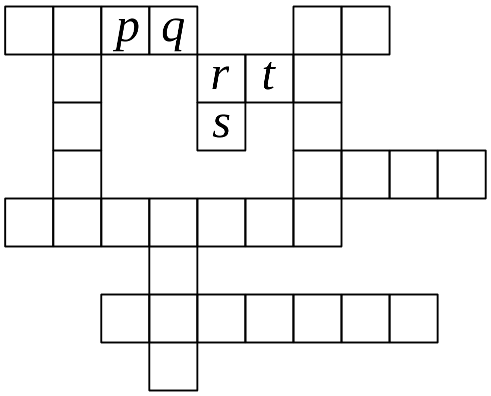
FINITE-STATE CONTROL FOR ROBOT-MOTION

Thin Maze (M):

- A finite, connected set of unit squares on a grid which form a pathway of *unit thickness* with branches. For simplicity, assume that the maze is *acyclic*, i.e., there are no cycles.
- It may be arbitrarily large, with size = #(unit squares) ≥ 2.
- There is *no coordinate system* and no numbering of the squares to identify them individually.
- A maze-square is considered *adjacent* to its immediate neighbors to the east, west, north, and south (and thus at most 4 neighbors).
  - The squares q and r below are not adjacent.
  - The square r has two neighbors $\{s, t\}$

Robot (R):

- It has the shape and size of a unit square. It can recognize the presence/absense of only the neighboring squares of its current position in the maze, and it can move to one of them by going east, west, north, or south. (It does not rotate.)



ROBOT-MOTION CONTROL (CONTD.)

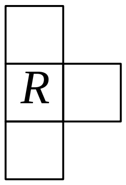
Problem.

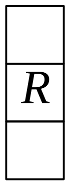
- Design a finite-state control (without any final-state) to direct the movements of R so that the following is true:
  - For any thin maze M and any given initial position in M, R repeatedly traverses the whole M, without ever coming to a stop or being confined in a particular part of M.
  - R does not need to know when a traversal of M is completed.
  - The output of the finite-state control is the move-direction at each successive time point.

Input symbols:

- Each input symbol consists of a 4-tuple, which describe the presence or absence of a neighbor in the east, west, north, and south, in that order. There are $2^4 - 1 = 15$ possible input symbols.

Input symbol = (e, –, n, s)





Input symbol = (–, –, n, s)

"–" indicates the absense of the corresponding neighbor.

- As the robot moves, the successive input symbols change depending on the thin maze M and the robot's motion.
  - If R moves south above, the new input becomes (–, –, n, –).

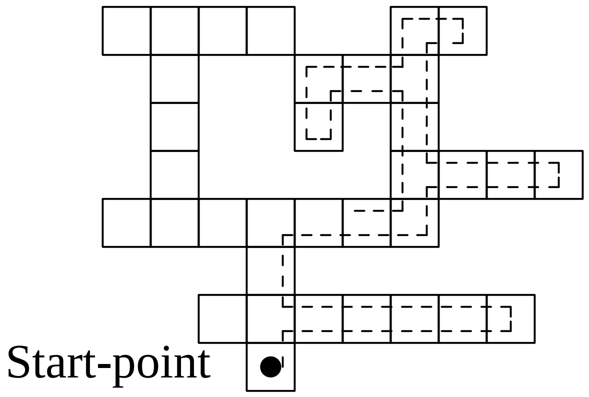
Output symbols:

- The four possible direction {e, w, n, s} of move by R.

FINITE-STATE CONTROL GIVES A SOLUTION ALGORITHM

Strategy to Systematically Visit All of M: Keep to the right.

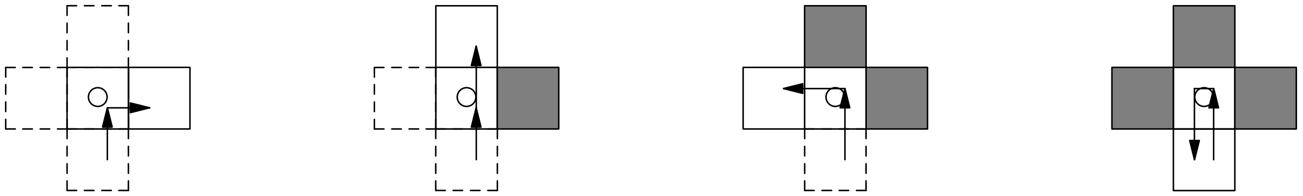
- Moving around in a haphazard fashion cannot possibly help.
- Not concerned at this point about optimizing the number of moves needed for a complete traversal of M.



Part of the move sequence based on "Keep to the right" strategy.

Implementing the Strategy:

- Need to know most *recent move-direction*; this forms the finite control-memory, i.e., the state.
- Need to choose the next move-direction among $\{e, w, n, s\}$ based on the local knowledge of M at the current position of R.

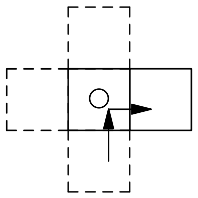


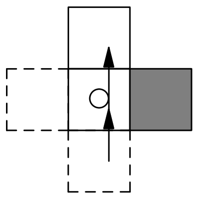
state = N, move = e input = (e,?, ?, ?)   state = N, move = n input = (–, ?, n, ?)   state = N, move = w input = (–, w, –, ?)   state = N, move = s input = (–, –, –, s)

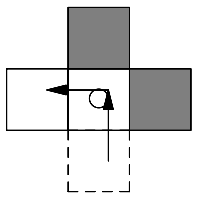
"?" = don’t care (dashed square), and shaded-square = not present

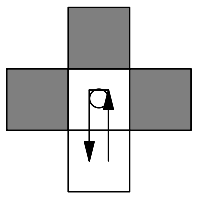
PART OF THE STATE-DIAGRAM FOR KEEP-TO-THE-RIGHT CONTROL STRATEGY

Transitions at state N:


state = N, move = e
input = (e,?, ?, ?)


state = N, move = n
input = (–, ?, n, ?)


state = N, move = w
input = (–, w, –, ?)


state = N, move = s
input = (–, –, –, s)

"?" = don't care (dashed square), and shaded-square = not present

$$\begin{tikzcd} W & N \arrow[l, "{(-,w,-,?)/w}"'] \arrow[r, "{(e,?,?,?)/e}"] \arrow[loop above, "{(-,?,n,?)/n}"] \arrow[rr, bend right, "{(-,-,-,s)/s}"'] & E & S \end{tikzcd}$$

Transition-label = input-symbol/move-direction.

Question:

•? Complete the state-diagram; indicate the start-state (there are no final-states because the robot never stops).

•? Complete the partial move-sequence in the thin maze shown earlier.

•? Prove that keep-to-the-right strategy works for all thin mazes and all start-positions. (Hint: Find a condition that holds when the robot returns for the *first time* to its initial position.

•? Give an example to show that the keep-to-the-right strategy does not always work if the maze has thickness > 1 in some places.

•? Give another another strategy that works for all mazes and show its transition-diagram.

EXERCISE.

1. Find a finite-state control such that the robot will come to a dead-end and stop for all thin mazes and all start-positions. (A dead-end is a square which has only one neighboring square. You do not need to go to any specific dead-end such as a northmost dead-end.) Describe in English your general strategy.

2. Consider the following control, which uses no control-memory.

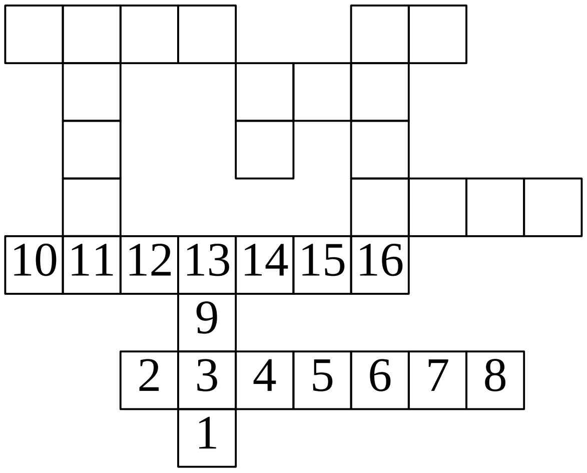
   Select move-direction according to the priority: $e > n > w > s$ irrespective of the preceding move-direction. For example, select move-direction s only if no other choice is possible.

   Show a thin maze M and a start-position for which this strategy fails, i.e., does not lead to a complete traversal of M. Show the move-sequence of R based on the above strategy.

3. What can you say about the following strategy:

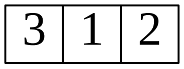
   First visit all squares that are 1 step away from the initial position, then visit the squares that are 2 steps away from the initial position, etc. (The robot may have to return to previously visited squares in between as it explores more and more distant squares.)

   Show a possible sequence of squares (say, for 15 moves or so) visited by this strategy for the following maze using the numbering of squares as indictated; only show the new squares visited (ignoring repeated visits to any square). Assume the initial position to be 1. What could be a potential problem in implementing this strategy by a finite control?



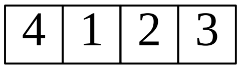
4. Consider a fi xed general maze M, which maybe a region (convex or non-convex) and which may have cycles and holes, in particular. Given a starting position s in M, we can consider all possible move sequences in M from s, and represent these move sequences using the symbols in $\Sigma = \{e, w, n, s\}$. The empty string λ will represent the empty move-sequence. Let $L(M, s)$ denote the language of these move-sequences over the alphabet Σ.

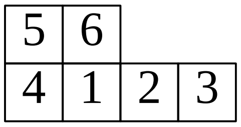
(i) Consider the following maze and the starting position $s = 1$.



What is the language $L(M, s)$? Give an English description of the language?

(ii) Show that $L(M, s)$ is regular for all M and arbitrary position s in M. Illustrate your answer by designing an FSA for the language in (i); the number of states may depend on M. Keep your design simple, name your states properly, and draw your state diagram in a way that shows its relationship to M; don't forget to specify the fi nal-states. You do not need to show the error state. Also show your FSA for each of the following mazes, with the start-position 1 in each case.





(iii) What does the quotient language $L(M, s)/x$ corresponds to (in the general case)? Illustrate your answer for the second maze in (ii) and the strings $x = ewwn$ and $x = ewww$.

(iv) Given two mazes M_1 and M_2, and the associated starting positions s_1 and s_2 in them, under what conditions will we have $L(M_1, s_1) = L(M_2, s_2)$?

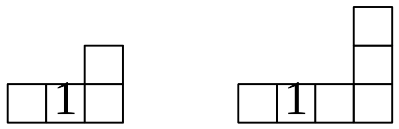
(v) Suppose we define $L(M) = \bigcup_{all\text{-}s} L(M, s)$. Show that $L(M)$ is a regular language for all M. Under what condition, will we have $L(M_1) = L(M_2)$?

(vi) Could we use the observation in (v) to design a method (algorithm) to test $M_1 = M_2$ for any two given mazes? Can we represent (the computations in) your method by an FSM? (What are its inputs and outputs?)

(vii) Which of your conclusions/observations in (i)-(vi) will not be valid if M is an infinite maze (still connected in one piece)?

5. Consider a variation of the FSM for traversing a maze using the "keep to the right" policy, where the robot comes to a stop when it reaches a dead-end. Now we present to the robot all L-shaped mazes shown below with $k + 2$ ($k \geq 1$) squares in the horizontal-chain and $k + 1$ squares in the vertical-chain. The start-position of

the robot is next to the leftmost square in the horizontal-chain.



k = 1 k = 2

The language of move-sequences produced by all such mazes is

$$L = \{e^k n^k : k \geq 1\}, \text{ which is non-regular.}$$

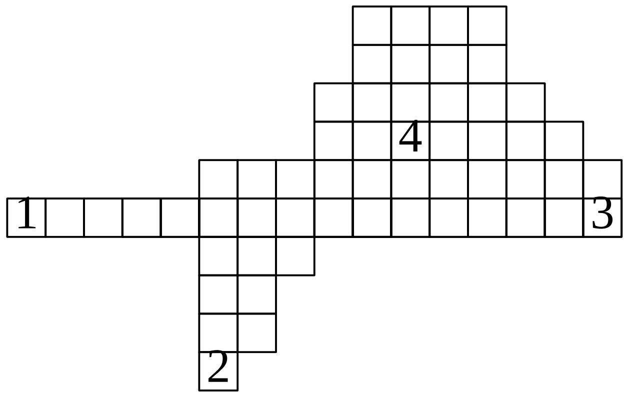
How is it that the underlying FSM in the robot is responsible for generating a non-regular language?

6. A maze is said to be *h-convex* (’*h*’ for "horizontal") if the following condition holds:

   If A and B are any two squares in the maze that are in the same horizontal row, then all squares between them are also in the maze. (See the squares marked 1 and 3 below.)

   The notion of *v-convexity* (’*v*’ for "vertical") is defined similarly. Finally, a maze is *convex* if it is both h-convex and v-convex. Shown below is an an example of a convex maze. In particular, the maze is not just a thin pathway any more, it has thickness; also, the thickness may be different in different places. If we remove any of the squares 1, 2, and 3, the maze still remains convex; however, removal of the square 4 destroys both *h*-convexity and *v*-convexity.

   Design an FSM which finds (and stops) at a (any) northmost square in a convex maze irrespective of where it is placed



initially. State in English the general "strategy" used by your FSM.

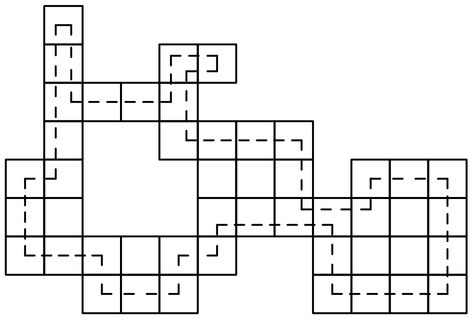
Submit the run of the program using the above maze for each of the starting positions marked 1, 2, 3, and 4. Also, mark the path (on your paper submission) taken by the robot in reaching the fi nal position in each case.

7. Design a fi nite-state control which will visit all squares of a convex maze repeatedly without ever stopping. (Hint: use the "horizontal sweeps" strategy, where the robot visits each horizontal strip all the way to the left and to the right before it moves to the next strip up. When it cannot go up any more, then it repeats the process, going down and visiting each horizontal strip completely on the way before stepping down to the next horizontal strip.)

Show the program output, including your transitions, for the convex maze shown in Problem 8.

8. Consider an arbitrary (fi nite and connected) region such as shown below, with possibly one or more holes. Design an fi nite-state control so that the robot will traverse along the outer-boundary of

the region, say, in the anti-clockwise direction, when initially placed at a northmost position. The dashed curve below shows such a boundary traversal; several internal squares in *G* are not visited in this traversal.



What problems would arise in designing a finite-state control if the initial position is only known to be a boundary position? (Certain animals, such as shrews, rats, etc., often use variations of boundary traversal in searching for food and exploring general areas, staying close to a boundary. An FSC for boundary traversal can be combined with other FSC's to solve many interesting traversal problems. For example, consider the following generalization of a convex region, where every square is *visible* – horizontally or vertically – from some boundary square. The region shown above has this property. Many famous gardens, where the holes correspond to areas covered by large bushes or trees, are designed with this property so that one can walk around the boundary of the garden and still be able to see all the flower-beds in the garden from at least one position.)