CSC 334: Programming (sort of) Assignment 2: *RoboDeathMatch!*

Your Task:

Design an NFA to control a robot tank, that will duel with your classmates' tanks in a 15x15 discrete bounded rectangular grid. Each transition in your NFA diagram will have two items attached to it: 1) a sensor input (which specifies whether the transition may be chosen) and 2) an action output (which specifies what the robot should do if this transition edge is chosen). *(Technically, because of the action attached to the transitions, this is a nondeterministic finite state transducer, not an NFA. Details...)*

Input Alphabet: {"e", "B", "W", "R"}
"e" *(Epsilon)* transitions has a random chance of being used regardless of sensor input.
"B" *(Blank)* transitions will only be used if the sensor detects nothing (no wall, no enemy robots)
"W" *(Wall)* transitions will only be used if there is a wall *immediately* in front of your robot.
"R" *(Robot)* transitions will only be used if there is another (enemy) robot within 5 units (the range of your robo-sensor) directly straight in front of your robot.

Note: To keep things reasonable, each robot only gets to follow one transition and take one action during each time step. (You might expect that a robot using epsilon transitions could have many transitions happen and take many actions during a single turn. Not so...)

Output Alphabet (robot actions): {"L", "R", "F", "M", "N"}

"L" - the robot rotates itself 90 degrees *left* (counter-clockwise). The robot does not move.
"R" - the robot rotates itself 90 degrees *right* (clockwise). The robot does not move.
"F" - the tank moves *forward* 1 unit.
"M" - the tank launches a *missile*, which will move forward straight (in the direction the tank is facing) at twice the speed that the tanks move, until it hits a tank, another missile, or the boundary (where it will explode). Launching a missile when you are immediately facing into wall will damage your tank. Your robot tank can sustain the damage of 1 missile, but the 2nd missile that hits will destroy it.
"N" - The tank does *nothing* this turn.

The states of your NFA should be identified by the integers 0, 1, …, N-1. You should have a minimum of 5 states. There is no maximum for N, but keep in mind that you must also turn in a complete drawing/diagram of your NFA, which will be difficult if you use large numbers of states. For this assignment, there is no way to specify "accept" states, since what we care about here is the actions produced by your finite state machine, rather than whether a certain sequence of inputs is considered to be "in the robot's language" or not. Since this is an NFA, you may provide multiple transitions edges for the same specified sensor input. If you do this, the robot will choose between all valid transitions uniformly at random.

Be careful to use this precise file format:

```
YOUR-NAME
[
  [STATE0 [SENSOR ACTION NEW-STATE] [SENSOR ACTION NEW-STATE] ...]
  [STATE1 [SENSOR ACTION NEW-STATE] [SENSOR ACTION NEW-STATE] ...]
  ...
]
```

For example, here is a complete simple specification with just 2 states.

<<start-of-file named "forrest_rules.txt">>

```
Forrest
[
  [0 ["B" "F" 0] ["R" "M" 0] ["W" "L" 1] ]
  [1 ["B" "L" 0]  ["R" "M" 1] ["W" "L" 0] ]
]
```

<<end-of-file>>

The NFA above is actually just a DFA, since it has no epsilon transitions nor multiple edges for the same symbols. State 0 is always assumed to be the start state.

Take care, as any *typo, misspelling, or capitalization error* may prevent your tank from competing in the arena. Make sure to use quotes on the sensor/action tokens, but not on the state numbers! (And be sure to use standard quote characters, not "smart quotes" like Word creates.)

Here's another short example:

<<start-of-file named "iguana_rules.txt">>

```
Iguana
[
  [0 ["e" "F" 0] ["e" "L" 0] ["e" "R" 0] ["e" "M" 0] ["e" "M" 0] ["e" "M" 0]]
]
```

<<end-of-file>>

The iguana's theory is that unpredictability and firepower the most important factors...

I will provide you with the simulation testbed (via Moodle) so you can test things out on your own prior to the official RoboDeathMatch. Running it requires NetLogo 4.1.3, which you can download here: http://ccl.northwestern.edu/netlogo/ . If it's not on the lab computers, you can download and install it locally within your home folder. Ask me if you need any help.

Due date: Friday, 9/16/2011

Grading: Total of 5 points. *You will get 4.5/5 as long as your robot's NFA meets the specifications above, and your NFA diagram appears to be drawn correctly. Only the winner of the RoboDeathMatch will receive 5/5. (Cut-throat, I know...)*

Submission: Submit the assignment on Moodle. You should upload a zip file, named LastName_FirstName_Tanks.zip containing:

1. A text file named YourName-rules.txt
2. A PDF file named LastName_FirstName_Tanks.pdf
   a) Containing your NFA diagram, *and* a brief description (in English) of the strategy you were attempting to have your robot use.