



## Lecture 25 of 42

### HTN Planning and Robust Planning Discussion: Planning and Logic, Uncertainty

Monday, 23 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 12.1 – 12.4, Russell & Norvig 2<sup>nd</sup> edition



## Lecture Outline

- Today's Reading: Sections 11.4 – 11.7, 12.1 – 12.4, R&N 2e
- Today and Wednesday: Practical Planning
  - \* Conditional Planning
  - \* Replanning
  - \* Monitoring and Execution
  - \* Continual Planning
- Wednesday: Hierarchical Planning Revisited
  - \* Examples: Korf
  - \* Real-World Example
- Friday and Next Week: Reasoning under Uncertainty
  - \* Basics of reasoning under uncertainty
  - \* Probability review
  - \* BNJ interface (<http://bnj.sourceforge.net>)
  - \* Graphical models problems
  - \* Algorithms





## POP Algorithm [1]: Review

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

```

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

```

**function** SELECT-SUBGOAL(*plan*) **returns** *S<sub>need</sub>*, *c*

```

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

```

Adapted from slides by S. Russell, UC Berkeley



## POP Algorithm [2]: Subroutines and Properties Review

**procedure** CHOOSE-OPERATOR(*plan*, *operators*, *S<sub>need</sub>*, *c*)

```

choose a step Sadd from operators or STEPS(plan) that has c as an effect
if there is no such step then fail
add the causal link Sadd → c → Sj to LINKS(plan)
add the ordering constraint Sadd < Sneed to ORDERINGS(plan)
if Sadd is a newly added step from operators then
  add Sadd to STEPS(plan)
  add Start < Sadd < Finish to ORDERINGS(plan)

```

**procedure** RESOLVE-THREATS(*plan*)

```

for each Sthreat that threatens a link Si → Sj in LINKS(plan) do
  choose either
    Demotion: Add Sthreat < Si to ORDERINGS(plan)
    Promotion: Add Sj < Sthreat to ORDERINGS(plan)
  if not CONSISTENT(plan) then fail
end

```



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

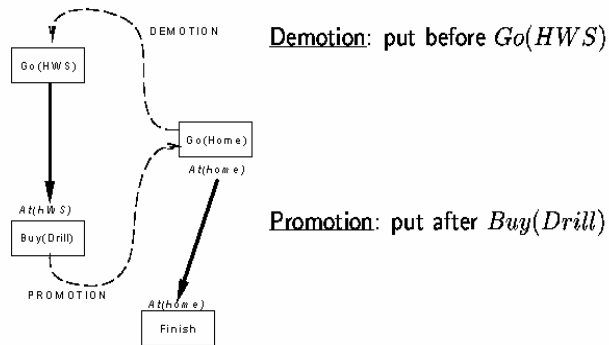
Extensions for disjunction, universals, negation, conditionals

Adapted from slides by S. Russell, UC Berkeley



## Clobbering and Promotion / Demotion

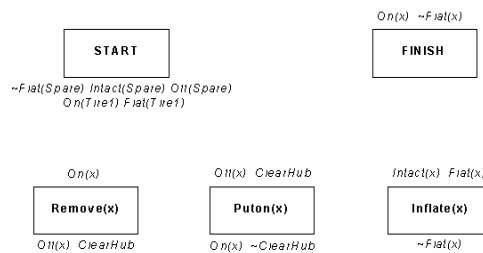
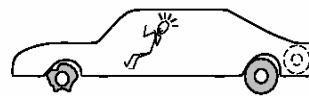
A **clobberer** is a potentially intervening step that destroys the condition achieved by a causal link. E.g.,  $Go(Home)$  clobbers  $At(HWS)$ :



Adapted from slides by S. Russell, UC Berkeley



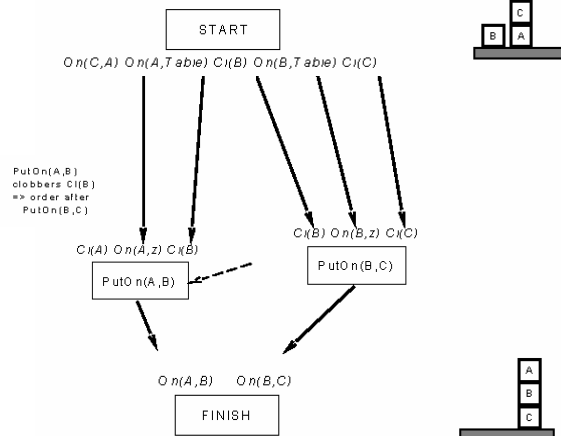
## Review: Clobbering and Promotion / Demotion in Plans



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)

Too big/coarse; not robust; can't recover from fail.  
 Too high level  
 computationally expensive  
 - combinatorial explosion  
 Adapting to prob-solving stages (nodes)  
 Human in the loop

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*



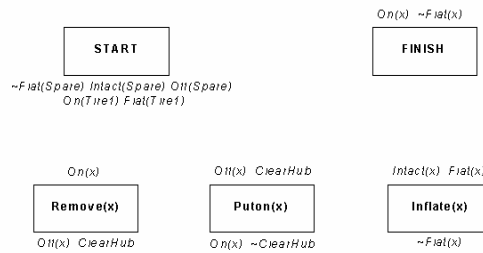
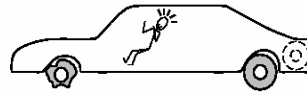
## Practical Planning

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





## Review: Robbering 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

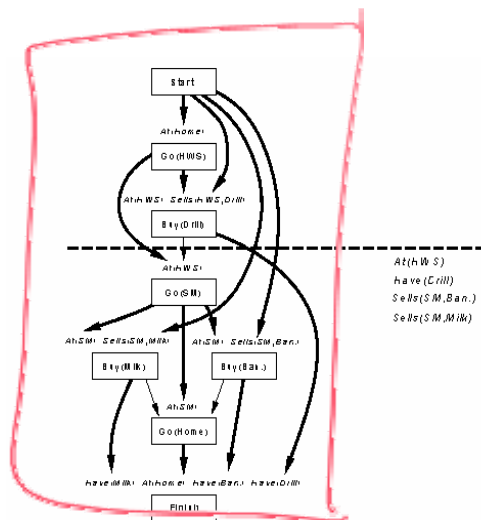
### Action monitoring

"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

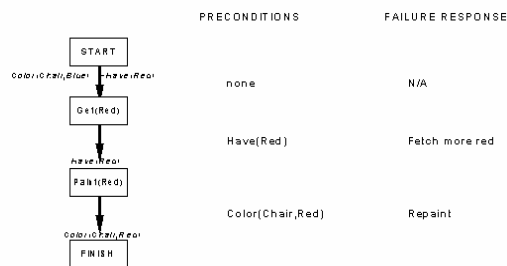
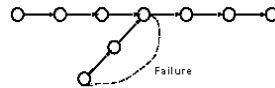




## Replanning

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