Knowledge representation (INFO0049-1)
Exercise session 6

24 Mar 2015

Try to draw search trees wherever possible to see how prolog executes a query

1. **Gray code**

1. An n-bit Gray code is a sequence of n-bit strings constructed according to certain rules.

   For example,
   n = 1: C(1) = ['0','1'].
   n = 2: C(2) = ['00','01','11','10'].
   n = 3: C(3) = ['000','001','011','010',´110´,´111´,´101´,´100´].

   Find out the construction rules and write a predicate gray(+N, -C) that returns the N-bit Gray code for a positive natural number N.

   ?- gray(2, L).

   L = [‘00’, ‘01’, ‘11’, ‘10’]

   For more information on Gray code, take a look at the following link:
   http://en.wikipedia.org/wiki/Gray_code

2. **Cube puzzle**

2. You are given 4 cubes. On each face (side) of each cube, there is a figure. There are 4 different figures in total (triangle, star, rectangle, circle).

   We will stack these four cubes on top of each other (cube1 at the bottom, then cube2, and so on). Note that for each cube, there are 24 possible orientations: there are 6 possibilities for the bottom, and once we know which is the bottom there are 4 possibilities for the front (once we know the bottom and the front everything else is fully determined).
   We define a stack as 'correct' if the orientations are such that on each 'visible' side of the stack (front side, back side, left side and right

side) each figure occurs exactly once. Your task is to write a Prolog-program that can find all orientations for the four cubes that yield a correct stack.

This is not an easy problem, so try to solve it gradually:

1. Think about alternative high-level strategies to solve the problem. Think about which strategy is most efficient.
2. This problem is all about cubes, orientations and stacks. Find a good representation in Prolog for
    - the 4 given cubes with their figures,
    - an orientation of a cube,
    - a stack of cubes in certain orientations.

   To decide about the representations, think about which actions will have to be performed on the orientations and on the stack, and which representations will be easiest or most efficient for this.

3. Write a predicate that takes as input a list of 4 cubes and has as output a correct stack of these cubes. Do this by transforming the high-level strategy you found above into Prolog code. Use non-existing predicates for lower level tasks.
4. Write Prolog predicates to solve the lower level tasks.
5. There is one complication that we did not consider yet: if we indeed try out all 24 possible orientations for the first cube, then we get many solutions that are simply 'variations' of each other but not really 'different' solutions. This is because if you have a correct stack, you can obtain another correct stack by rotating all cubes 90, 180 or 270 degrees, or by turning every cube upside down. We do not want to consider all these variations as different solutions. We can solve this problem by noticing that *for the first cube*, the only thing that matters is which four sides are visible (front, back, left and right). This gives only 3 different possibilities ('basic' orientations) for the first cube, instead of 24. Adapt your program in order to take this into account, i.e. make sure you consider only the 3 basic orientations for the first cube, but all 24 orientations for the second, third and fourth cube
6. If you defined the 24 possible orientations by simply listing them, can you think of a smarter/shorter way of defining them, making use of the 3 'basic' orientations that you just defined? Hint: each of the 24 orientations can be derived from one of the 3 basic orientations by means of rotating the cube (rotating horizontally and/or turning upside-down).