

6.004 Computation Structures
Lab #3

Roboant®

The 6.004 ant brain as shown in lecture consisted of a simple FSM that implemented a “right antenna to the wall” algorithm for traversing a maze. This algorithm works for the simple mazes shown during lecture but fails if the maze contains an “island” where some inner wall of the maze is completely disconnected from the outer wall. Once the ant starts following the inner wall it will do so forever.

Your job is to help our ant step up the evolutionary ladder by changing the FSM shown lecture so that the ant can successfully navigate mazes with islands. To help in this task we’ve upgraded the ant hardware; the total array of sensors and actuators at the ant’s disposal now include:

- L** *left antenna sensor*: 1 if sensor is in contact with a wall, 0 otherwise
- R** *right antenna sensor*: 1 if sensor is in contact with a wall, 0 otherwise
- S** *smell sensor*: 1 if center of ant’s body is over some portion of the maze that has been previously scent-marked by the ant (see the M actuator).
- TL** *turn left actuator*: 1 causes the ant to rotate counterclockwise by 10 degrees, 0 means no left turn. Probably wouldn’t be asserted in the same cycle as TR.
- TR** *turn right actuator*: 1 causes the ant to rotate clockwise by 10 degrees, 0 means no right turn. Probably wouldn’t be asserted in the same cycle as TL.
- F** *step actuator*: 1 causes the ant to move forward one step, 0 means no forward motion. Can be asserted with either TL or TR, but any turns are completed before the ant moves forward. The ant will refuse to take a step if either antenna is touching a wall.
- M** *scent actuator*: 1 causes the ant to scent-mark the maze floor directly beneath the center of its body; 0 means no scent-marking takes place. The S sensor will detect this scent. Scent-marking happens before any forward motion. Probably wouldn’t be asserted in the same cycle as E.
- E** *erase actuator*: 1 causes any previous scent-marks directly beneath the center of the ant’s body to be erased; 0 means no erasing takes place. Erasing happens before any forward motion. Probably wouldn’t be asserted in the same cycle as M.

To run Roboant on a UNIX machine, type the following at the UNIX prompt

```
% roboant [filename]
```

You can supply an optional filename argument which will be loaded into the FSM editing buffer (you can load and save FSM files from within roboant too). If no argument is supplied, the buffer is initialized with the FSM shown in lecture. After roboant starts, you'll see a window with the FSM displayed on the left and the maze displayed on the right.

The FSM display shows the finite-state machine that will control the ant:

State table. The table is organized with one transition specified on each line. Blank and comment lines (lines beginning with “;”) are ignored. A transition specification has two parts: a pattern that will be matched against the current state and sensor readings of the ant, and an output field that indicates the name of the ant's next state and values for the various control signals that run the ant machinery. The syntax for a transition is

```
current_state L R S | next_state TL TR F M E
```

The states (current_state, next_state) are specified using symbolic names, which may be any sequence of letters or digits. Inputs (L, R, S) can be specified as either “0”, “1” which must match exactly the value of the corresponding sensor; or “-“ which indicates that any sensor value will match. Outputs (TL, TR, F, M, E) must be either “0” or “1”.

The maze display consists of the following parts:

Maze select radio buttons. Select which of the five mazes the ant should try to navigate.

Maze map: Shows the current position of the ant within the maze.

Speed control. This slider controls the speed of the animation when you press the “Run” button. At the fastest speed, no status updates are performed which leads to a much faster simulation.

“Reset” button. Reset the ant to its initial state (“lost”) and position.

“Step” button. Let the ant progress one state of the FSM.

“Run” button. Like “Step” except the ant will continue running the FSM until it reaches the goal square in the maze (marked with a cyan circle), the “Stop” button is pressed, or an error is detected.

“Stop” button. Stop the ant. You can proceed by pressing the “Step” or “Run” button.

At the bottom of the screen is a state display showing the ant's current state and sensor readings.

The ant operates by looking for a row in the state table that matches its current state and sensor values; it will complain if no rows match or if more than one row matches. The ant's state is changed to that specified in the “next-state” field of that row and any actuator actions are performed. This sequence is repeated until the ant reaches a specially marked (with a cyan circle) goal square.

- (A) Verify that everything is operating correctly by starting roboant, selecting maze “m1” and pressing the “Run” button. The ant should quickly follow the wall around to the goal square. There’s nothing to hand in from this step.
- (B) Select maze “m2” and press the “Run” button. Observe that the ant runs around the perimeter of the island indefinitely. Now modify the state table so that the ant can deal with mazes that have islands. Verify that the ant can traverse mazes “m1”, “m2”, “m3”, “m4” and “Hampton”.

Hint: it’s fun to try to figure this out on your own, but if you’re stumped here’s an outline of a solution. To keep the ant from making the same choices again and again, you can have it mark the maze as it travels. Whenever it completes a “wide” right-hand turn around a corner of the maze, if it’s standing on a marked spot, it should straighten itself out and pretend it’s lost – this will cause the ant to travel straight ahead until it hits a wall at which point the process begins all over. Eventually, by traveling straight ahead after some previously visited turn it will cross over from the island to the outer wall. There are still a lot of details to work out: for example, distinguishing “wide” turns around corners from the smaller turns made when traveling down a wall. A simple marking algorithm may not solve all the mazes... the ant may have to erase marks it has already made to keep from getting stuck inside a maze with a large number of wide right-hand turns.