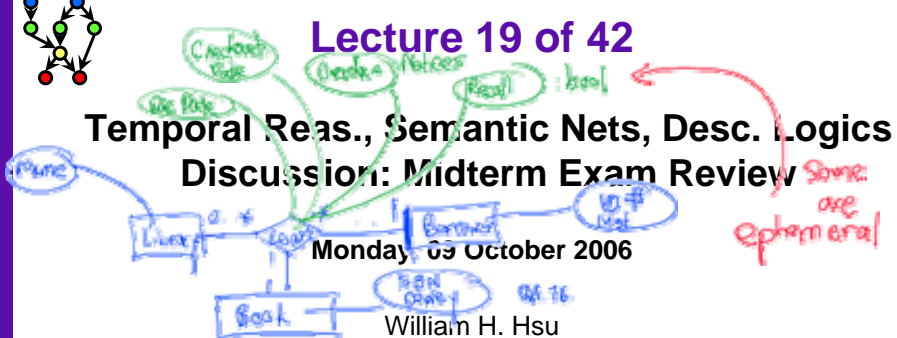




## Lecture 19 of 42

### Temporal Reas., Semantic Nets, Desc. Logics Discussion: Midterm Exam Review



Monday, 09 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>

Modal Logic: one that has operators (usually) that apply in a specialized context  
eg Temporal Logic  
- When  
- Until  
- Before (<) & Up-Until (≤)

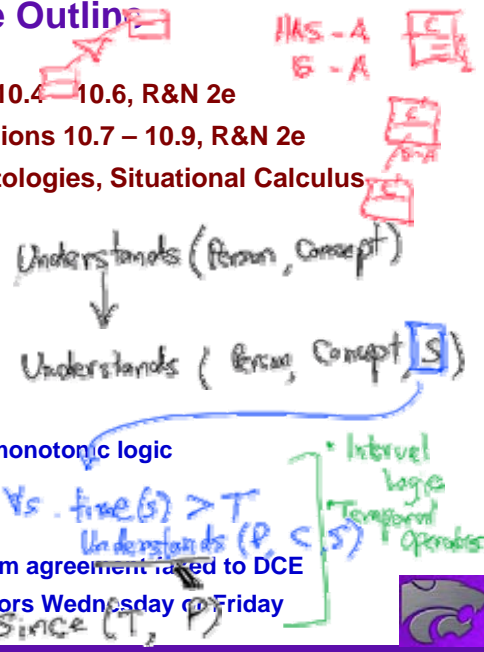
Reading for Next Class:

Sections 10.4 – 10.6, Russell & Norvig 2<sup>nd</sup> edition



## 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

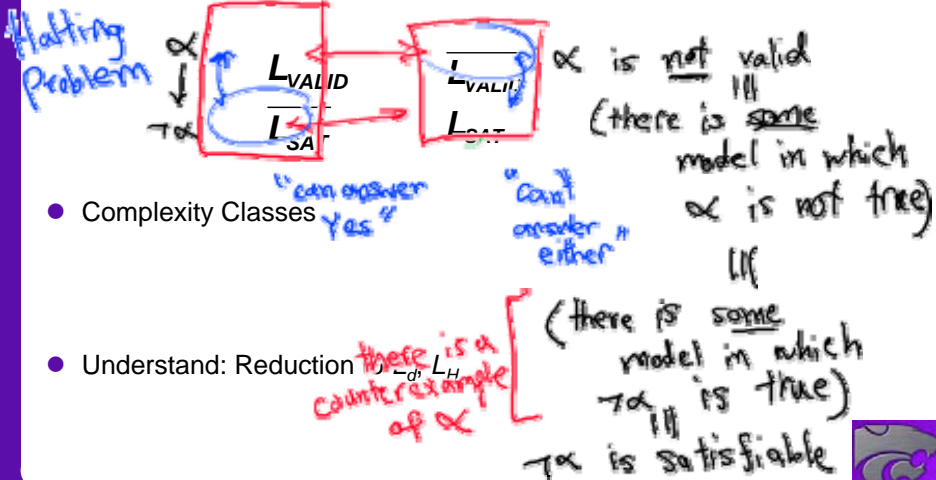




## Decidability Revisited

- See: Section 9.7 Sidebar, p. 288 R&N

- Duals (Why?) *Semi-decidable* *undecidable* *Ld*



## 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



## Describing Actions [1]: Frame, Qualification, and 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



## 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):

$$\begin{aligned} P \text{ true afterwards} &\Leftrightarrow [\text{an action made } P \text{ true} \\ &\vee P \text{ true already and no action made } P \text{ false}] \end{aligned}$$

For holding the gold:

$$\begin{aligned} \forall a, s \text{ Holding}(\text{Gold}, \text{Result}(a, s)) &\Leftrightarrow \\ &[(a = \text{Grab} \wedge \text{AtGold}(s)) \\ &\vee (\text{Holding}(\text{Gold}, s) \wedge a \neq \text{Release})] \end{aligned}$$

Adapted from slides by S. Russell, UC Berkeley



## Making Plans

Initial condition in KB:

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

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

Query:  $ASK(KB, \exists s \text{ Holding}(Gold, s))$

i.e., in what situation will I be holding the gold?

Answer:  $\{s / \text{Result}(Grab, \text{Result}(Forward, S_0))\}$

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

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



## First-Order Logic: Summary

First-order logic:

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

Increased expressive power: sufficient to define wumpus world

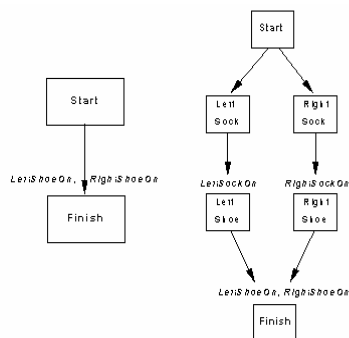
Situation calculus:

- 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



## 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)

1. Search
2. Logic
3. Resolution & Unification

- First Hour Exam: Friday 14 Oct 2005, in class

- \* Two pages of notes allowed
- \* Remote students: have exam agreement faxed to DCE
- \* Exam will be faxed to proctors Friday morning

$$\text{Adm: } \forall n. h(n) \leq h^*(n)$$

Pathmax:

Ch. 1-8



## 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)
```

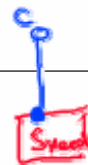
```
  end
```

```
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
```





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

procedure RESOLVE-THREATS(plan)
  for each Sthreat 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)

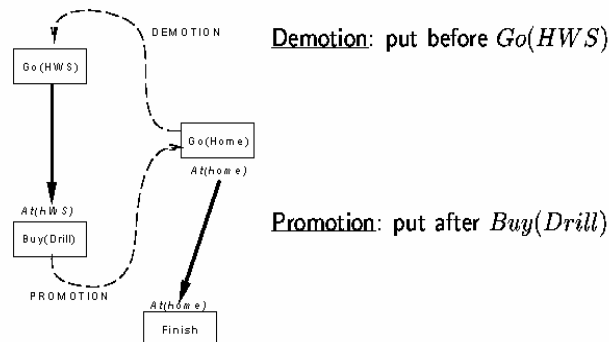
Extensions for disjunction, universals, negation, conditionals

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)*:

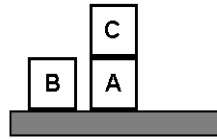


Adapted from slides by S. Russell, UC Berkeley

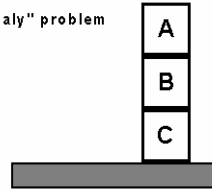


## Example: Blocks World [1] Specification

"Sussman anomaly" problem



Start State



Goal State

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

PutOn(x,y)

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

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

PutOnTable(x)

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

+ several inequality constraints

Adapted from slides by S. Russell, UC Berkeley

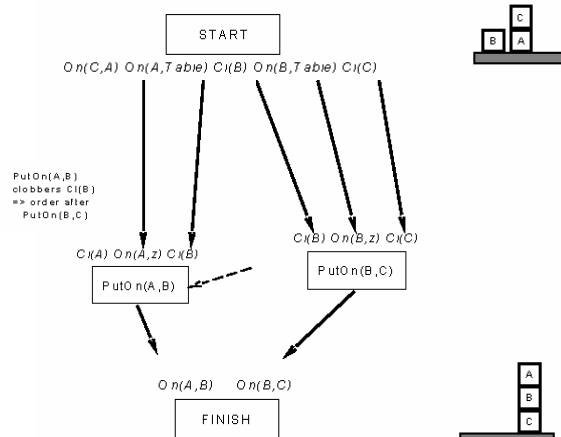
CIS 490 / 730: Artificial Intelligence

Monday, 09 Oct 2006

Computing & Information Sciences  
Kansas State University



## Example: Blocks World [2] POP Trace



Adapted from slides by S. Russell, UC Berkeley

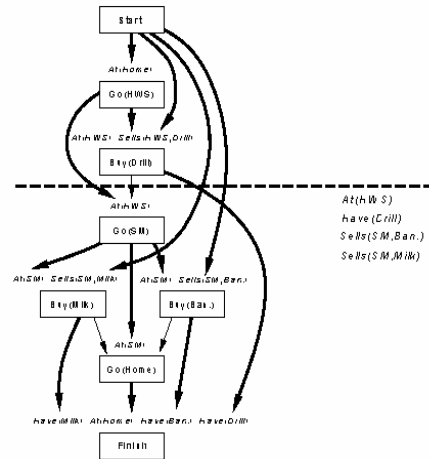
CIS 490 / 730: Artificial Intelligence

Monday, 09 Oct 2006

Computing & Information Sciences  
Kansas State University



## 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 Russell and Norvig



## 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*



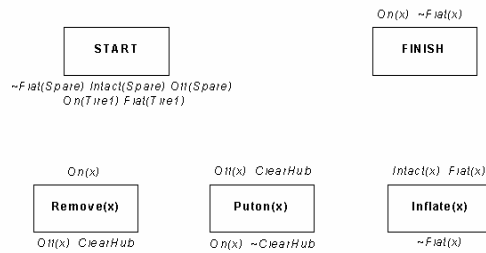
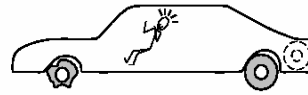
## Practical Planning

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



## Review:

### Grabbing and Promotion / Demotion in Plans



Adapted from slides by S. Russell, UC Berkeley



## Things Go Wrong

### 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



## 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

- Monday: Introduction to Classical Planning
  - \* Search vs. planning
  - \* STRIPS axioms
    - ⇒ Operator representation
    - ⇒ Components: preconditions, postconditions (ADD, DELETE lists)
- Today: 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)
- Next Week
  - \* Hierarchical abstraction planning: ABSTRIPS
  - \* Conditional plans
  - \* Reactive plans and policies
  - \* Markov decision processes

Adapted from slides by S. Russell, UC Berkeley

