MAU11602 Assignment 4, Due Wednesday 28 February 2024
Solutions

1. An old German puzzle toy has a set of blocks of square or rectangular shape which can be slid within a frame. The goal is to get from the position

to

As you can probably guess, the images above are from a computerised version of the original mechanical toy. The dark grey squares or rectangles represent spaces without a block, which are necessary because if you're sliding a block you need a free space to slide it to.

(a) Can you formulate this as a non-deterministic computation?

(b) Is it possible for your algorithm to terminate unsuccessfully?

(c) Consider the tree of all possible states reachable from the initial state. Can you be sure that if the puzzle is solvable then a breadth first traversal of this tree will find a solution?

(d) Can you be sure that if the puzzle is solvable then a depth first traversal of this tree will find a solution?

Note: You don't have to give a very detailed answer to the first question. The only reason I'm asking you that question at all is that there is in fact

more than one way to formulate this as a non-deterministic computation and the answer to at least one of the subsequent questions depends on which formulation you pick so your description should be clear enough that your marker can figure out whether your subsequent answers are correct for your formulation of the problem.
Solution: The most obvious choice is as follows.

(a) The first diagram is the initial state and any move which slides a block into an adjacent blank space is an allowed state transition. The successful termination condition is reaching the position in the second diagram. Having no available moves is an unsuccessful termination.

(b) No. All moves are reversible so if we got to a state then undoing the previous move is an allowed state transition. The only state for which this argument doesn't work is the initial state, but it's clear that there are allowed moves there. In fact there are six of them.

(c) Yes. There are only finitely moves in any state so in the tree every node has finitely many children. Breadth first traversal always works in such a case.

(d) No. There are infinite branches. For example we could take the middle block in the initial position and simply slide it back and forth forever. Depth first traversal can fail when there are infinite branches.

A somewhat cleverer option is the following.

(a) The first diagram is the initial state and any move which slides a block into an adjacent blank space is an allowed state transition, provided the configuration it leads to is not one we've seen before. The successful termination condition is reaching the position in the second diagram. Having no available moves is an unsuccessful termination.

(b) The argument for a no answer above certainly no longer works and in fact the answer is yes. It is possible to back yourself into a corner where there are no moves which don't repeat a previous position, although finding such a sequence of moves is rather tricky and I don't expect anyone to supply an example but for completeness I'll mention the following one: If we label the columns left to right as a, b, c, and d and label the rows from bottom to top as 1, 2, 3, 4, 5 and the directions left, right, up and down as w, e, n and s then one such sequence is a1n, b1w, b3e, b2n, c2w, c1w, d1w. The only move which is physically possible in this configuration is c1e, which simply undoes the move d1w, repeating the previous position.

(c) Yes. There are still only finitely moves in any state so in the tree every node has finitely many children. Breadth first traversal always works in such a case. There is a bit more work to be done though. Our tree

now includes only those states which were reachable without repeating a position but the original problem had no such prohibition. Is it possible that the problem is solvable, but only by repeating positions, in which case our tree traversal won't find the solution? No. If there's a solution then there's a shortest one. This shortest solution has no repeated positions because if it did then you could find a shorter one by picking a repeated position and removing all moves between its first occurence and its last.

(d) Yes. There are only finitely many configurations and we're not allowed to repeat them so the tree must be finite. Depth first traversal always works on finite trees. There is the same issue as in the previous question but the solution is the same.

2. As explained in the notes, one shouldn't strictly speaking refer to free variables or bound variables in an expression but rather to free and bound occurences of a variable. Normally the distinction isn't relevant because either all occurences of a variable are free or all of them are bound, but it is possible for a variable to have some free occurences and some bound occurences in an expression.

   (a) Give an example of an expression where at least one variable has at least one free occurence and at least one bound occurence.

   (b) Writing expressions like this is generally discouraged. Why do you think it isn't simply forbidden?

   Solution:

   (a) A short example is $\{(fx) \wedge [\forall x.(gx)]\}$ but there are a lot of possibilities. In this one the first x is free and the other two are bound.

   (b) Such expressions aren't needed for descriptive completeness since we can always find another expression with the same meaning.

   $$\{(fx) \wedge [\forall x.(gx)]\},$$

   for example, means the same thing as

   $$\{(fx) \wedge [\forall y.(gy)]\}.$$

   The reason such expressions aren't forbidden is that It's hard to find a set of rules of inference which enforce this. We inherited a rule of inference, for example, from zeroeth order logic that allowed us to infer $(P \wedge Q)$ from P and Q, where P and Q are Boolean expressions. (fx) and $[\forall x.(gx)]$ are each perfectly good Boolean expressions in first order logic. If we wanted to forbid expressions where one occurence of a variable is free and another is bound we would have to modify our rule of inference for the $\wedge$ operator to say that we can infer $(P \wedge Q)$ from P and Q only if the set of free variables in P is disjoint from the set of bound variables in Q and vice versa. We'd have

to do similar things to each of our rules of inference. The resulting formal system would be quite messy.