



## Lecture 19 of 42

### More Description Logics, Classical Planning Discussion: Midterm Exam Review

Monday, 08 October 2007

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-2007/CIS730>

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

Reading for Next Class:  
Sections 10.4 – 10.6, Russell & Norvig 2<sup>nd</sup> edition.  
*(Handwritten notes: domain-dep, operators in planning, postulates from 10.6)*



## Lecture Outline

- Today's Reading: Sections 10.4 – 10.6, R&N 2e
- Wednesday's Reading: Sections 10.7 – 10.9, R&N 2e
- Friday: Knowledge Rep, Ontologies, Situational Calculus
- Today
  - \* Temporal logic
  - \* Semantic networks
  - \* Description Logics
- Next Week
  - \* Description Logics
  - \* Defeasible reasoning: nonmonotonic logic
  - \* Intro to Planning
- Midterm Exam: 16 Oct 2006
  - \* Remote students: have exam agreement faxed to DCE
  - \* Exam will be faxed to proctors Wednesday or Friday



## First-Order Logic: Summary

Literal:  $P(x, y)$  <sup>or more args</sup>

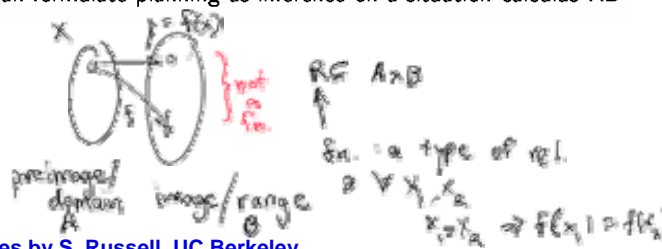
First-order logic:

- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers

Increased expressive power: sufficient to define wumpus world

Situation calculus:

- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB

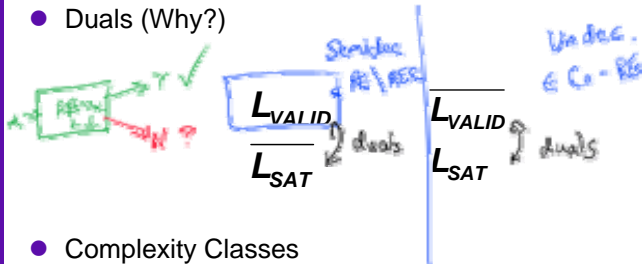


Adapted from slides by S. Russell, UC Berkeley



## Decidability Revisited

- See: Section 9.7 Sidebar, p. 288 R&N
- Duals (Why?)



- Complexity Classes

- Understand: Reduction to  $L_d, L_H$



## 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

CIS 530 / 730: Artificial Intelligence

Monday, 08 Oct 2007

Computing & Information Sciences  
Kansas State University



## Describing Actions [1]: Frame, Qualification, Ramification Problems

"Effect" axiom—describe changes due to action

$\forall s \text{ AtGold}(s) \Rightarrow \text{Holding}(\text{Gold}, \text{Result}(\text{Grab}, s))$

"Frame" axiom—describe non-changes due to action

$\forall s \text{ HaveArrow}(s) \Rightarrow \text{HaveArrow}(\text{Result}(\text{Grab}, s))$

Frame problem: find an elegant way to handle non-change

→ (a) representation—avoid frame axioms

→ (b) inference—avoid repeated "copy-overs" to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or ...

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, ...

Adapted from slides by S. Russell, UC Berkeley

CIS 530 / 730: Artificial Intelligence

Monday, 08 Oct 2007

Computing & Information Sciences  
Kansas State University



## Describing Actions [2]: Successor State Axioms

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



## Classical Planning: Review

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

$\text{PlanResult}(p, s)$  is the result of executing  $p$  in  $s$

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

Definition of  $\text{PlanResult}$  in terms of  $\text{Result}$ :

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

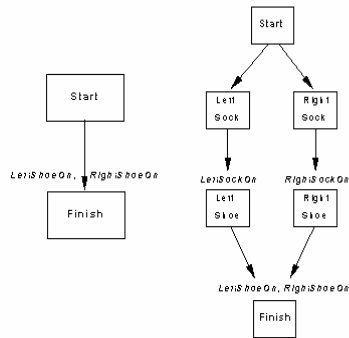
$$\forall a, p, s \text{ PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))$$

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



## Lecture Outline

- **Today's Reading**
  - \* Sections 11.5 – 11.9, Russell and Norvig
  - \* References: to be posted on class web board
- **Next Week's Reading: Chapter 12, Russell and Norvig**
- **Previously: Logical Representations and Theorem Proving**
- **Today: More Classical Planning**
  - \* STRIPS axioms (review)
  - \* Partial-order planning (NOAH, etc.)
  - \* Limitations of POP
    - ⇒ Need for abstraction
    - ⇒ Hierarchical abstraction (ABSTRIPS)
- **First Hour Exam: Friday 19 Oct 2007, in class**
  - \* Two pages of notes allowed
  - \* Remote students: have exam agreement faxed to DCE
  - \* Exam will be faxed to proctors Friday morning
- **Next Week: More Planning – Conditional and Reactive**



## 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
    Sneed, c ← SELECT-SUBGOAL(plan)
    CHOOSE-OPERATOR(plan, operators, Sneed, c)
    RESOLVE-THREATS(plan)
  end
  
```



```

function SELECT-SUBGOAL(plan) returns Sneed, 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

```

procedure CHOOSE-OPERATOR(plan, operators, Sneed, 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-> Sneed 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 -c-> 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)

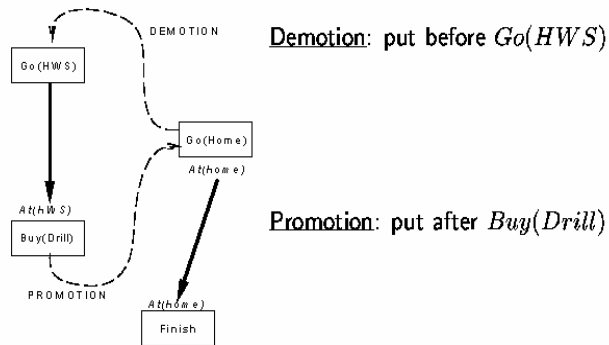
Extensions for disjunction, universals, negation, conditionals

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  $At(HWS)$ :

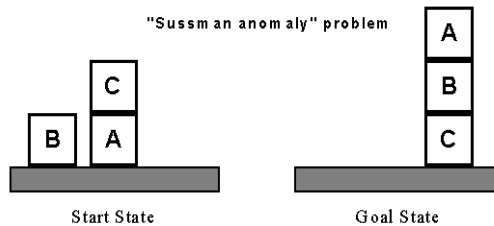


Adapted from slides by S. Russell, UC Berkeley



## Example: Blocks World [1] Specification

"Sussman anomaly" problem



$Clear(x) \ On(x,z) \ Clear(y)$

PutOn(x,y)

$\sim On(x,z) \ \sim Clear(y)$   
 $Clear(z) \ On(x,y)$

+ several inequality constraints

$Clear(x) \ On(x,z)$

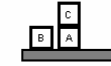
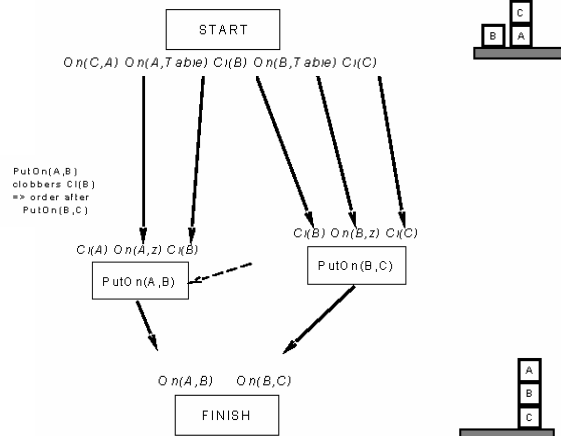
PutOnTable(x)

$\sim On(x,z) \ Clear(z) \ On(x,Table)$

Adapted from slides by S. Russell, UC Berkeley



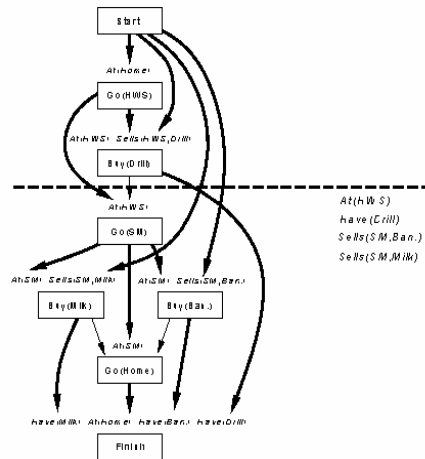
## Example: Blocks World [2] POP Trace



Adapted from slides by S. Russell, UC Berkeley



## Example: Preconditions for Remaining Plan



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 slides by S. Russell, UC Berkeley



## Universal Quantifiers in Planning

- **Quantification *within* Operators**
  - \* Chapter 11, R&N 2e
  - \* **Examples**
    - ⇒ Shakey's World
    - ⇒ Blocks World (R&N; also in Winston, Rich and Knight)
    - ⇒ Grocery shopping
  - \* **Others (from projects?)**
- **Exercise for Next Tuesday: *Blocks World***





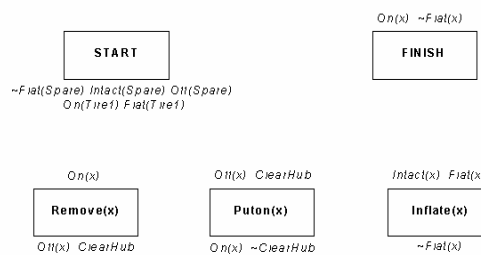
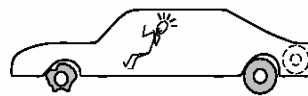
## Practical Planning

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

Adapted from Russell and Norvig



## Cobbling and Promotion / Demotion in Plans



Adapted from slides by S. Russell, UC Berkeley



## Things Go Wrong

### 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



## 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

- **Today: Introduction to Classical Planning**
  - \* **Search vs. planning**
  - \* **STRIPS axioms**
    - ⇒ Operator representation
    - ⇒ Components: preconditions, postconditions (ADD, DELETE lists)
- **Wednesday: More Classical Planning**
  - \* **Partial-order planning (NOAH, etc.)**
    - ⇒ Old terminology (*deprecated*): “linear” vs. “non-linear”
    - ⇒ Modern terminology (*preferred*): “partial-order (POP)” vs. “non-POP”
  - \* **Limitations of POP**
    - ⇒ Haven’t considered conditionals yet (qualification problem revisited)
    - ⇒ Frame problems: representational, inferential; circumscription issues



## Terminology

- **Classical Planning Framework**
  - \* **Planning versus search**
  - \* **Representation: initial state, goal state / test, operators**
- **STRIPS Operators**
  - \* **Components: preconditions, postconditions (ADD, DELETE lists)**
  - \* **STRIPS and interference**
    - ⇒ Clobbering / threatening
    - ⇒ Promotion / demotion
  - \* **Partial-Order Planners (POP systems)**
- **This Week**
  - \* **Hierarchical abstraction planning: ABSTRIPS**
  - \* **Conditional plans**
  - \* **Reactive plans and policies**
  - \* **Markov decision processes**

Adapted from slides by S. Russell, UC Berkeley

