



Lecture 25 of 42

HTN Planning and Robust Planning Discussion: Midterm Exam Review

Prob. or known diff. (e.g. class) L_d = L_H Next prob. For - SAT

Wednesday, 17 October 2007



$L_d \subseteq L_H$
 $\therefore L_H \in \mathcal{R}/\mathcal{C}$

William H. Hsu

Department of Computing and Information Sciences, KSU

DELETE (A1G)
ADD (A1 G)
Go (from 24)

KSOL course page: <http://snipurl.com/v9v3>

Course web site: <http://www.kddresearch.org/Courses/Fall-2007/CIS730>

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

Reading for Next Class:

Section 12.1 – 12.4, Russell & Norvig 2nd edition



Lecture Outline

- Today's Reading: Sections 11.4 – 11.7, 12.1 – 12.4, R&N 2e
- Today and Next Monday: Practical Planning
 - * Conditional Planning
 - * Replanning
 - * Monitoring and Execution
 - * Continual Planning
- Next Monday: Hierarchical Planning Revisited
 - * Examples: Korf
 - * Real-World Example
- Later 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*

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 Review

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



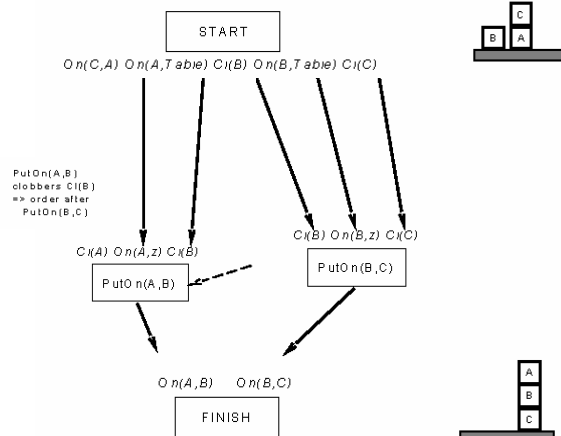
POP is sound, complete, and systematic (no repetition)

Extensions for disjunction, universals, negation, conditionals

Adapted from slides by S. Russell, UC Berkeley



Review: POP Example – Sussman Anomaly

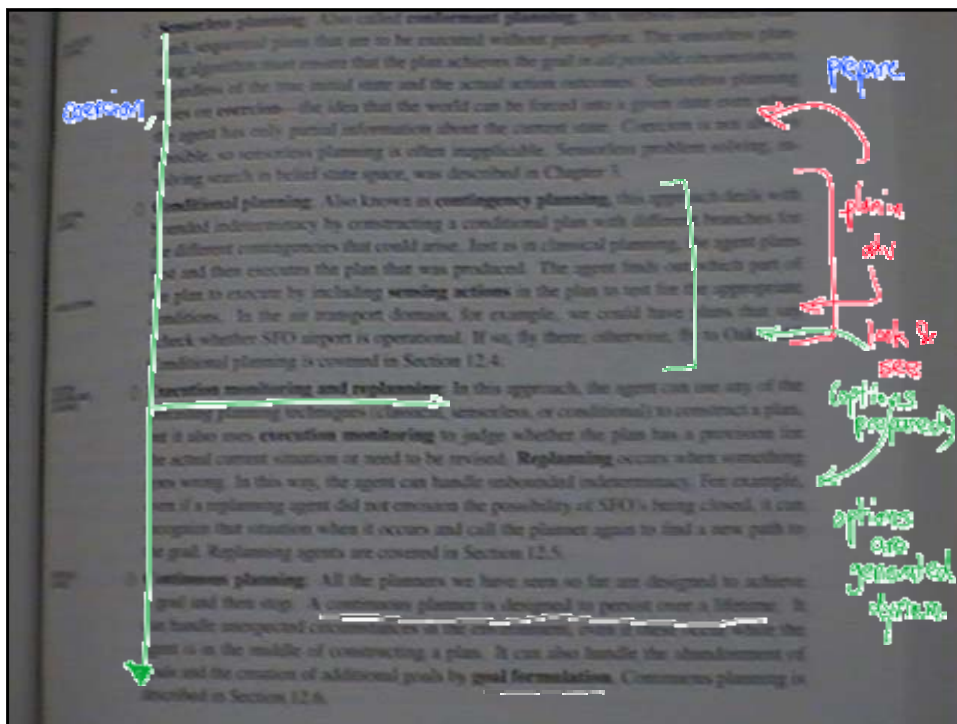
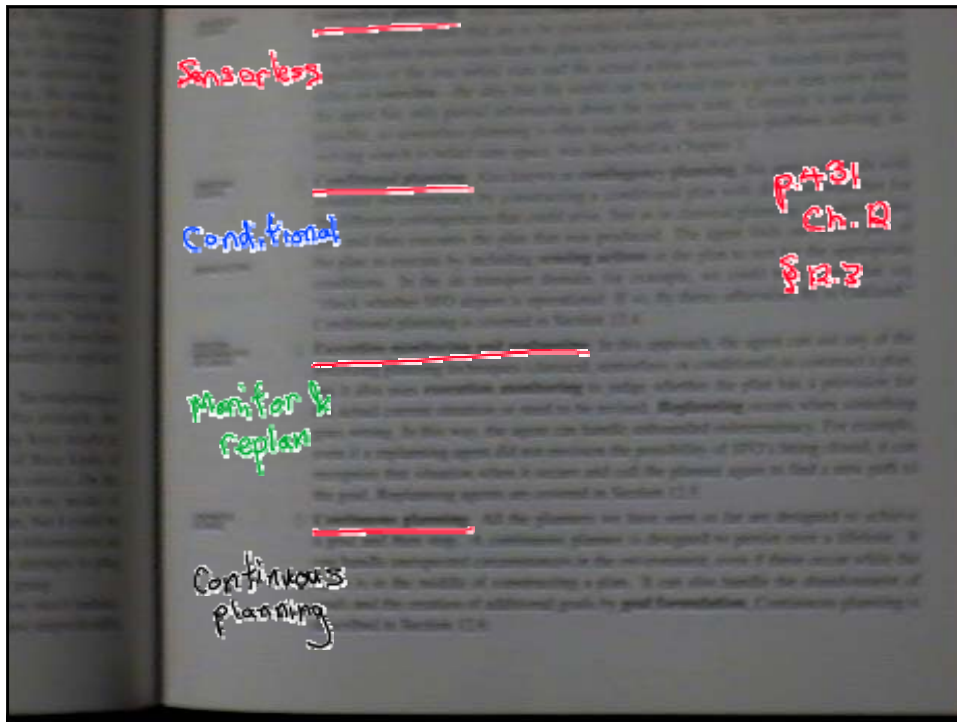


Adapted from slides by S. Russell, UC Berkeley



Planning and Learning Roadmap

- **Bounded Indeterminacy (12.3)**
- **Four Techniques for Dealing with Nondeterministic Domains**
- **1. Sensorless / Conformant Planning: “Be Prepared” (12.3)**
 - * Idea: be able to respond to any situation (universal planning)
 - * Coercion
- **2. Conditional / Contingency Planning: “Plan B” (12.4)**
 - * Idea: be able to respond to many typical alternative situations
 - * Actions for sensing (“reviewing the situation”)
- **3. Execution Monitoring / Replanning: “Show Must Go On” (12.5)**
 - * Idea: be able to resume momentarily failed plans
 - * Plan revision
- **4. Continuous Planning: “Always in Motion, The Future Is” (12.6)**
 - * Lifetime planning (and learning!)
 - * Formulate new goals





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 Wednesday: *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)**

Adapted from Russell and Norvig



Review: How Things Go Wrong in Planning

Incomplete information

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

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

$Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \dots$

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

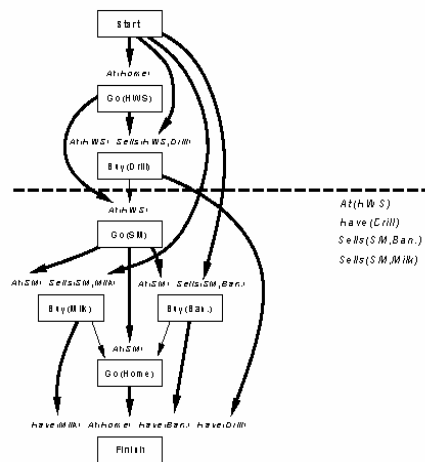
"failure" = preconditions of *next action* not met
 (or action itself fails, e.g., robot bump sensor)

In both cases, need to *replan*

Adapted from slides by S. Russell, UC Berkeley



Preconditions for Remaining Plan



Adapted from slides by S. Russell, UC Berkeley

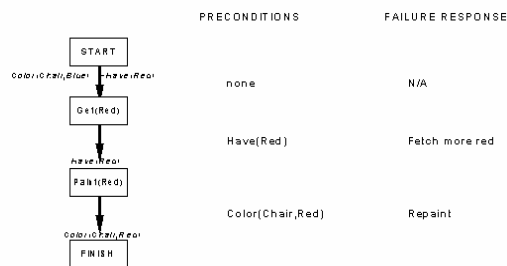
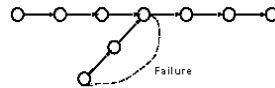




Replanning

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)
- **Next Week: More Classical Planning**
 - * Partial-order planning (NOAH, etc.)
 - * Limitations

Adapted from slides by S. Russell, UC Berkeley



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

