# Notes for Meeting 14
## Goal-Driven Problem Solving

## Review of Problem Solving

To review, we can formulate the abstract task of problem solving as:

- Given: A current situation for a real or imagined environment;

- Given: A specification for some desired situation; and

- Given: A set of operations for changing situations and constraints on their use;

- Generate: A situation that satisfies the specification and (optionally) a sequence of operations that produces it.

The ability to solve unfamiliar problems of this sort is one of the hallmarks of human intelligence.

# Review of State-Driven Problem Solving

One approach to heuristic problem solving carries out forward-chaining search that is driven by the state. This involves:

- Selecting a state to chain off
- Finding all operator instances applicable in that state
- EITHER
  - Applying these operators to generate successor states OR
  - Selecting an operator and applying it to generate a new state

Many action-oriented AI systems adopt this framework, and it is especially popular in work on game playing.

However, it is clear that humans sometimes engage in another form of problem solving that chains backward from goals.

# Review of Query-Driven Inference

We have seen that an agent can draw inferences and prove theorems by a process of backward chaining that alternates between:

- Selecting a rule to use when chaining off a literal (OR search)

- Selecting an antecedent to chain over within a rule (AND search)

This process produces a proof tree that derives answers to the query from ground literals.

Many AI reasoning systems adopt this framework, especially those concerned with deductive inference.

However, it assumes that inference rules are monotonic, making it insufficient for action-oriented tasks.

# Means-Ends Analysis

Newell and Simon (1961) report analyses of think-aloud protocols on a problem-solving task.

To explain the behavior, they propose means-ends analysis, which:

- Transforms state S into desired state D by:
  - Finding differences between S and D;
  - Selecting an operator O that reduces some of D;
  - If operator O’s conditions do not match S, then transforming S into a state S’ that matches O’s conditions;
  - Applying O to S or S’ to produce new state S’’;
  - Transforming S’’ into D;

They incorporate this mechanism (organized differently) into the General Problem Solver (GPS), an account of human problem solving.

## Applications of Means-Ends Analysis

Researchers have shown that means-ends analysis is useful in a variety of settings, including:

- Many standard puzzles

- Algebra and physics problems

- Many planning tasks

There is compelling evidence that means-ends analysis plays a key role in many human problem-solving activities.

## Extensions to Means-Ends Analysis

Researchers have repeatedly built on the versions of means-ends analysis found in Newell and Simon’s GPS:

- STRIPS extended the approach to operate over logic-like formalisms and introduced macro-operator learning;

- Prodigy introduced search-control rules and techniques for learning them from success and failure;

- GIPS added probabilistic annotations on operators and incorporated methods for updating them; and

- Icarus combined means-ends analysis with hierarchical task networks and provided a way to acquire them.

These advances have produced a deeper understanding of goal-driven approaches to problem solving.

## Critiques of Means-Ends Analysis

AI planning researchers often criticize means-ends analysis because it:

- is entirely driven by goals but ignores interactions among them
  - this causes it difficulty on tasks like the "Sussman anomaly"
  - but one can adapt the method to prefer backward chaining but fall back on forward chaining
- treats plans as totally ordered, whereas using a partial ordering can avoid redundant search
  - but experiments show that partial-order planners are not always superior to total-order methods.

In response, there has been substantial work on "least-commitment" methods for problem solving and planning.

## Methods for Partial-Order Planning

The 1980s and 1990s saw substantial work on a new class of planning methods that:

- encoded plans as a set of partially ordered operator instances;

- carried out search through the space of partial plans;

- chained backward from unachieved goals or subgoals; and

- checked for and avoided interactions among these goals.

Work on partial-order planning emphasized logical properties like completeness of search and correctness of resulting plans.

## Means-Ends and Hierarchical Task Networks

Means-ends analysis has an interesting connection to hierarchical task networks that is seldom noted:

- Means-ends solves a problem by decomposing it into subproblems, but it must search through the space of such decompositions.

- An HTN planner takes advantages of known decompositions to constrain forward-chaining search or execution.

- This suggests that hierarchical task networks are generalized traces of successful means-ends analysis.

Icarus adopts this relationship between knowledge, problem solving, and learning.

# Interleaving Planning with Execution

The 1980s saw a shift from "classical planning" to reactive behaviors, which emphasized execution and control because:

- Most AI researchers had assumed that planning operate entirely in the agent’s head.

- But basic means-ends analysis alternates between chaining backwards from goals and applying operators when conditions are met.

- Newell and Simon never said that means-ends was entirely internal, and human problem solving (e.g., on Tower of Hanoi) is not.

Thus, means-ends analysis lends itself naturally to interleaving of problem solving/planning with execution.

# Problem Solving in Icarus

The Icarus architecture includes a problem-solving module that:

- Builds on modules for conceptual inference and skill execution;
- Comes into play whenever the system encounters problems that known skills cannot handle;
- Incorporates a modified version of means-ends analysis; and
- Interleaves backward chaining from goals with forward chaining execution from the current state.

This module provides Icarus with the ability to solve novel problems in similar ways to humans.

# Goals and Problems in Icarus

Although the Icarus problem solver draws on existing structures like concepts, skills, beliefs, and intentions, it also requires:

- goals, which describe desired beliefs that
  - may be uninstantiated, e.g., (on ?x ?y)
  - may be partly instantiated, e.g., (on ?x A)
  - may involve negation, e.g., (not (on ?x A))
- problems, which describe desired belief states as
  - a set of goals that may share variables
  - bindings between variables and constants
  - differences between goals and beliefs

Problems may also have an associated intention that aims to achieve its goals, and intentions may have associated subproblems.

Problems and Intentions from the Blocks World

```
(problem :id P1
 :goals       ((not (on ?other A)) (ontable A) (not (holding ?any)))
 :bindings    ( )
 :differences ((not (on B A))))

(intention :id I1  :parent P1
 :head        (unstack B A)
 :bindings    ((?from . A) (?on . B))
 :conditions  ((on B A) (not (on ?other B)) (not (holding ?any)))
 :effects     ((not (on B A)) (holding B))

(problem :id P2    :parent I1
 :goals       ((on B A) (not (on ?other B)) (not (holding ?any)))
 :bindings    ((?from . A) (?on . B))
 :differences ((not (on C B))))

(intention :id I2  :parent P1
 :head        (unstack C B)
 :bindings    ((?from . B) (?on . C))
 :conditions  ((on C B) (not (on ?other C)) (not (holding ?any)))
 :effects     ((not (on C B)) (holding C))
```

# Problem Solving in Icarus

Icarus uses a variant of means-ends analysis to solve novel task; given a new or current problem, it:

- Generates all bindings and associated differences
- Selects one set of bindings / differences at random
- Finds all skill instances that would remove any difference or that match the current state
- Selects one skill instance heuristically and stores as intention with the problem
- If the intention’s conditions match current beliefs, calls the execution module to carry out the intention
- Otherwise creates a new subproblem based on these conditions and stores with the intention

This mechanism is similar to earlier work on GPS, STRIPS, and Prodigy, but differs in important ways.

## Heuristics for Selecting Intentions

Icarus relies on two numeric heuristics to select among candidate intentions:

- Prefer skill instances that reduce more differences (unsatisfied goals) associated with the problem;

- Prefer skill instances that have fewer conditions unmatched by the current belief state.

The problem solver applies these lexicographically, with the first one taking precedence by default to produce backward chaining.

When one changes the setting to give the second priority, the system carries out forward-chaining execution biased by goals.

# Assignments for Meeting 15
# Creativity in Problem Solving

Read the articles:

- Goel, A. (1997). Design, analogy, and creativity. IEEE Expert, 12, 62-70.

- Work on the fourth exercise (due 11:59 PM on 3/11/2011) and bring questions about it to class.