



Lecture 24 of 42

Graph Planning Discussion: Exam 1 Review

1. Check in
w/instructor
@project interim rep.
2. Review model solns
for midterm exam
3. Check mailing list
- HW schedule
- lectures

Friday, 20 October 2006

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/Fall-2006/CIS730>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Reading for Next Class:

Section 11.4 – 11.7, Russell & Norvig 2nd edition



Lecture Outline

- Next Monday's Reading: Sections 11.4 – 11.7, R&N 2e
- Monday
 - * Midterm exam review: search and constraints, game tree search
 - * Planning continued
- Today: Classical Planning
 - * Graph planning: STRIPS and more, Sussman anomaly
 - * Operator definitions
 - * Threat resolution: clobbering, promotion / demotion
 - * Hierarchical planning overview
- Next Week: Practical Planning
 - * Conditional Planning
 - * Replanning
 - * Monitoring and Execution
 - * Continual Planning

max	70
max	173
μ	110.2





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

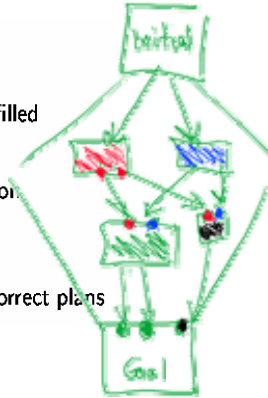
Operators on partial plans:

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



Adapted from slides by S. Russell, UC Berkeley



Successor State Axioms: Review

Successor-state axioms solve the representational frame problem

Each axiom is "about" a predicate (not an action per se):

P true afterwards \Leftrightarrow [an action made P true

\vee P true already and no action made P false]

For holding the gold:

$$\forall a, s \text{ Holding}(\text{Gold}, \text{Result}(a, s)) \Leftrightarrow$$

$$[(a = \text{Grab} \wedge \text{AtGold}(s))$$

$$\vee (\text{Holding}(\text{Gold}, s) \wedge a \neq \text{Release})]$$

Adapted from slides by S. Russell, UC Berkeley



Making Plans: A Better Way

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

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

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

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

$$\forall s \text{ PlanResult}([], s) = s$$

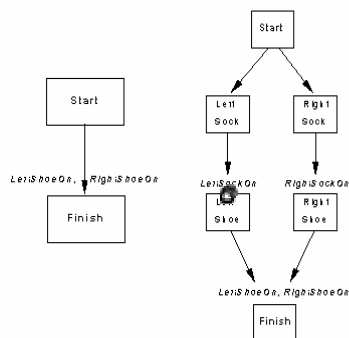
$$\forall a, p, s \text{ PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))$$

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

Adapted from slides by S. Russell, UC Berkeley



Partially-Ordered 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

Adapted from slides by S. Russell, UC Berkeley



POP Algorithm [1]: Sketch

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



Adapted from slides by S. Russell, UC Berkeley



POP Algorithm [2]: Subroutines and 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 $S_{add} \xrightarrow{c} S_j$ 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)

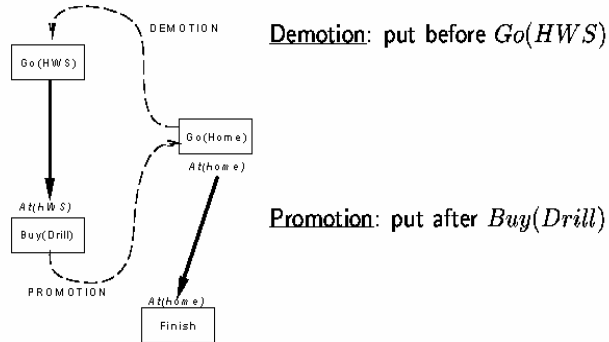
Extensions for disjunction, universals, negation, conditionals

Adapted from slides by S. Russell, UC Berkeley



Clobbering and Promotion / Demotion

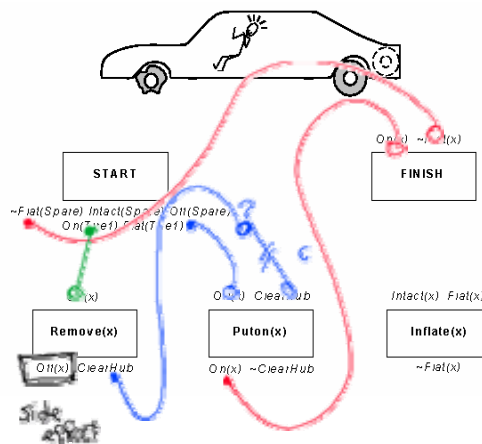
A **clobberer** is a potentially intervening step that destroys the condition achieved by a causal link. E.g., $Go(Home)$ clobbers $At(HWS)$:



Adapted from slides by S. Russell, UC Berkeley



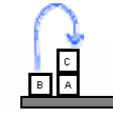
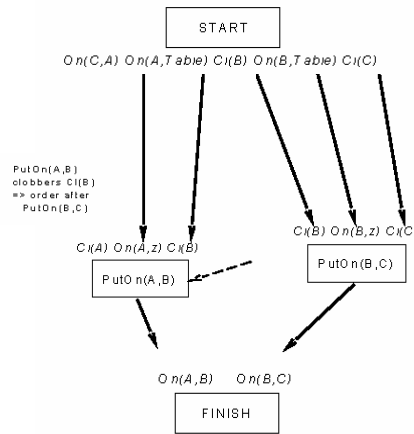
Review: Clobbering and Promotion / Demotion in Plans



Adapted from slides by S. Russell, UC Berkeley



Review: POP Example – Sussman Anomaly



Adapted from slides by S. Russell, UC Berkeley



Hierarchical Abstraction Planning

- Need for Abstraction
 - * Question: *What is wrong with uniform granularity?*
 - * Answers (among many)
 - ⇒ Representational problems
 - ⇒ Inferential problems: inefficient plan synthesis
- Family of Solutions: Abstract Planning
 - * But what to abstract in “problem environment”, “representation”?
 - ⇒ Objects, obstacles (quantification: later)
 - ⇒ Assumptions (closed world)
 - ⇒ Other entities
 - ⇒ Operators
 - ⇒ Situations
 - * Hierarchical abstraction
 - ⇒ See: Sections 12.2 – 12.3 R&N, pp. 371 – 380
 - ⇒ Figure 12.1, 12.6 (examples), 12.2 (algorithm), 12.3-5 (properties)

Adapted from Russell and Norvig



Universal Quantifiers in Planning

- Quantification *within* Operators
 - * p. 383 R&N
 - * Examples
 - ⇒ Shakey's World
 - ⇒ Blocks World
 - ⇒ Grocery shopping
 - * Others (from projects?)
- Exercise for Next Tuesday: *Blocks World*



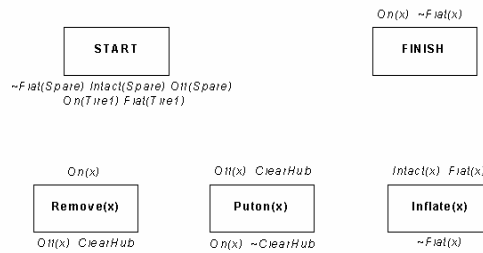
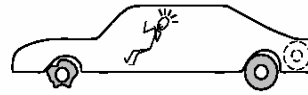
Practical Planning

- The Real World
 - * *What can go wrong with classical planning?*
 - * *What are possible solution approaches?*
- Conditional Planning
- Monitoring and Replanning (Next Time)





Review: Cluttering and Promotion / Demotion in Plans



Adapted from slides by S. Russell, UC Berkeley



Review: How Things Go Wrong in Planning

Incomplete information

Unknown preconditions, e.g., $\text{Intact}(\text{Spare})?$

Disjunctive effects, e.g., $\text{Inflate}(x)$ causes

$\text{Inflated}(x) \vee \text{SlowHiss}(x) \vee \text{Burst}(x) \vee \text{BrokenPump} \vee \dots$

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

Adapted from slides by S. Russell, UC Berkeley



Review: Practical Planning Solutions

Conditional planning

Plan to obtain information (observation actions)

Subplan for each contingency, e.g.,

$[Check(Tire1), If(Intact(Tire1), [Inflate(Tire1)], [CallAAA])]$

Expensive because it plans for many unlikely cases

Monitoring/Replanning

Assume normal states, outcomes

Check progress *during execution*, replan if necessary

Unanticipated outcomes may lead to failure (e.g., no AAA card)

In general, some monitoring is unavoidable

Adapted from slides by S. Russell, UC Berkeley



Conditional Planning

$[\dots, If(p, [then\ plan], [else\ plan]), \dots]$

Execution: check p against current KB, execute "then" or "else"

Conditional planning: just like POP except

if an open condition can be established by observation action

add the action to the plan

complete plan for each possible observation outcome

insert conditional step with these subplans

CheckTire(x)

$KnowsIf(Intact(x))$

Adapted from slides by S. Russell, UC Berkeley



Monitoring and Replanning

Execution monitoring

"failure" = preconditions of *remaining plan* not met
 preconditions = causal links at current time

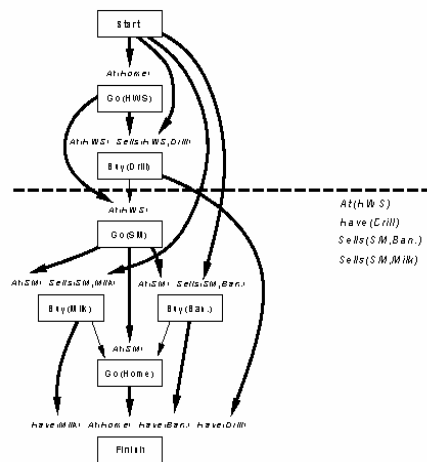
Action monitoring

"failure" = preconditions of *next action* not met
 (or action itself fails, e.g., robot bump sensor)

In both cases, need to *replan*



Preconditions for Remaining Plan



Adapted from slides by S. Russell, UC Berkeley

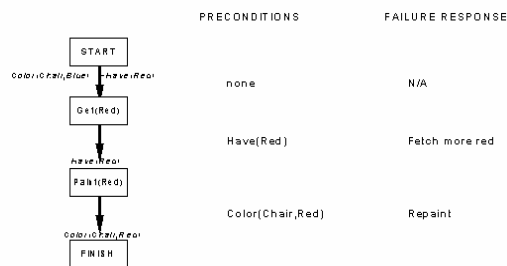
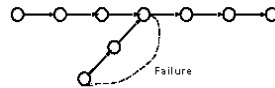




Replanning

Simplest: on failure, replan from scratch

Better: plan to get back on track by reconnecting to best continuation
Generates "loop until done" behavior with no explicit loop



Adapted from slides by S. Russell, UC Berkeley



Solutions

Conditional planning

Plan to obtain information (**observation actions**)

Subplan for each contingency, e.g.,

$[Check(Tire1), If(Intact(Tire1), [Inflate(Tire1)], [CallAAA])]$

Expensive because it plans for many unlikely cases

Monitoring/Replanning

Assume normal states, outcomes

Check progress *during execution*, replan if necessary

Unanticipated outcomes may lead to failure (e.g., no AAA card)

In general, some monitoring is unavoidable

Adapted from slides by S. Russell, UC Berkeley



Summary Points

- Previously: Logical Representations and Theorem Proving
 - * Propositional, predicate, and first-order logical languages
 - * Proof procedures: forward and backward chaining, resolution refutation
- Today: Introduction to Classical Planning
 - * Search vs. planning
 - * STRIPS axioms
 - ⇒ Operator representation
 - ⇒ Components: preconditions, postconditions (ADD, DELETE lists)
- Thursday: More Classical Planning
 - * Partial-order planning (NOAH, etc.)
 - * Limitations



Terminology

- Classical Planning
 - * Planning versus search
 - * Problematic approaches to planning
 - ⇒ Forward chaining
 - ⇒ Situation calculus
 - * Representation
 - ⇒ Initial state
 - ⇒ Goal state / test
 - ⇒ Operators
- Efficient Representations
 - * STRIPS axioms
 - ⇒ Components: preconditions, postconditions (ADD, DELETE lists)
 - ⇒ Clobbering / threatening
 - * Reactive plans and policies
 - * Markov decision processes

Adapted from slides by S. Russell, UC Berkeley

