CSE6390 3.0 Special Topics in AI & Interactive Systems II
Introduction to Computational Linguistics
Instructor: Nick Cercone - 3050 CSEB - nick@cse.yorku.ca
Winter Semester, 2010

Representational Typology

A typology or Representations (for problem-solving in Artificial Intelligence)

Criteria

1. Sufficiently accurate model of "reality"
2. Easy to use

Interesting Fact

Representations are heuristic devices for looking at problems. A good representation suggests a good problem-solving method. That a good representation is formally equivalent to a bad one is an interesting mathematical fact, but that's all.

Types

1. Enumeration
2. State-Space
3. Problem Reduction
4. String Rewriting
5. Combinations

Enumeration

An "enumerative representation" begins with a set **U** of potential solutions and a rule for evaluating them. If **U** is finite, solutions may be examined in some order, and a correct answer selected. If **U** is infinite but denumerable the members of **U** can be generated and tested for correctness of solution. Enumerative methods have been maligned in Artificial Intelligence, in part because early writings in the field implied that they relied on "brute force and ignorance" to keep generating trial solutions blindly until a correct solution was found. In practice, however, some quite elegant enumerative techniques have been developed. Often, when no known analytic solution is applicable, a backtracking method is used, and backtracking is "controlled" enumeration. An abstract representation of the method itself suggests when enumeration is likely to be useful. Let **f** be a function which selects at least one solution **s** in **S** for any **S** which is a (proper) subset of **U**. An enumerative technique is executed in the following steps:

1. Begin with a set S_0 (proper) subset of **U** of trial answers. Set $i \rightarrow 0$.
2. If S_i contains an answer, end. Otherwise goto 3.
3. Construct $S_i + 1 \rightarrow f(S_i)$, increase **i** by **1** and goto 2.

The feature of an enumerative method that separates it from other methods is that each step $S_i + 1$ are produced solely from consideration of S_i. The next step is, then, a function of the value of the current step and does not depend upon a comparison between the current step and the answer. Intuitively, this seems like blind problem-solving, and indeed it is "unless the characteristics of an answer are always the same". If this is true, then the implied comparison between the answer state and the members of S_i can be built into the definition of the enumerating function **f.** Enumeration has been used successfully in theorem- provers, game playing programs, and other AI endeavors.

State-Space Approach

The "state-space" approach is a very popular problem-solving representation. It assumes the existence of a countable set **S** of "states" and a set **O** of "operators" which map the states of **S** into themselves. The problem-solver is seen as moving through space defined by the states in an attempt to reach one of a designated set of goal states. A problem is solved when a sequence of operators

$$O = O(1), O(2), \ldots, O(k)$$

is found such that the following relationship holds for some state s_0 in the set of possible starting states and s_g in the set of goal states

$$sg = O(k)\,(O(k-1)\,(\ \ldots\ O(2)\,(O(1)\,(s0))\ \ldots\ ))$$

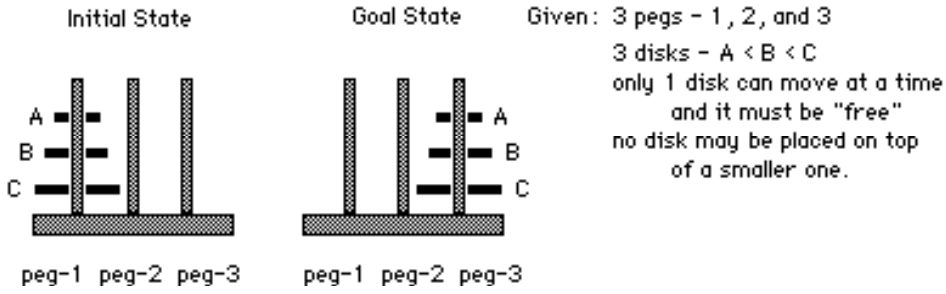
There is a useful isomorphism between the problem of finding the sequence **O** and the problem of finding a path through a graph. Think of **S** as defining the nodes of a graph, with arcs between the nodes **i** and **j** ⇔ there is an operator **o** in **O** which maps s_i into s_j. A path from one state to another is equivalent to a sequence of operators mapping from the first state to the second. The graphical representation of state-space problem-solving has three advantages:

1. it is intuitively easy to grasp;
2. it leads to the natural extension in which we associate a cost, ck, with the application of each operator ok. Graphically, this can be represented by labeling each arc in the graph with its cost; and
3. the next step to be explored can be made a function of a comparison between a goal state and a final state.

Problem Reduction

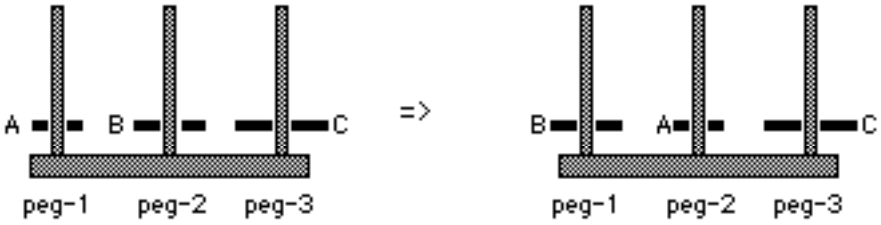
In the problem reduction method one identifies subproblems, which, if solved separately, will constitute a solution to the original problem. Each problem is then attacked, and the results combined to create a solution to the total problem. This process can be applied recursively, to generate subproblems, sub-subproblems, etc., until finally a trivial set of "problems" whose solution is known is identified. The solutions to these problems thus can be combined into the solution of the larger problem. The 3 disk, 3 peg Tower of Hanoi problem offers an interesting illustration. The problem begins with 3 pegs and disks as illustrated below:

The goal is to move the disk to the right hand peg, peg-3, within the given constraints. To solve the problem, the large disk must be placed on peg-3. This, of course, implies that disk C is exposed on peg-1 and that peg-3 is empty, a situation which could only occur if peg-2 contained disks A and B in the proper order. This (sub) problem involves two subproblems to be solved in order, as illustrated below:



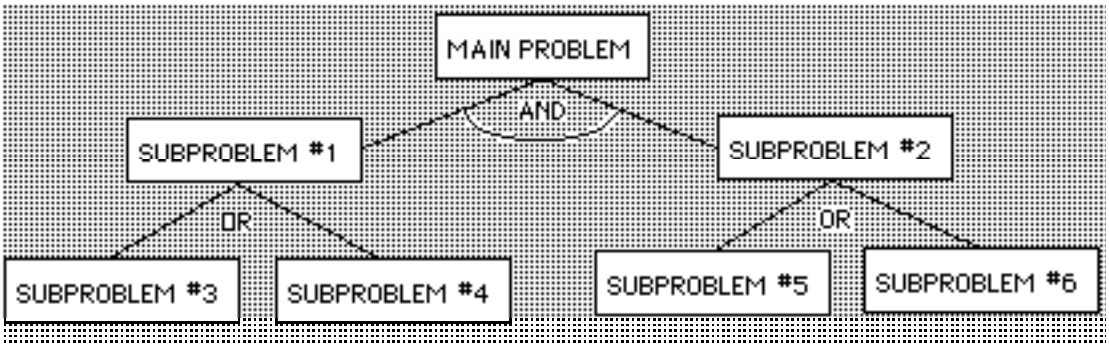
This last diagram illustrates a trivial solution which we can assert in a single move. After these problems have been solved, however, there remains an additional problem, moving disks A and B into place. The reader can easily apply the preceding reasoning to generate further subproblems, thus arriving at solutions for all the unsolved problems.

The subproblems discussed thus far are called collectively AND subproblems since all of them must be solved in order to arrive at a solution to the main problem. It is often the case that several OR subproblems can be located, subproblems such that if any one of them is solved, the main problem is solved. This occurs in the Tower of Hanoi problem. The following figure illustrates two configurations of disks such that if either of them is reached, the main problem is trivial to solve.



The problem reduction representation can be depicted by a special sort of graph, called a "tree graph" or an "AND/OR graph" as illustrated below.

To solve the main problem, subproblems 1 and 2 must be solved. To solve subproblem 1, either subproblem 3 or 4 may be solved; to solve subproblem 2, either subproblem 5 or 6 must be solved.



String Rewriting

In the string rewriting representation, possible "states of the world", including starting and goal states, are treated as well formed expressions in some language. Operators mapping from one state of the world to another are treated as rewriting rules which map one well-formed expression into another, e.g., high school algebra. Suppose we wanted to prove

X + (B + C) = B + (X + C)

using the rules of commutivity and associativity. The proof is

X + (B + C) → (X + B) + C	associativity of +
→ (B + X) + C	commutivity of +
→ B + (X + C)	associativity of +

The selection of the operation (rewriting rule) to use at each step is based upon a comparison of the structures of the current and desired well formed expression. There is clearly an identity between the state-space representation and the string rewriting procedure. As a matter of heuristics only some problems are easily conceptualized as string rewriting problems; most are not.

Combining Representations

The problem reduction representation works because problem-solving is often a hierarchical affair. Problem reduction alone, however, hides an important point. It is not necessary that the main problem and all the subproblems are attacked using the same representation. For example, consider Shakey, the SRI robot. When Shakey is ordered to change the environment, the main problem faced by the robot is broken down into subproblems, and a graphic (state-space) representation is used to determine the sequence in which the subproblems should be attempted. Theorem proving techniques are applied to specific steps of the sequence to demonstrate that applying a certain operator will have the desired effect. The methods used are enumerative and serious consideration has been given to representing this problem as a string rewriting problem.

**** Biggest Overall Problem ****

How do we restrict the search space to a space of feasible solutions?

HEURISTICS! rule of thumb, strategy, method or trick used to improve the efficiency of a system which tries to discover the solutions to complex problems.

any stratagem for improving the performance of an artificial intelligence program