LECTURE 22 OF 42

Planning: Sensorless and Conditional Planning Discussion: More Abstraction & Hierarchical Task Networks (HTN)

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/CIS730

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

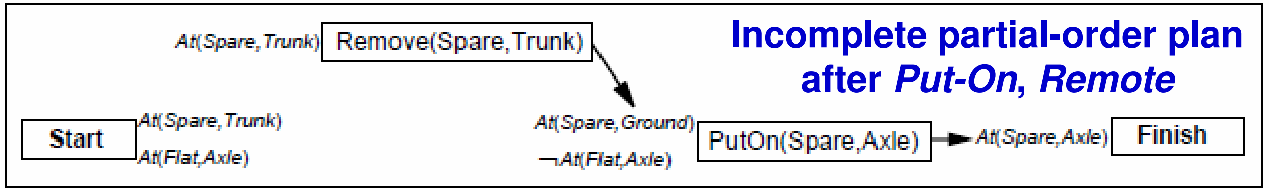
Reading for Next Class:

Section 12.1 – 12.4, p. 417– 440, Russell & Norvig 2nd edition

Lecture Outline

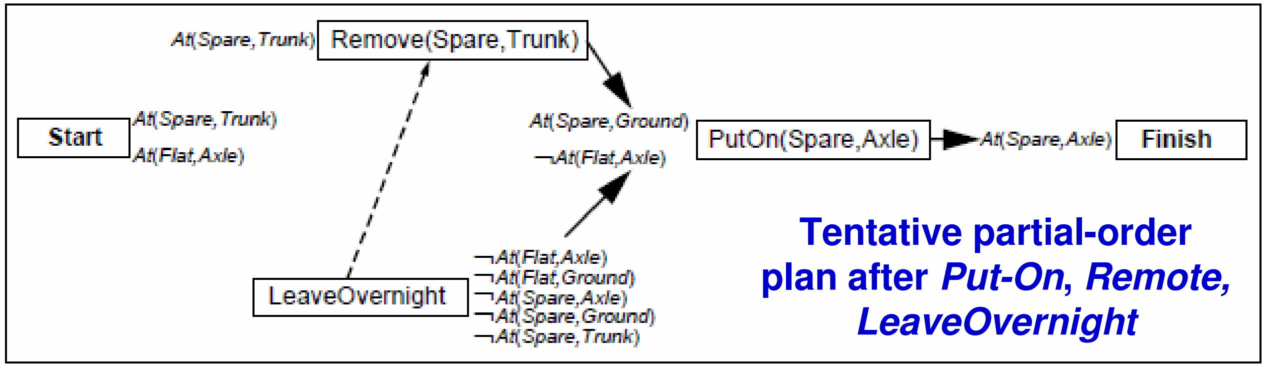
- Reading for Next Class: Sections 12.1 – 12.4 (p. 417 – 440), R&N 2e
- Last Class: Section 11.3 (p. 387 – 394), R&N 2e
  - Plan linearization
  - Extended POP example: changing spare tire
  - Graph planning
  - Hierarchical abstraction planning (ABSTRIPS)
- Today: Sections 11.4 – 11.7 (p. 395 – 408), R&N 2e
  - Graph planning (11.4)
  - Planning with propositional logic (11.5)
  - Analysis of planning approaches (11.6)
  - Summary (11.7)
- Coming Week: Robust Planning Concluded; Uncertain Reasoning
  - Practical planning: sensorless, conditional, replanning, continual (Ch. 12)
  - Uncertainty in planning
  - Need for representation language for uncertainty

Spare Tire Example – POP Trace: Review



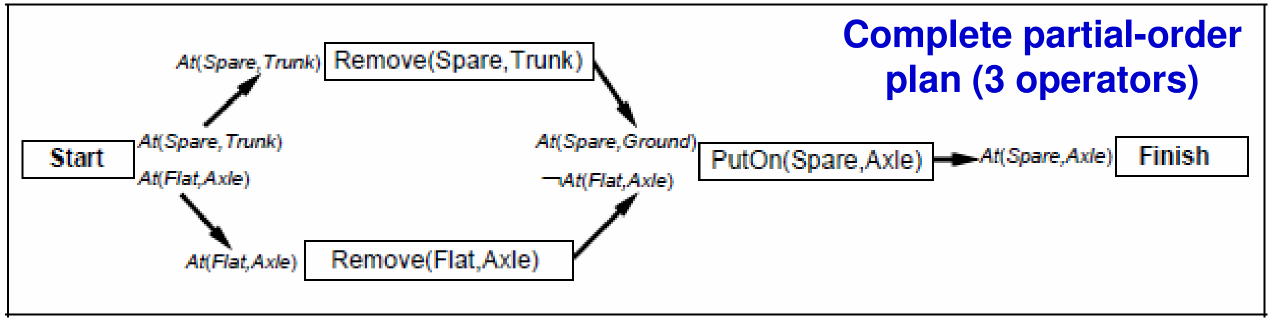
Incomplete partial-order plan after *Put-On*, *Remote*

Figure 11.8 p. 392 R&N 2e



Tentative partial-order plan after *Put-On*, *Remote*, *LeaveOvernight*

Figure 11.9 p. 392 R&N 2e



Complete partial-order plan (3 operators)

Figure 11.10 p. 393 R&N 2e

Heuristics for Classical Planning: Review

- **Problem: Combinatorial Explosion due to High Branch Factor**
  - Branch factor (main problem in planning): possible operators
  - Fan-out: many side effects
  - Fan-in: many preconditions to work on at once
- **Goal: Speed Up Planning**
- **Heuristic Design Principles**
  - Favor general ones (domain-independent)
  - Treat as goals as countable or continuous instead of boolean (true/false)
  - Use commonsense reasoning (need commonsense knowledge)
    - ⇨ Counting, weighting partially-achieved goals
    - ⇨ Way to compute preferences (utility estimates)
- **Domain-Independent *h*: Number of Unsatisfied Conjuncts**
  - e.g., Have(A) ∧ Have(B) ∧ Have(C) ∧ Have(D)
  - Have(A) ∧ Have(C): *h* = 2
- **Domain-Dependent *h*: May Be Based on Problem Structure**

Graph Planning – **Graphplan** Motivation: Review

- **Previous Heuristics for STRIPS/ADL**
  - **Domain-independent heuristics: counting parts (conjuncts) of goal satisfied**
  - **Domain-dependent heuristics: based on (many) domain properties**
    - ⇨ **problem decomposability (intermediate goals)**
    - ⇨ **reusability of solution components**
    - ⇨ **preferences**
- **Limitation: Heuristics May Not Be Accurate**
- **Objective: Better Heuristics**
  - **Need: structure that clarifies problem**
  - **Significance: faster convergence, more manageable branch factor**
- **Approach: Use Graphical Language of Constraints, Actions**
- **Notation**
  - **Operators (real actions): large rectangles**
  - **Persistence actions (for each literal): small squares, denote non-change**
  - **Gray links: mutual exclusion (mutex)**

GRAPH PLANNING – CAKE PROBLEM [1]: INITIAL CONDITIONS & GRAPH

Init(*Have*(*Cake*))
Goal(*Have*(*Cake*) $\wedge$ *Eaten*(*Cake*))
Action(*Eat*(*Cake*)
  PRECOND: *Have*(*Cake*)
  EFFECT: $\neg$ *Have*(*Cake*) $\wedge$ *Eaten*(*Cake*))
Action(*Bake*(*Cake*)
  PRECOND: $\neg$ *Have*(*Cake*)
  EFFECT: *Have*(*Cake*)

Figure 11.11
p. 396 R&N 2^e

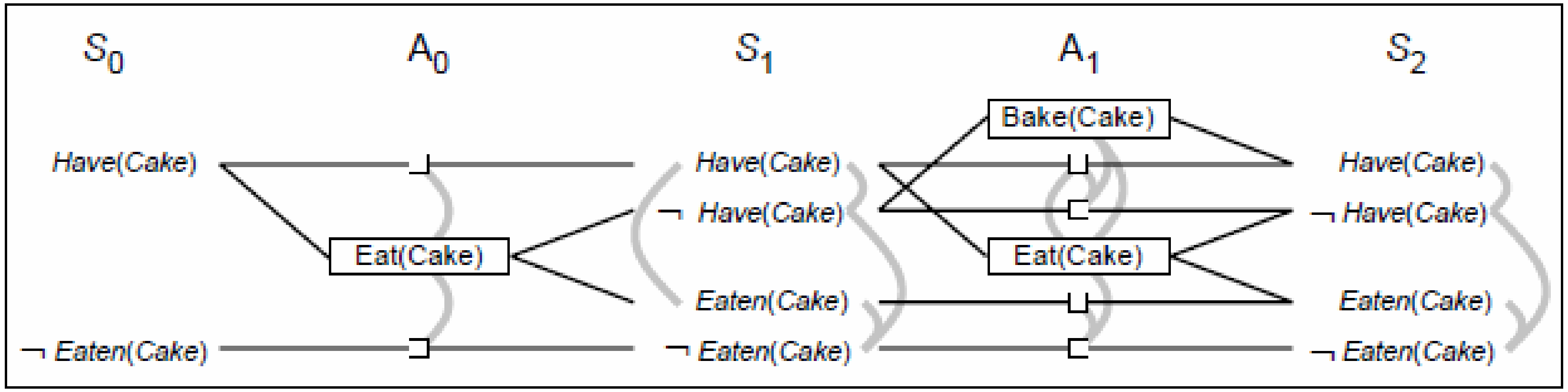
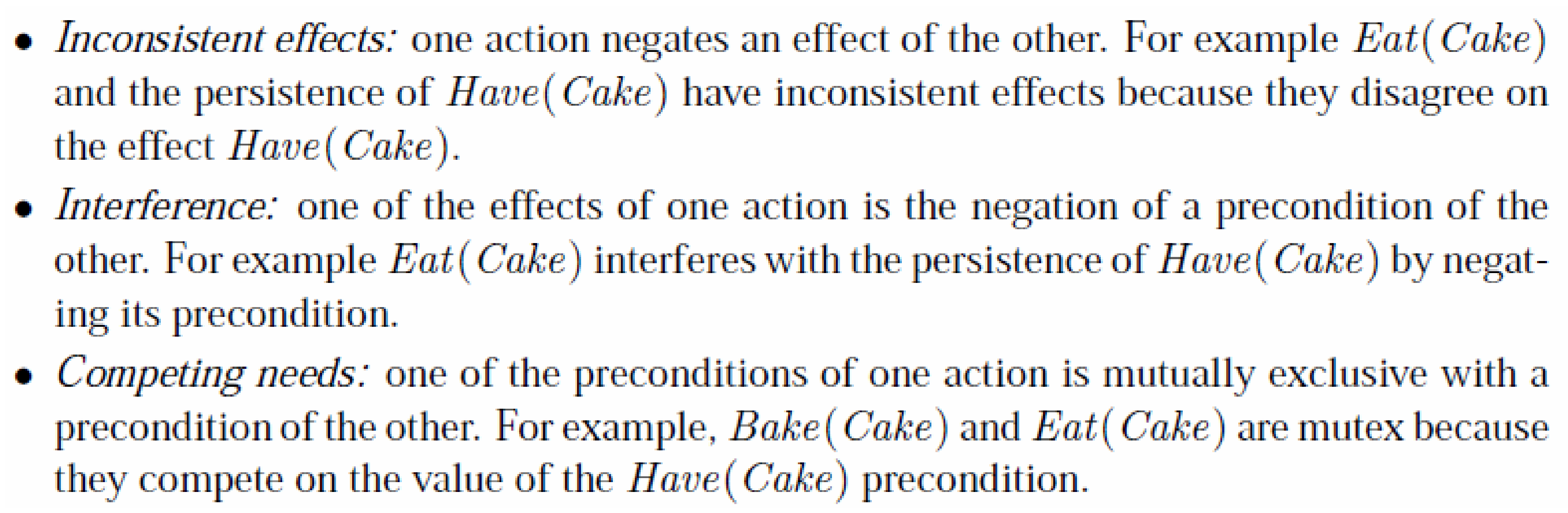


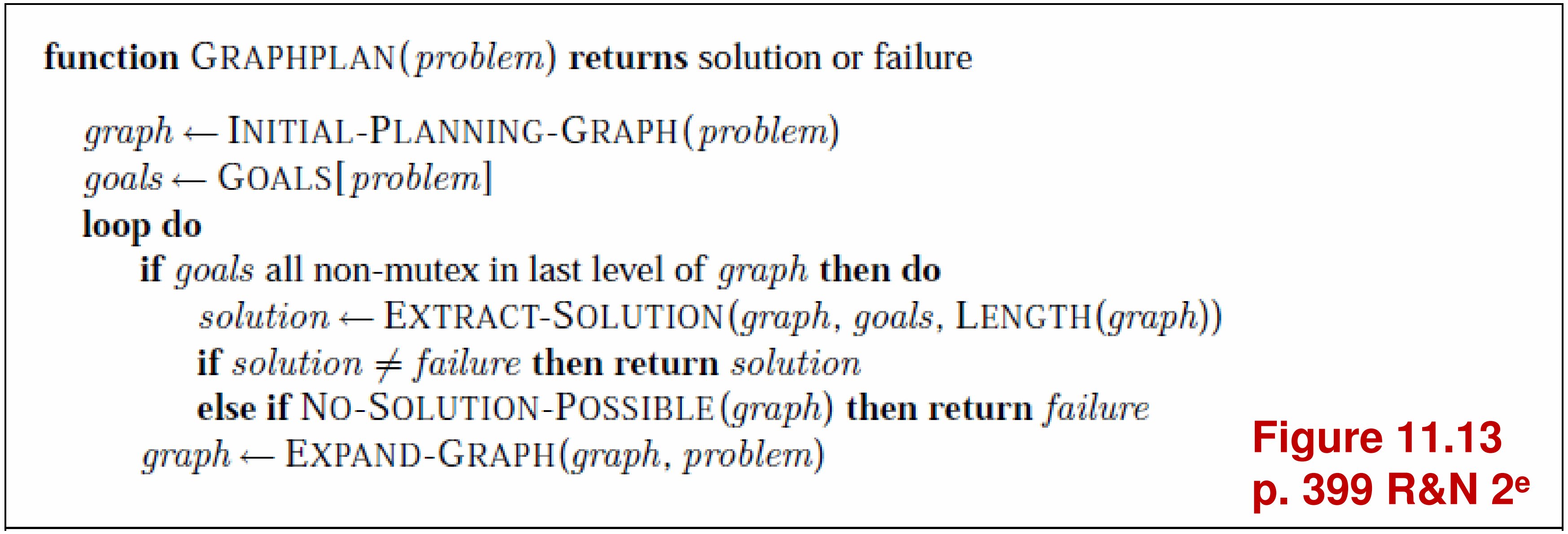
Figure 11.12
p. 396 R&N 2^e

Graph Planning – Cake Problem [2]: Mutex Conditions



- *Inconsistent effects:* one action negates an effect of the other. For example $Eat(Cake)$ and the persistence of $Have(Cake)$ have inconsistent effects because they disagree on the effect $Have(Cake)$.
- *Interference:* one of the effects of one action is the negation of a precondition of the other. For example $Eat(Cake)$ interferes with the persistence of $Have(Cake)$ by negating its precondition.
- *Competing needs:* one of the preconditions of one action is mutually exclusive with a precondition of the other. For example, $Bake(Cake)$ and $Eat(Cake)$ are mutex because they compete on the value of the $Have(Cake)$ precondition.

Graph Planning: **Graphplan** Algorithm



```
function GRAPHPLAN(problem) returns solution or failure

  graph ← INITIAL-PLANNING-GRAPH(problem)
  goals ← GOALS[problem]
  loop do
      if goals all non-mutex in last level of graph then do
          solution ← EXTRACT-SOLUTION(graph, goals, LENGTH(graph))
          if solution $\neq$ failure then return solution
          else if NO-SOLUTION-POSSIBLE(graph) then return failure
      graph ← EXPAND-GRAPH(graph, problem)
```

Figure 11.13
p. 399 R&N 2^{e}

- **Alternating Steps**
  - **Solution extraction**
  - **Expansion**
- Extract-Solution: **Goal-Based (Regression)**
- Expand-Graph: **Adds Operators, State Literals**

Graph Planning: Spare Tire Example (Expanded to S_2)

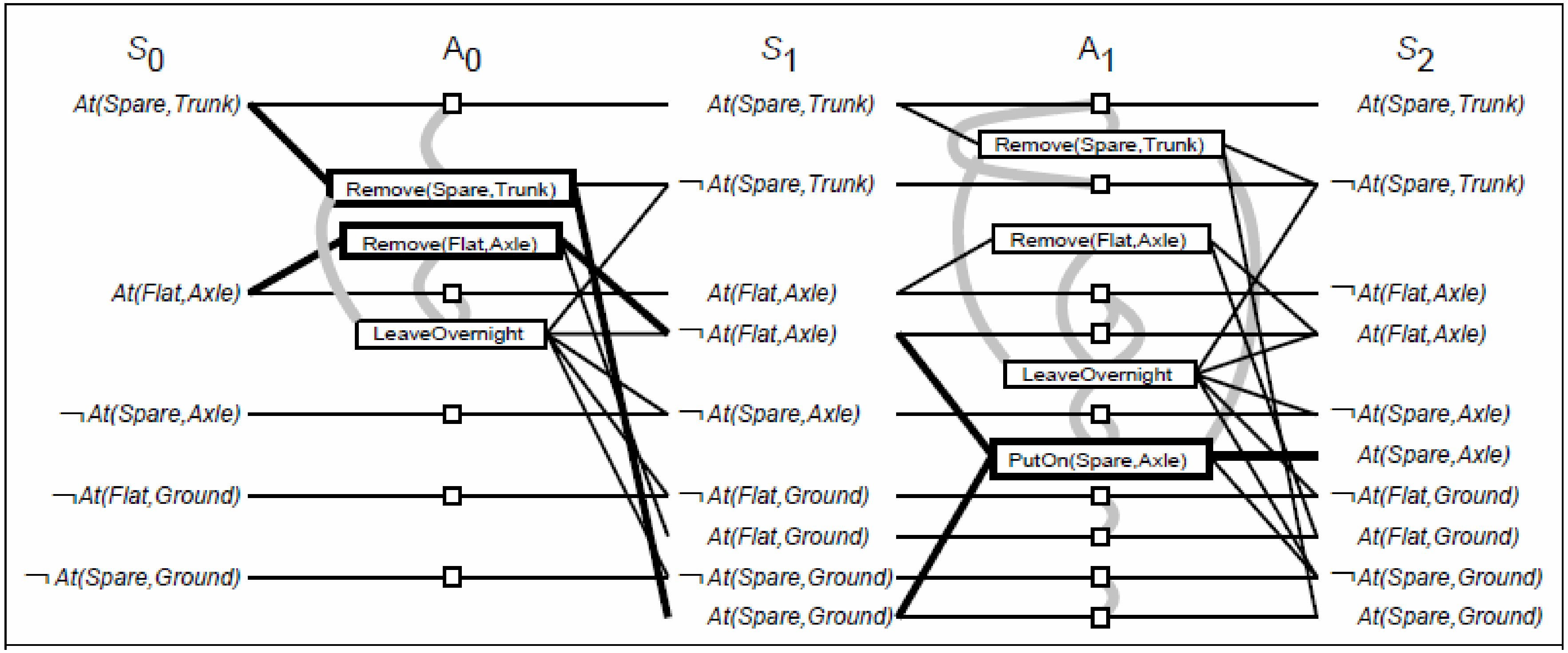
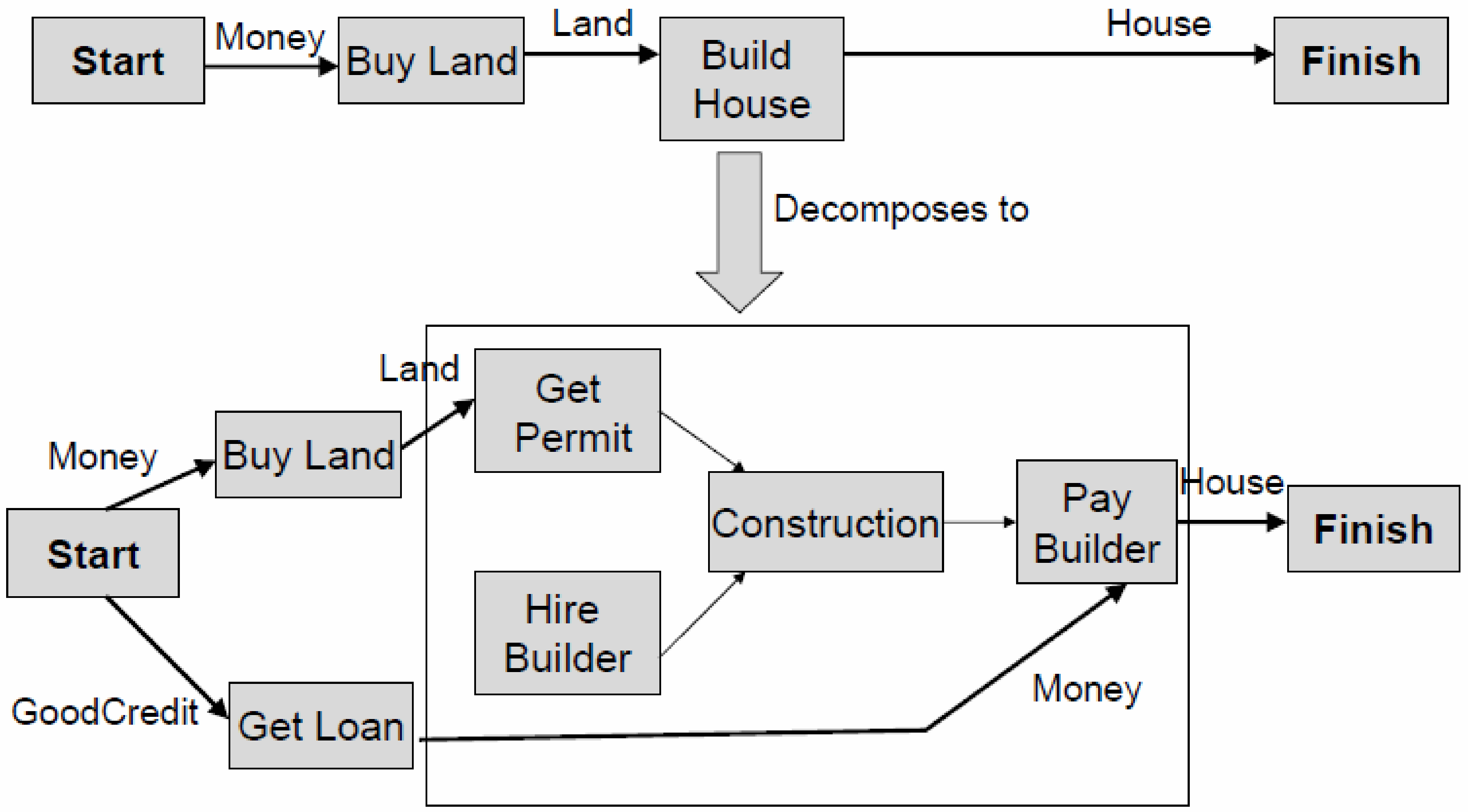


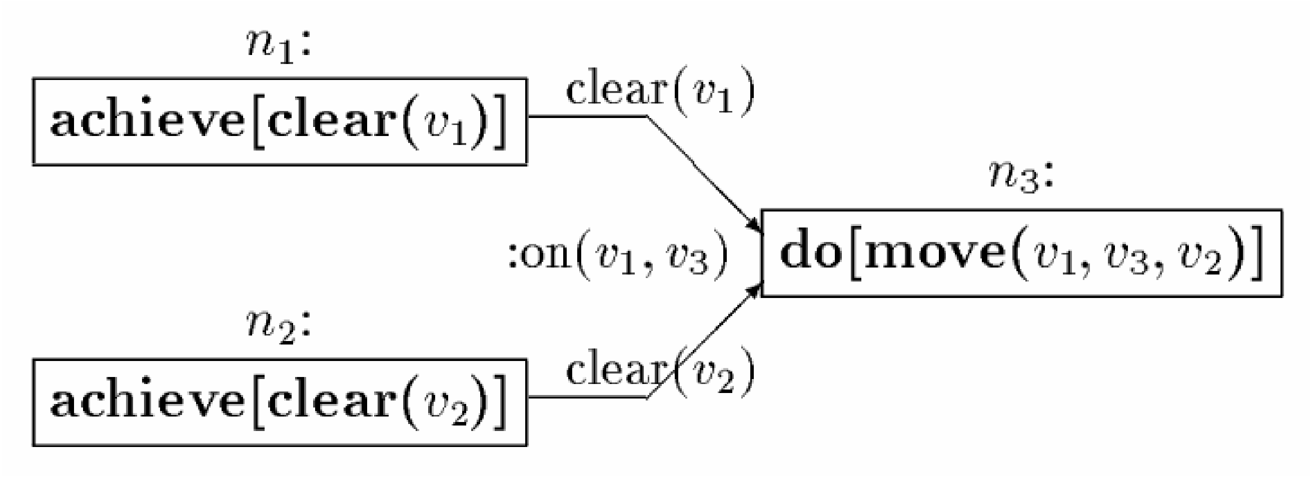
Figure 11.14
p. 399 R&N 2e

Hierarchical Abstraction – Example: Review



http://bit.ly/3nly0N

Hierarchical Task Network Planning: Review



$$[(n_1 : achieve[clear(v_1)])(n_2 : achieve[clear(v_2)])(n_3 : do[move(v_1, v_3, v_2)])$$
$$(n_1 \prec n_3) \wedge (n_2 \prec n_3) \wedge (n_1, clear(v_1), n_3) \wedge (n_2, clear(v_2), n_3) \wedge (on(v_1, v_3), n_3)$$
$$\wedge \neg(v_1 = v_2) \wedge \neg(v_1 = v_3) \wedge \neg(v_2 = v_3)]$$

Planning in Propositional Domains: Cargo Plane Example Redux

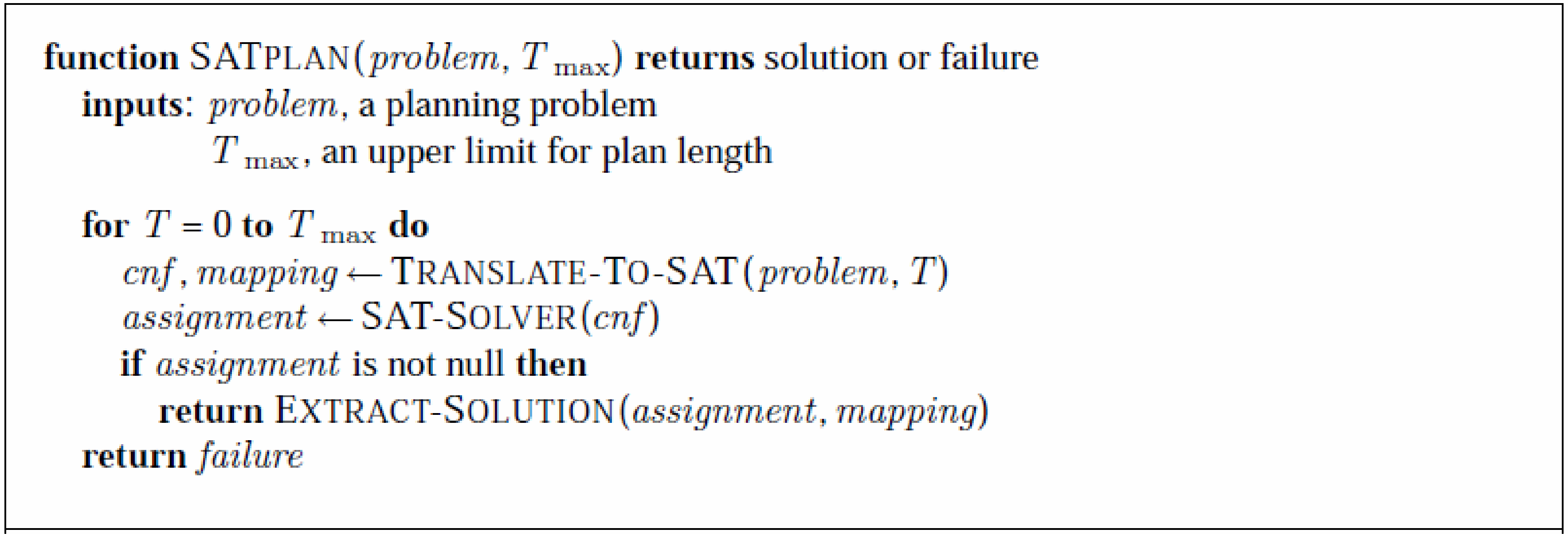
$$\textit{initial state} \wedge \textit{all possible action descriptions} \wedge \textit{goal}\,.$$

$$At(P_1, SFO)^0 \wedge At(P_2, JFK)^0\,.$$

$$\neg At(P_1, JFK)^0 \wedge \neg At(P_2, SFO)^0\,.$$

$$\begin{aligned} At(P_1, JFK)^1 \;\Leftrightarrow\; & (At(P_1, JFK)^0 \wedge \neg(Fly(P_1, JFK, SFO)^0 \wedge At(P_1, JFK)^0)) \\ & \vee (Fly(P_1, SFO, JFK)^0 \wedge At(P_1, SFO)^0)\,. \end{aligned}$$

SATPlan Algorithm



```
function SATPLAN(problem, $T_{max}$) returns solution or failure
  inputs: problem, a planning problem
          $T_{max}$, an upper limit for plan length

  for $T$ = 0 to $T_{max}$ do
    cnf, mapping ← TRANSLATE-TO-SAT(problem, $T$)
    assignment ← SAT-SOLVER(cnf)
    if assignment is not null then
      return EXTRACT-SOLUTION(assignment, mapping)
  return failure
```

Figure 11.15
p. 403 R&N 2e

Practical Planning Solutions [1]: Sensorless Planning – Preview

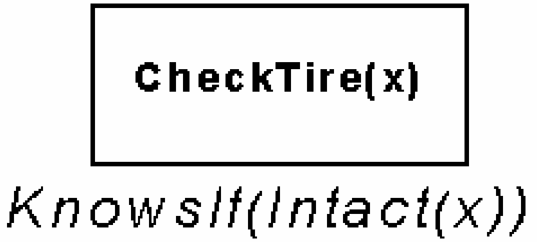
- **Problem: Bounded Indeterminacy**
  - **“How world is like”**
  - **“How it will be if I do A”**
- **Idea: Coerce State of World**
  - **Complete plan in all possible situations**
  - **Example: move forward to walk through door OR push it open**
- **Not Always Possible!**

Practical Planning Solutions [2]: Conditional Planning – Preview

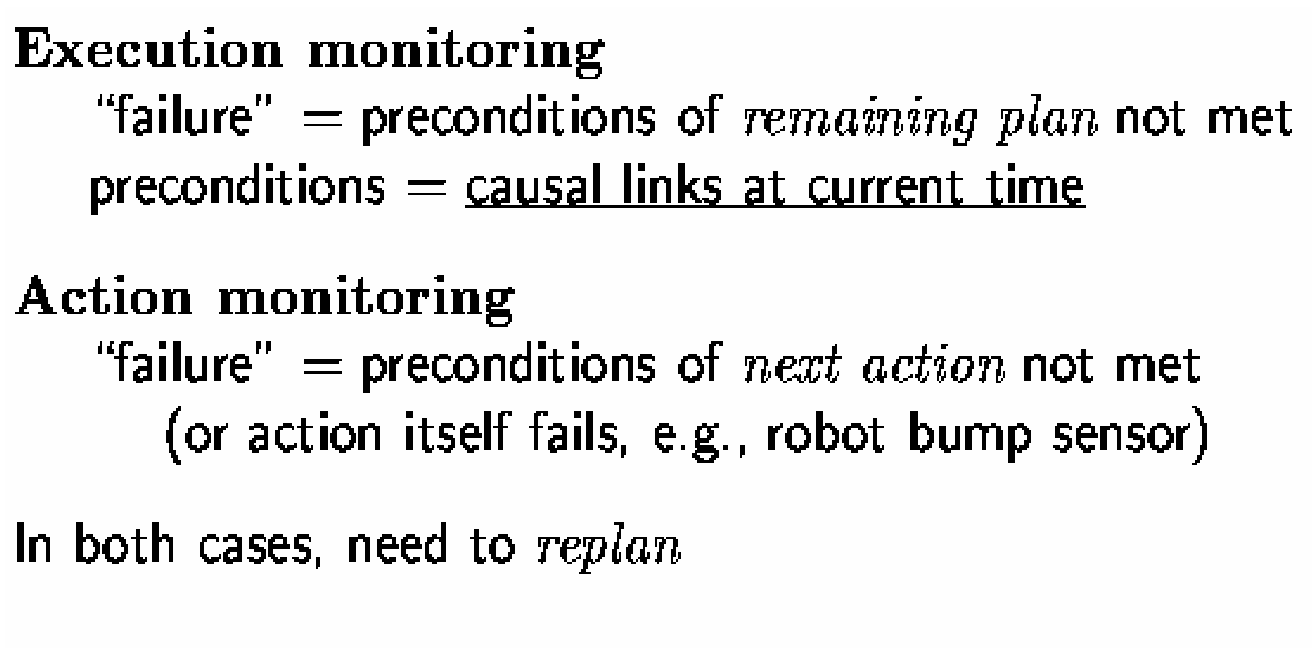
$[\ldots, \mathbf{If}(p, [then\, plan], [else\, plan]), \ldots]$

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



Practical Planning Solutions [3]: Monitoring & Replanning – Preview



Execution monitoring

"failure" = preconditions of *remaining plan* not met

preconditions = <u>causal links at current time</u>

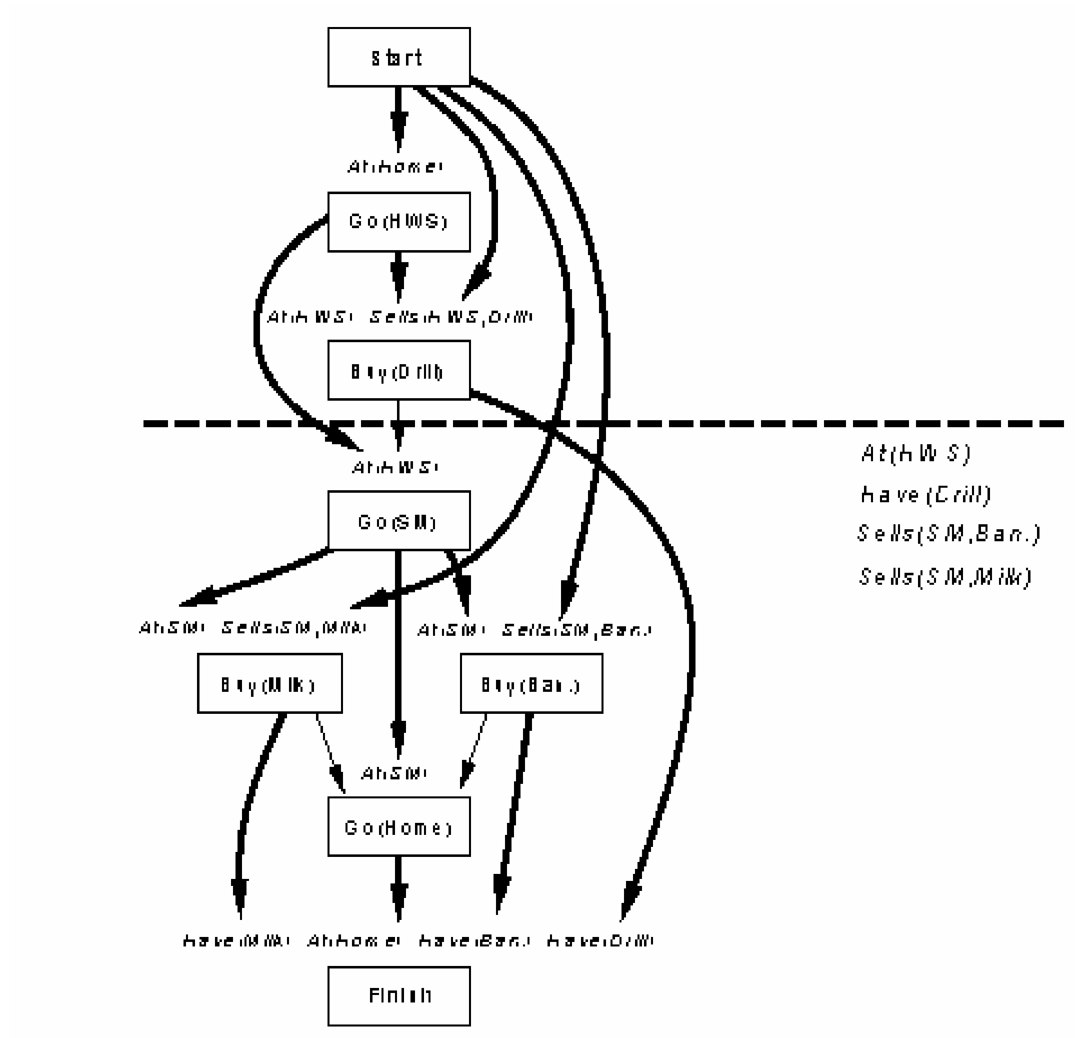
Action monitoring

"failure" = preconditions of *next action* not met

(or action itself fails, e.g., robot bump sensor)

In both cases, need to *replan*

Practical Planning Solutions [4]: Continual Planning – Preview



Terminology

- <u>Propositional Domains</u>
- <u>Hierarchical Abstraction</u> Planning: Refinement of Plans into Subplans
- <u>Bounded Indeterminacy</u>: Kind of Uncertainty about Domain
  - "How world is like"
  - "How it will be if I do A"
- <u>Robust Planning</u>
  - <u>Sensorless</u>: use coercion and <u>reaction</u>
  - <u>Conditional</u> *aka* <u>contingency</u>: IF statement
  - <u>Monitoring and replanning</u>: resume temporarily failed plans
  - <u>Continual</u> *aka* <u>lifelong</u>: multi-episode, longeval or "immortal" agents

Summary Points

- **Last Class: Sussman Anomaly, POP in Detail; Intro to Graph Planning**
  - **Plan linearization**
  - **Extended POP example: changing spare tire**
  - **Graph planning preview**
  - **Hierarchical abstraction planning (ABSTRIPS)**
- **Today: Graph Planning and HTN Preview**
  - **Graph planning (11.4)**
  - **Planning with propositional logic (11.5)**
  - **Analysis of planning approaches (11.6)**
  - **Summary (11.7)**
- **Next Class: Real-World Planning**
  - **Time (12.1)**
  - **HTN Planning (12.2)**
  - **Nondeterminism (12.3)**
  - **Conditional planning (12.4)**
- **Coming Week: Robust Planning Concluded; Uncertain Reasoning**