| | | |
|---|---|---|
| CS 39<br>Fall 2006<br>Theory of Computation | Homework 2<br>Due Oct 11, 2006 | Prof. Amit Chakrabarti<br>Computer Science Department<br>Dartmouth College |

Please think carefully about how you are going to organise your answers *before* you begin writing. Make sure your answers are complete, clean, concise and rigorous.

1. Let $L$ be the language over the alphabet $\{a, b\}$ given by the regular expression $(ab \cup aab \cup aba)^*$.

   1.1. Design an NFA for $L$ that has no $\varepsilon$-transtions and has only $4$ states. [6 points]

   1.2. Convert the above NFA into a DFA for $L$ by mechanically using the *subset construction* we studied in class. [10 points]

   1.3. Remove all states that are unreachable from the start state of the resulting DFA, to get a 7-state DFA for $L$. [3 points]

   1.4. If you carefully observe this DFA, you will notice two states that can be replaced by a single state. Do this and draw the resulting DFA. Your final DFA should have exactly $6$ states. [7 points]

2. Construct NFAs equivalent to following regular expressions (your NFAs may have $\varepsilon$-transitions):

   2.1. $10 \cup (0 \cup 11)0^*1$ [7 points]

   2.2. $((0 \cup 1)(0 \cup 1))^* \cup ((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$ [7 points]

3. Give regular expressions for the following languages.

   3.1. $\{w \in \{0, 1\}^* : w$ has three consecutive 0's or three consecutive 1's or both$\}$. [7 points]

   3.2. $\{w \in \{0, 1\}^* : w$ has three consecutive 0's and three consecutive 1's$\}$. [7 points]

   3.3. The set of strings in $\{0, 1\}^*$ with an equal number of 0's and 1's such that no prefix has two more 0's than 1's nor two more 1's than 0's. [10 points]

   3.4. Let us define a valid floating point number as $u.v$, where $u$ and $v$ are (finite) strings of decimal digits (0..9) satisfying the following constraints: (the symbol "." between $u$ and $v$ is the decimal point.)

      i. Neither $u$ nor $v$ may be $\varepsilon$.
      ii. $u$ can be just 0. If $u$ is not 0, $u$ has no leading 0's.
      iii. $v$ can be just 0. If $v$ is not 0, $v$ has no trailing 0's.

      (Thus, for example, 0.0, 231.0 and 5.608 are valid, but 0.00, 05.68, .65, 12. and 4.5100 are not valid.)

      Give a regular expression for the set of valid floating point numbers described above. You might want to introduce some notation first to keep your expression small and readable. [10 points]

4. Let $L$ be a nonempty language and $M$ an NFA that recognizes $L$. Prove that $M$ can be converted into an NFA $M'$ which recognizes the same language $L$ and has exactly one accept state. Your proof *must* describe $M'$ both informally, using plain English, *and* formally, using mathematical notation. [10 points]

5. For a language $L$ over alphabet $\Sigma$, define $\textsc{Half}(L) = \{x \in \Sigma^* : \exists\, y \in \Sigma^* \, (|x| = |y| \text{ and } xy \in L)\}$. Prove that if $L$ is regular, then so is $\textsc{Half}(L)$. Your proof *must* be formal; proofs not written in a formal mathematical style get very little credit even if they express the right intuition. [16 points]

   Hint: Approach 1: Build an NFA. Suppose $x$ is the input string. Nondeterministically guess which state the DFA for $L$ will end up in after reading $x$ and nondeterministically guess a $y$ to append to $x$ as in the definition of $\textsc{Half}(L)$. Approach 2: Build a DFA. As you read $x$, work forwards and backwards simultaneously and try to meet in the middle.

**Challenge Problems**

Remember that challenge problems carry no regular credit, but are intended to provide a higher level of challenge for those who want to think further about the theory of computing.

**CP1:** For the language $L$ from Problem 1, prove that it is impossible to design a DFA with $5$ or fewer states.

**CP2:** For a language $L$ over alphabet $\Sigma$, define $\textsc{Log}(L) = \{x \in \Sigma^* : \exists\, y \in \Sigma^* \, (|y| = 2^{|x|} \text{ and } xy \in L)\}$. Prove that if $L$ is regular, then so is $\textsc{Log}(L)$.