School of Mathematical and Computing Sciences
COMP 202 : Formal Methods of Computer Science

Test — Solutions

1. A string s is a *common prefix* of strings t and u iff s is a prefix of t and also a prefix of u; i.e. $t = st'$ and $u = su'$ for some strings t' and u'.

   s is the *longest common prefix* of t and u iff s is a common prefix of t and u and there is no string s' such that s' is a common prefix of t and u and $|s'| > |s|$.

   (a) Prove the following properties of common prefixes:

   (i) Every pair of strings s and t has at least one common prefix. [2 marks]

   λ is a prefix of any string (i.e. $\lambda s = s$, for any string s), so λ is a common prefix for any pair of strings.

   (ii) If s is a common prefix of t and u, and v is a common prefix of $s\backslash t$ and $s\backslash u$, then $s \frown v$ is a common prefix of t and u. [4 marks]

   If s is a common prefix of t and u, there are strings t' and u' such that $t = st'$ and $u = su'$, i.e. $t' = s\backslash t$ and $u' = s\backslash u$.
   Now, if v is a common prefix of $s\backslash t$ and $s\backslash u$, there must be strings t'' and u'' such that $s\backslash t = vt''$ and $s\backslash u = vu''$.
   But since $t' = s\backslash t$ and $u' = s\backslash u$, this means (substituting in $t = st'$ and $u = su'$), $t = svt''$ and $u = svu''$.
   Thus, sv is a common prefix of t and v.

   (b) Explain how the above properties can be used to construct an algorithm to find the longest common prefix of two strings, s and t.
   Write the resulting algorithm as a functional program, using *head*, *tail*, *NULL* and $\frown$ to access and manipulate strings. [6 marks]

   These properties mean that we can find the longest common prefix, p, of s and t by starting with $p = \lambda$ (which is the always the shortest common prefix) and adding characters from the front of s and t to p, until $p\backslash s$ and $p\backslash t$ have no common prefix other than λ (in which case either $p\backslash s = \lambda$, $p\backslash t = \lambda$ or $head(p\backslash s) \neq head(p\backslash t)$).
   As a functional program:

$$
\begin{array}{lcl}
lcp(s,t) & \triangleq & \textbf{if } s = NULL \ \vee \ s = NULL \textbf{ then } NULL \\
 & & \textbf{elsif } head(s) \neq head(t) \textbf{ then } NULL \\
 & & \textbf{else } \langle head(s) \rangle \frown lcp(tail(s), tail(t)) \textbf{ fi}
\end{array}
$$

   or:

$$
\begin{array}{lcl}
lcp(s,t) & \triangleq & lcp'(s,t,NULL) \\
lcp'(s,t,u) & \triangleq & \textbf{if } s = NULL \ \vee \ s = NULL \textbf{ then } u \\
 & & \textbf{elsif } head(s) \neq head(t) \textbf{ then } u \\
 & & \textbf{else } lcp'(tail(s), tail(t), u \frown \langle head(s) \rangle) \textbf{ fi}
\end{array}
$$

2. (a) Write a Regular Expression to describe the language consisting of all strings over $\{a, b, c\}$ which contain *either* exactly one a or exactly two bs.

   For example, a, bb, bab, $ccbbbabc$ and $acacbcacba$ are in this language, but λ, b, aba and ccc are not. [4 marks]

   $$(b|c)^*a(b|c)^* \mid (a|c)^*b(a|c)^*b(a|c)^*$$

   (b) Write a Regular Expression to describe the language consisting of all strings over $\{a, b, c\}$ for which all of the following conditions hold:

   - all of the as and bs occur before all of the cs,
   - if the first a occurs *before* the first b, there must be an *even* number of cs, and
   - if the first a occurs *after* the first b, there must be an *odd* number of cs.

   For example, λ, a, b, c, cc, ac, bcc, $abcc$, bac, $abaacccccc$ and $bbaabccc$ are in this language, but ba, abc, $abccc$, $bacc$ and $babababacccc$ are not. [4 marks]

   $$a^*c^* \mid b^*c^* \mid aa^*b(a|b)^*(cc)^* \mid bb^*a(a|b)^*c(cc)^*$$

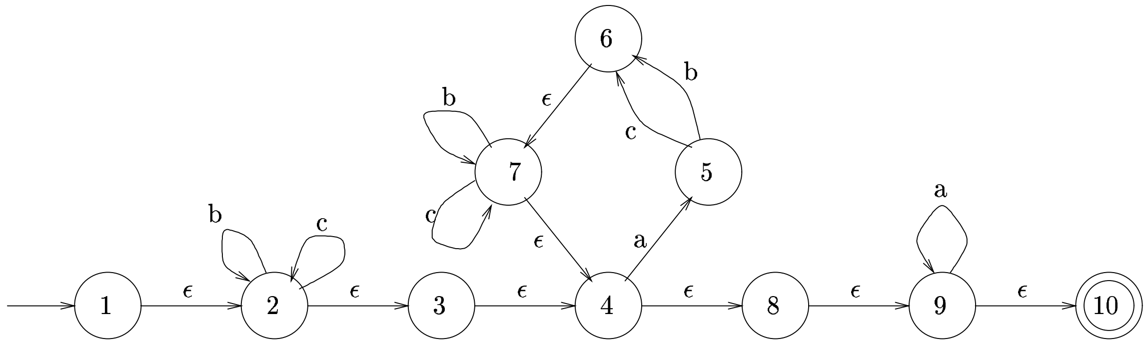
   or

   $$(a^* \mid b^*)c^* \mid a(a|b)^*(cc)^* \mid b(a|b)^*c(cc)^*$$

3. Consider the regular expression $(b|c)^*(a(b|c)(b|c)^*)^*a^*$.

   Draw a transition diagram for the NFA obtained by applying the "top down" construction (as described in the Course Notes).

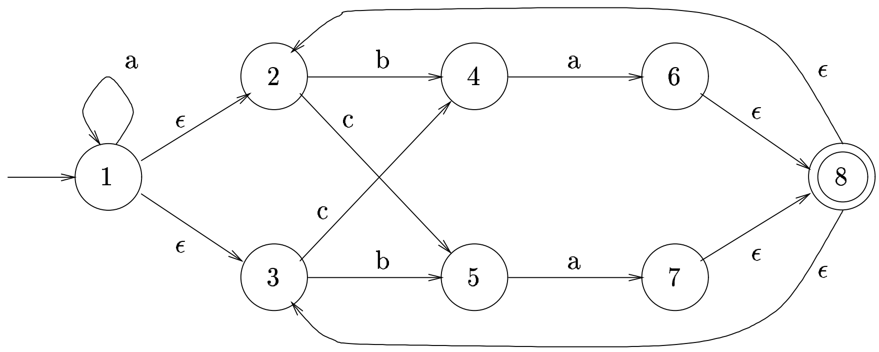
   Make sure you show *all* states and edges given by the construction! [6 marks]



4. Consider the NFA $M = (\{1, 2, 3, 4, 5, 6, 7, 8\}, 1, \{a, b, c\}, N, \{8\})$, where N is defined by the following transitions:

   $$(1, a; 1), (1, \epsilon; 2), (1, \epsilon; 3), (2, b; 4), (2, c; 5), (3, b; 5), (3, c; 4),$$
   $$(4, a; 6), (5, a; 7), (6, \epsilon; 8), (7, \epsilon; 8), (8, \epsilon; 2), (8, \epsilon; 3)$$

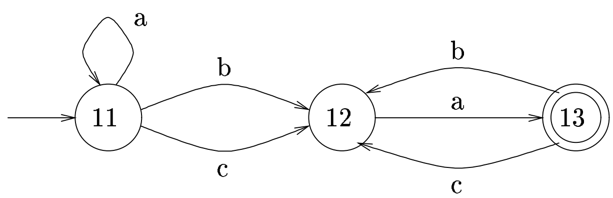
   (a) Draw a transition diagram depicting this NFA. [2 marks]



(b) Give a trace showing the behaviour of the the NFA with *aabaca* as input.
Be sure to show *all* states the NFA could be in at each step. [4 marks]

States	Input	
$1, 2, 3$	*aabaca*	
$1, 2, 3$	*abaca*	
$1, 2, 3$	*baca*	
$4, 5$	*aca*	
$6, 7, 8, 2, 3$	*ca*	
$4, 5$	*a*	
$6, 7, 8, 2, 3$	λ	*Accept*

(c) Construct an equivalent DFA, using the subset construction.
Show the relationships between the states in your DFA and those of the NFA. [6 marks]



DFA state	NFA states
11	$\{1, 2, 3\}$
12	$\{4, 5\}$
13	$\{2, 3, 6, 7, 8\}$

(d) Give the equations relating the regular expressions denoting the *From* sets for the states in your DFA.
Solve these equations to obtain a regular expressions denoting the *From* set for each state in your DFA. [6 marks]

Let E, F and G be the RE's corresponding to states 11, 12 and 13, respectively. The equations are:

$$E = a\,E \mid b\,F \mid c\,G$$
$$F = a\,G$$
$$G = \lambda \mid b\,F \mid c\,F$$

Solving these equations (solving for G first), we get:

$$G = \lambda \mid (b|c)\ F \;=\; \lambda \mid (b|c)\ a\ G \;=\; ((b|c)\ a)^*$$
$$F = a\ ((b|c)\ a)^*$$
$$E = a\ E \mid (b|c)\ F \;=\; a^*\ (b|c)\ a\ ((b|c)\ a)^*$$

Solving for F first leads to the alternative (but equivalent) solutions:

$$
\begin{aligned}
F &= (a\ (b|c))^*\ a \\
G &= \lambda \mid (b|c)\ (a\ (b|c))^*\ a \\
E &= a^*\ (b|c)\ (a\ (b|c))^*\ a
\end{aligned}
$$

5. Consider the language L containing all strings over $\{a, b, c\}$, in which there are an equal number of as and bs.

   Prove that L is not regular, using the Myhill-Nerode Theorem. [6 marks]

   Consider the set $D = \{ a^k \mid k \geq 0 \}$.

   If a^i and a^j are distinct members if D (i.e. $i \neq j$), then a^i and a^j must be distinguished (with respect to L), since $a^i b^i \in L$ (because $a^i b^i$ has an equal number of as and bs) and $a^j b^i \notin L$ (because $a^j b^i$ does not have an equal number of as and bs, as $i \neq j$).

   Since D is infinite (it has a unique element corresponding to every natural number) and any two members of D must be distinguished (with respect to L), L is not regular.