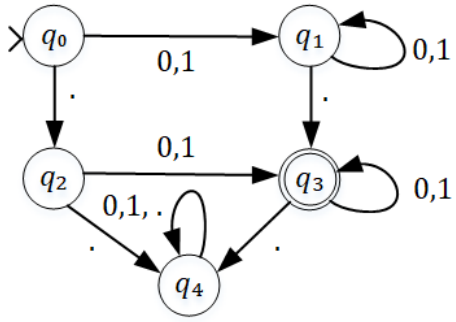
CS390 Principles of Programming Language

Assignment 7

Background:

In this assignment, we revist the deterministic finite automaton from the Topic 2 Assignment, which it given in the following figure and formally described below.



Recall, a ***Deterministic Finite Automaton*** (DFA) is as quintuple M over an alphabet Σ,

$$M = \langle Q, \Sigma, q_s, F, \delta \rangle$$

where Q is a set of ***states***, Σ is a set of ***alphabet*** symbols, $q_s \in Q$ is an ***initial*** state, F is a set of ***accepting*** (final) states, and δ is a ***transition function*** from a *current state* and *input alphabet symbol* to a *next state*, $\delta: Q \times \Sigma \to Q$. For example, the automaton depicted above is defined as,

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$
$$\Sigma = \{0,1,.\}$$
$$q_s = q_0$$
$$F = \{q_3\}$$
$$\delta = \begin{Bmatrix} \langle q_0, 0, q_1\rangle, \langle q_0, 1, q_1\rangle \langle q_0, ., q_2\rangle, \\ \langle q_1, 0, q_1\rangle, \langle q_1, 1, q_1\rangle \langle q_1, ., q_3\rangle, \\ \langle q_2, 0, q_3\rangle, \langle q_2, 1, q_3\rangle, \langle q_2, ., q_4\rangle, \\ \langle q_3, 0, q_3\rangle, \langle q_3, 1, q_3\rangle, \langle q_3, ., q_4\rangle, \\ \langle q_4, 0, q_4\rangle, \langle q_4, 1, q_4\rangle, \langle q_4, ., q_4\rangle \end{Bmatrix}$$

Functional Requirements:

1. Create a Haskell language program that can be used to execute any arbitrary using the above DFA as a specific example. Represent the DFA as a **five-tuple** corresponding to its formal definition where,
   a. represent each state with its name as a **string**
   b. represents all states as a **list** of states
   c. represent each transition as a **three-tuple** and the transitions as a list of these tuples.

2. To assist you, implement the following **functions** in your solution:

Name	Arguments	Description
dfaStateFactory	None	Returns a DFA 5-tuple definition
alphabet	DFA	Returns the given DFA's alphabet
allStates	DFA	Returns the given DFA's states
firstState	DFA	Returns the given DFA's first state
acceptStates	DFA	Returns the given DFA's accept states
allTransitions	DFA	Returns the given DFA's transitions list
transFromState	Transition	Returns the transition's from-state
transLabel	Transition	Return transition's label
transToState	Transition	Return transition's to-state
findTransition	CurrentState TransitionLabel TransitionList	Returns a list containing a single element, which is the transition matching the current state argument to the fromState element of the transition and the transition's label
findNextState	DFA Input	Returns a state matching the current input (internally, this must use the current state)
dfaAccept	DFA InputString	Returns true if the string is accepted by the DFA and false otherwise. Note: it may be helpful to implement this as a recursive function with a base case empty list and a non-empty list recursive call

3. Here are some examples executions your program should handle:

```
Prelude> dfaAccept stateFactory ""
False
Prelude> dfaAccept stateFactory "1"
False
Prelude> dfaAccept stateFactory "1.0"
True
Prelude> dfaAccept stateFactory "11.11"
True
Prelude> dfaAccept stateFactory "10.10.10"
False
```

4. You can assume the input string contains only characters in Σ (i.e. you don't have to validate the input, simply run the DFA with the input).

Non-Functional Requirements

1. Implement this assignment in a single Haskell file that executes using WinGHCi
2. Your program **must** be your own individual work. Any external assistance (i.e. reading material on the Internet, tutoring help from Regis, etc., must be cited with an

explanation of why you needed this external material and why you know understand it. Your faculty and the CS department reserve the right to reassess you, and, if necessary, re-determine your grade for this assignment, based on any additional assessments including an oral examination).

3. Do not use any predefined data structures for a DFA, (i.e.. create your own tuples and lists.)
4. Appropriately comment your program

Submission:
Submit your Haskell source file <yourName_Assignment_7>.hs to the Assignment 6 Dropbox in the Worldclass course shell associated with your current CS390 Section. (Although you will not earn points for testing, you should appropriately test your code for all requirements since you may not earn points, if it doesn't meet all the requirements and possible test input strings.)

CS 390 Principles of Programming Languages

Assignment 7 Rubric

Assignment 7 DFA Haskell program design and implementation.

Assignment	Exemplary	Advanced	Proficient	Not Demonstrated or Major Issues
DFA Definition and stateFactory			DFA tuple, states, first state, final state list, transition list and transition tuple correct	
			20	
Decomposition Functions (e.g. allStates, transInput, etc.)			All decomposition function appropriately defined and correctly implemented	
			20	
Execution and	Program executes correctly for all DFA test cases	DFA handle one or more test case inputs, but not all	findTransition and nextState appropriately defined and execute. DFA handles no input	
	60 – 46	45 – 31	30	
Deductions	Submitted on time Appropriately commented Executes using WinGHCi	Inappropriate comments 1-10% Compiles/Loads correctly	3% deducted per day late	**Not** submitted within six days of due date or **does not** compile