROPERTIES

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 $S_{add} \xrightarrow{c} S_{need}$ to LINKS(*plan*)
add the ordering constraint $S_{add} \prec S_{need}$ to ORDERINGS(*plan*)
if S_{add} is a newly added step from *operators* **then**
  add S_{add} to STEPS(*plan*)
  add $Start \prec S_{add} \prec Finish$ to ORDERINGS(*plan*)

---

procedure RESOLVE-THREATS(*plan*)

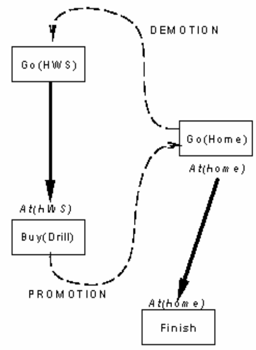
for each S_{threat} that threatens a link $S_i \xrightarrow{c} S_j$ in LINKS(*plan*) **do**
  choose either
    Demotion: Add $S_{threat} \prec S_i$ to ORDERINGS(*plan*)
    Promotion: Add $S_j \prec S_{threat}$ to ORDERINGS(*plan*)
  if not CONSISTENT(*plan*) **then fail**
end

POP is sound, complete, and systematic (no repetition)

Extensions for disjunction, universals, negation, conditionals

Clobbering and Promotion / Demotion

A clobberer is a potentially intervening step that destroys the condition achieved by a causal link. E.g., $Go(Home)$ clobbers $At(HWS)$:



Demotion: put before $Go(HWS)$

Promotion: put after $Buy(Drill)$

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Preview: How Things Go Wrong in Planning

Incomplete information

Unknown preconditions, e.g., $Intact(Spare)$?

Disjunctive effects, e.g., $Inflate(x)$ causes

$Inflated(x) \lor SlowHiss(x) \lor Burst(x) \lor BrokenPump \lor \ldots$

Incorrect information

Current state incorrect, e.g., spare NOT intact

Missing/incorrect postconditions in operators

Qualification problem:

can never finish listing all the required preconditions and possible conditional outcomes of actions

Based on slide © 2004 S. Russell & P. Norvig. Reused with permission.

Terminology

- Classical Planning – STRIPS and ADL
  - Planning problem defined
    - Initial conditions
    - Actions: preconditions, effects (postconditions)
    - Goal conditions / goal test
  - STRIPS operators: action specifications
  - ADL operators: allow negated preconditions, unequality
- Partial-Order Planning
  - Represent multiple possible interleavings
  - Keep track of which ones are achievable
  - Complete plans
    - Every precondition achieved,
    - No clobberings by possibly intervening steps
- Sussman Anomaly
  - Contains threat that needs to be resolved to get to goal
  - Illustrates need for partial-order planning, promotion / demotion

Summary Points

- Last Class: Knowledge Representation Concluded; Midterm Review
  - Inheritance semantics
  - Midterm emphasis: intelligent agents, search, KR, resolution/unification
- Today: Classical Planning – STRIPS and ADL
  - Planning problem defined
    - Initial conditions
    - Actions: preconditions, postconditions
    - Goal conditions / goal test
  - Limitations of situation calculus and FOL
  - STRIPS operators
  - ADL operators: allow negated preconditions, unequality
- Next Time (After Exam): More Classical and Robust Planning
  - Hierarchical abstraction planning (ABSTRIPS)
  - Robust planning: sensorless, conditional, monitoring/replanning, continual
- Coming Week: Midterm; Planning Continued