Generate and Test, Means-Ends Analysis, and Problem Reduction
—
Winston, Chapter 3

Michael Eisenberg and Gerhard Fischer
TA: Ann Eisenberg

AI Course, Fall 1997

Generate and Test

To perform generate and test

- Until a satisfactory solution is found or no more candidate solutions can be generated,
  - Generate a candidate solution
  - Test the candidate solution
- If an acceptable solution is found, announce it; otherwise, announce failure.

Questions to ask:

- which sort of problems does the “generate-and-test” solve (and not solve)?
- criteria that good generators always satisfy?

Example: To Break into a Safe

lock: 00-00-00

number of combinations: 100^3= 1 million combinations

how long will it take

assumptions: 3 per minute, half the combinations on average

165 000 minutes ---> 2750 hours ---> 114 days ---> 16 weeks (working 24 hours per day)

Example: Fonts, Printing, Color, Graphs

claim: powerful generators need powerful testing procedures

example: fonts

IN THE OLD DAYS COMPUTER COULD ONLY CREATE WRITTEN MATERIAL IN THIS FORM

Nowadays, we *have* the **power** for misuse of fonts!

color:

- high performance work station can display several million colors
- but: when color is used inappropriately it can be very counterproductive and few software designers have much experience with the use of color

Quality of Generators

complete: produce all possible solutions

nonredundant: do not propose the same solution twice

informed: use knowledge to restrict the solutions proposed

a powerful idea (of Goldstein and Papert, 1977):

> "The fundamental problem of understanding intelligence is not the identification of a few powerful techniques, but rather the question of how to represent large amounts of knowledge in a fashion that permits their effective use and interaction."

Clues that Signal Progress and Stable Intermediate Forms
—
The Defect Safe

see Simon, "Sciences of the Artificial", p 193

intact safe:

- 10 dials, each with 100 settings
- blind trial-and-error search: 100^{10} settings,
- on average: inspect half of them ---> $50*10^{19}$ (50 billion billion)

defect safe:

- a click is heard when one dial is turned to the correct setting
- each dial can be adjusted independently
- total number of trials: 10*50 = 500

The Evolutionary Model

- things evolve in response to some kind of selective force

- simple scheme of evolution:
  - generate: produce variety (e.g. genetic mutation)
  - test: to evaluate the newly generated forms (e.g. natural selection)

- **Problems with Evolution:**
  - is myopic
  - reaches local maxima (instead of global ones)
  - moving away from a local maxima implies: going across a valley

State Space

A state space is a representation that is a semantic net in which

- The nodes denote states

- The links denote transitions between states

definitions:

current state:	state one is in
goal state:	state where one want to be

problem:

to find a sequence of transitions that leads from the initial state to the goal state

Means-Ends Analysis

To perform means-ends analysis,

- Until the goal is reached or no more procedures are available,
  - Describe the current state, the goal state, and the difference between the two.
  - Use the difference between the current state and goal state, possibly with the description of the current state or goal state, to select a promising procedure.
  - Use the promising procedure and update the current state.
- If the goal is reached, announce success; otherwise, announce failure.

Determining the Means: Difference-Procedure Tables

distance	airplane	train
more than 300 miles	V	
between 100 and 300 miles		V
less than 100 miles		

Semantic Tree

A semantic tree is a representation that is a semantic net in which

- Certain links are called branches. Each branch connects two nodes; the head node is called the **parent node** and the tail node is called the **child node**
- One node has no parent; it is called the **root node**. Other nodes have exactly one parent.
- Some nodes have no children; they are called **leaf nodes**.
- When two nodes are connected to each other by a chain of two or more branches, one is said to be the **ancestor**; the other is said to be the **descendant**.

With constructors that

- Connect a parent node to a child node with a branch link

With readers that

- Produce a list of a given node's children
- Produce a given node's parent

Problem Reduction

To determine, using **REDUCE**, whether a goal is achieved,

- Determine whether the goal is satisfied without recourse to subgoals:
  - If it is, announce that the goal is satisfied.
  - Otherwise, determine whether the goal corresponds to an And goal:
  - If it does, use the REDUCE-AND procedure to determine whether the goal is satisfied.
  - Otherwise, use the REDUCE-OR procedure to determine whether the goal is satisfied.

To determine, using **REDUCE-AND**, whether a goal has been satisfied,

- Use REDUCE on each immediate subgoal until there are no more subgoals, or until REDUCE finds a subgoal that is not satisfied.
- If REDUCE has found a subgoal that is not satisfied, announce that the goal is not satisfied; otherwise, announce that the goal is satisfied.

To determine, using **REDUCE-OR**, whether a goal has been satisfied,

- Use REDUCE on each subgoal until REDUCE finds a subgoal that is satisfied.
- If REDUCE has found a subgoal that is satisfied, announce that the goal is satisfied; otherwise, announce that the goal is not satisfied.

Examples of Applications

- **DENDRAL — analyzes mass spectrograms**
  illustrates: generates-and-test method
  generator: structure enumerator and synthesizer
  test: compare real mass spectrogram with those produced by the generator

- **SAINT — Mathematics Toolkits**
  illustrates: problem reduction
  further developments:
    - Macsyma
    - Mathematica

problem reduction is a ubiquitous

- for problem solving and programming (subroutines)
- for understanding complex things: “if there is a complex thing that we do not yet understand, we can come to understand it in terms of simpler parts that we do already understand.” (Dawkins, The Blind Watchmaker, p 11)