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](http://snipurl.com/v9v3)
Course web site: [http://www.kddresearch.org/Courses/CIS730](http://www.kddresearch.org/Courses/CIS730)
Instructor home page: [http://www.cis.ksu.edu/~bhsu](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

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

Complete partial-order plan (3 operators)

Figure 11.8 p. 392 R&N 2e

Figure 11.9 p. 392 R&N 2e

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., \( \text{Have}(A) \land \text{Have}(B) \land \text{Have}(C) \land \text{Have}(D) \)
  - \( \text{Have}(A) \land \text{Have}(C) : h = 2 \)

- 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) ∧ Eaten(Cake))
Action(Eat(Cake))
  PRECOND: Have(Cake)
  EFFECT: ¬ Have(Cake) ∧ Eaten(Cake))
Action(Bake(Cake))
  PRECOND: ¬ Have(Cake)
  EFFECT: Have(Cake)

Figure 11.11 p. 396 R&N 2e

Figure 11.12 p. 396 R&N 2e
Graph Planning — Cake Problem [2]: Mutex Conditions

- **Inconsistent effects**: one action negates an effect of the other. For example, \( \text{Eat}(\text{Cake}) \) and the persistence of \( \text{Have}(\text{Cake}) \) have inconsistent effects because they disagree on the effect \( \text{Have}(\text{Cake}) \).

- **Interference**: one of the effects of one action is the negation of a precondition of the other. For example, \( \text{Eat}(\text{Cake}) \) interferes with the persistence of \( \text{Have}(\text{Cake}) \) by negating its precondition.

- **Competing needs**: one of the preconditions of one action is mutually exclusive with a precondition of the other. For example, \( \text{Bake}(\text{Cake}) \) and \( \text{Eat}(\text{Cake}) \) are mutex because they compete on the value of the \( \text{Have}(\text{Cake}) \) 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 ≠ failure then return solution
        else if NO-SOLUTION-POSSIBLE(graph) then return failure
        graph ← EXPAND-GRAPH(graph, problem)
end loop

● Alternating Steps
  ● Solution extraction
  ● Expansion

● EXTRACT-SOLUTION: Goal-Based (Regression)
● EXPAND-GRAPH: Adds Operators, State Literals

Figure 11.13
p. 399 R&N 2e

Graph Planning: Spare Tire Example (Expanded to $S_2$)

Figure 11.14
p. 399 R&N 2e

Hierarchical Abstraction — Example: Review

- Start
  - Money → Buy Land
  - Good Credit → Get Loan

- Buy Land
  - Land → Build House

- Build House
  - Decomposes to
    - Get Permit
      - Land → Buy Land
    - Construction
      - Hire Builder
        - House
          - Finish
Hierarchical Task Network Planning: Review

\[
\begin{align*}
 n_1: & \quad \text{achieve}[\text{clear}(v_1)] \quad \text{clear}(v_1) \\
 n_2: & \quad \text{achieve}[\text{clear}(v_2)] \quad \text{clear}(v_2) \\
 n_3: & \quad \text{on}(v_1, v_3) \quad \text{do}[\text{move}(v_1, v_3, v_2)]
\end{align*}
\]

\[
[(n_1 : \text{achieve}[\text{clear}(v_1)])(n_2 : \text{achieve}[\text{clear}(v_2)])(n_3 : \text{do}[\text{move}(v_1, v_3, v_2)])
\quad (n_1 \prec n_3) \land (n_2 \prec n_3) \land (n_1, \text{clear}(v_1), n_3) \land (n_2, \text{clear}(v_2), n_3) \land (\text{on}(v_1, v_3), n_3)
\land \neg (v_1 = v_2) \land \neg (v_1 = v_3) \land \neg (v_2 = v_3)]
\]
Planning in Propositional Domains: Cargo Plane Example Redux

\[ \text{initial state} \land \text{all possible action descriptions} \land \text{goal} . \]

\[ At(P_1, SFO)^0 \land At(P_2, JFK)^0 . \]

\[ \neg At(P_1, JFK)^0 \land \neg At(P_2, SFO)^0 . \]

\[ At(P_1, JFK)^1 \iff (At(P_1, JFK)^0 \land \neg (Fly(P_1, JFK, SFO)^0 \land At(P_1, JFK)^0)) \lor (Fly(P_1, SFO, JFK)^0 \land At(P_1, SFO)^0) . \]
function SATPLAN\left(\text{problem}, \ T_{\text{max}}\right) \text{ returns } \text{solution or failure}
\text{inputs: } \text{problem}, \text{ a planning problem}
\qquad T_{\text{max}}, \text{ an upper limit for plan length}
\begin{align*}
&\text{for } T = 0 \text{ to } T_{\text{max}} \text{ do} \\
&\quad \text{cnf, mapping } \leftarrow \text{TRANSLATE-TO-SAT}(\text{problem}, T) \\
&\quad assignment \leftarrow \text{SAT-SOLVER}(\text{cnf}) \\
&\quad \text{if } assignment \text{ is not null then} \\
&\qquad \text{return } \text{EXTRACT-SOLUTION}(assignment, mapping) \\
&\text{return } \text{failure}
\end{align*}
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

\[ \ldots, \textbf{If}(p, [\textit{then plan}], [\textit{else plan}]), \ldots \]  

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

\[
\textbf{CheckTire}(x) \\
\textit{KnowsIf}(\textit{Intact}(x))
\]
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
PRACTICAL PLANNING SOLUTIONS [4]: CONTINUOUS PLANNING — PREVIEW

### Terminology

- **Propositional Domains**

- **Hierarchical Abstraction** Planning: Refinement of Plans into Subplans

- **Bounded Indeterminacy**: Kind of Uncertainty about Domain
  - “How world is like”
  - “How it will be if I do A”

- **Robust Planning**
  - **Sensorless**: use coercion and reaction
  - **Conditional aka contingency**: IF statement
  - **Monitoring and replanning**: resume temporarily failed plans
  - **Continual aka lifelong**: 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