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

Test — Solutions

1. (a) Write a Regular Expression to describe the language in which every string consists of either an even number of as followed by an odd number of bs or an odd number of as followed by an even number of cs. [4 marks]

   $(aa)^*b(bb)^*|a(aa)^*(cc)^*$ or $(aa)^*(b(bb)^*|a(cc)^*)$

   (b) Write a Regular Expression to describe the language consisting of all strings of as, bs and cs, where as occur in multiples of two, bs occur in multiples of three, and consecutive groups of as or bs are separated by one of more cs. For example, "aa", "$cbbb$", "$cccc$", "$aacbbbcccaaaac$" and "$aaccaacbbbcbbbbbb$" are in this language, but "a", "bb", "aca" and "$aaab$" are not. [4 marks]

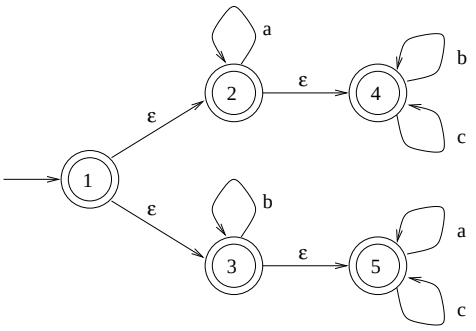
   $((aa)^*|(bbb)^*)(c((aa)^*|(bbb)^*))^*$ or $(((aa)^*|(bbb)^*)c)^*((aa)^*|(bbb)^*)$

   (c) Write an Extended Regular Expression to describe the language consisting of lists of one or more *items*, separated by commas, where each *item* is either a number (a non-empty sequence of digits), two numbers separated by a colon, or three numbers separated by colons. For example, "123", "$1, 2, 3$" and "$1, 2: 3, 23:1:52, 1$" are in this language, but "$1, , 2$", "$1:2,$" and "$1:2:3:1$" are not. You may write d to stand for any digit. [4 marks]

   $d^+[:d^+]^2(,d^+[:d^+]^2)^*$ or $d^+(:d^+)_0^2(,d^+(:d^+)_0^2)^*$

2. Consider the regular expression $E = a^*(b|c)^* \mid b^*(a|c)^*$.

   (a) Draw a transition diagram for the simplest NFA you can construct to recognise the language defined by E. [4 marks]



   Note that states 1, 2 and 3 need not be marked as being final states.

(b) Give a trace showing the behaviour of your DFA with *aacabc* as input.

[2 marks]

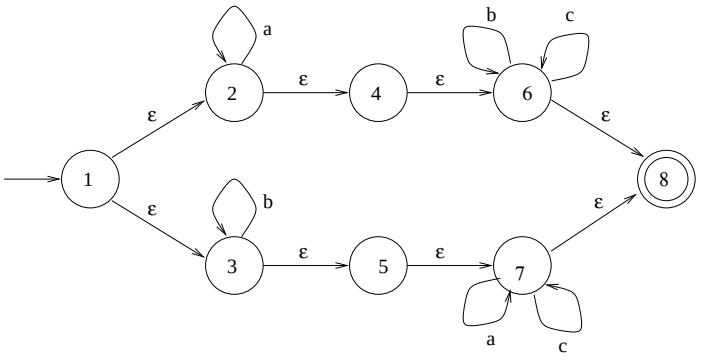
States	Input	
1,2,3,4,5	*aacabc*	
2,4,5	*acabc*	
2,4,5	*cabc*	
4,5	*abc*	
5	*bc*	Reject

(c) Write a left-regular grammar equivalent to E. [3 marks]

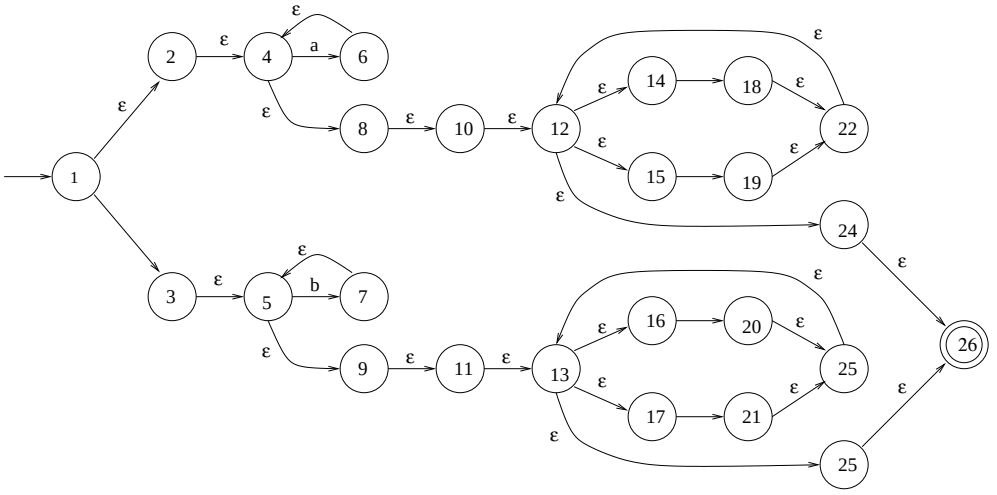
$S \rightarrow T \mid U$
$T \rightarrow aT \mid V$
$V \rightarrow bV| \mid cV \mid \lambda$
$U \rightarrow bU \mid W$
$W \rightarrow aW \mid cW \mid \lambda$

(d) Draw a transition diagram for the NFA obtained by applying **either** the "top down" **or** the "bottom up" construction (as described in the Course Notes) to E. [4 marks]

Top down:

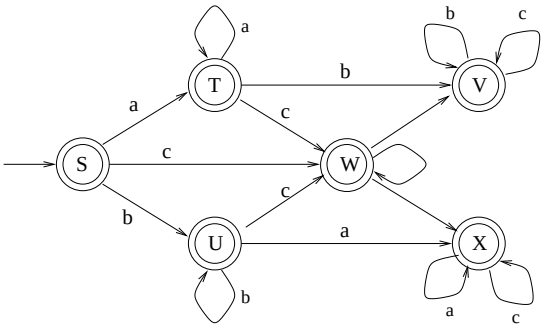


Bottom up:



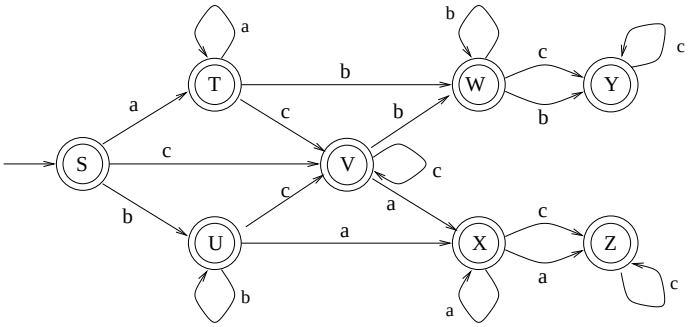
(e) Draw a transition diagram for the DFA obtained by applying the subset construction to your NFA from part (d). Show the correspondence between states in your DFA and sets of states in the NFA. [4 marks]

Top down:



DFA States	NFA States
S	$\{\underline{1}, 2, 3, 4, 5, 6, 7, 8\}$
T	$\{\underline{2}, 4, 6, \underline{7}, 8\}$
U	$\{\underline{3}, 5, \underline{6}, 7, 8\}$
V	$\{\underline{6}, \underline{7}, 8\}$
W	$\{\underline{6}, 8\}$
X	$\{\underline{7}, 8\}$

Bottom up:



DFA States	NFA States
S	$\{\underline{1}, 2, 3, 8, 10, 11, 12, 13, 14, 15, 16, 17, 24, 26\}$
T	$\{4, \underline{6}, 8, 10, 12, 13, 14, 15, 17, \underline{20}, 23, 24, 25, 26\}$
U	$\{\underline{5}, 9, 11, 12, 13, 14, 15, 16, 17, \underline{18}, 22, 25\}$
V	$\{12, 13, 14, 15, 16, 17, \underline{19}, \underline{21}, 22, 24, 225, 26\}$
W	$\{12, 14, 15, \underline{18}, 22, 24, 26\}$
X	$\{13, 16, 17, \underline{20}, 23, 25, 26\}$
Y	$\{12, 14, 15, \underline{19}, 22, 24, 26\}$
Z	$\{13, 16, 17, \underline{21}, 23, 25, 26\}$

The underlined states are ones reached by a non-null transition; the others are all reachable from these by following null transitions.

(f) Identify any states in your DFA in part (e) that can be eliminated to obtain a smaller DFA. [1 marks]

For the top down dfa, no states can be eliminated.

For the bottom up dfa, states W and Y can be merge, so can states X and Z.

3. Consider the language PAL of palindromes over $\{a, b, c\}$, i.e. the set of all strings $\alpha \in \{a, b, c\}^*$ such that $\alpha^R = \alpha$.

   (a) Prove that PAL is not regular, using the Myhill-Nerode Theorem. [4 marks]

   Consider two strings, a^m and a^n, where $m \leq n$. These strings must be distinguished (relative to PAL) because $a^m b a^m \in PAL$ (since $(a^m b a^m)^R = a^m b a^m$), but $a^n b a^m \in PAL$ (since $(a^n b a^m)^R = a^m b a^n \neq a^n b a^m$, as $m \neq n$).

   Now consider the set $S = \{a^i \mid i \geq 0\}$. It follows from the above that any two strings in S must be distinguished.

   Since we have an infinite set of strings, any two of which must be distinguished (relative to PAL), so PAL is not regular.

   Note that it is **not** *sufficient to show that there is an infinite number of pairs of strings that must be distinguished! If we consider the language* $L = \{ a^i \mid i \geq 0 \} \cup \{ b^i \mid i \geq 0 \}$, *there are an infinite number of pairs* (a^i, b^i) *that must be distinguished, but* L *is clearly regular since it is denoted by the regular expression* $a^*|b^*$.

   (b) Write a Context Free Grammar that defines PAL. Give a brief explanation of why your grammar defines the required language. [4 marks]

   $S \rightarrow \lambda$ (1) $\mid a$ (2) $\mid b$ (3) $\mid c$ (4) $\mid aSa$ (5) $\mid bSb$ (6) $\mid cSc$ (7)

   Note that PAL *includes palindromes of both even and odd length! Omitting rule (1) gives only palindromes of odd length; omitting rules (2), (3) and (4) gives only palindromes of even length.*

   To show that this grammar defines PAL, we show that (i) every parse tree from S has a palindrome as its fringe (i.e. $\mathcal{L}(S) \subseteq PAL$), and (ii) every palindrome can be parsed with this grammar (i.e. $PAL \subseteq \mathcal{L}(S)$).

   To simplify the argument we introduce the following piece of terminology. We say a parse tree *produces* a string α if it has α as its fringe.

   (i) We prove that every parse tree produces a palindrome by induction on the height of the parse tree.

   Base: Every parse tree of height 1 produces either λ, a, b or c (using rules (1), (2), (3) and (4), respectively). Since these are all palindromes, every parse tree of height 1 produces a palindrome.

   Step: Assume that every parse tree of height h (where $h \geq 1$) produces a palindrome. Suppose T is a parse tree of height h with fringe α (so α must be a palindrome). We can extend T to a parse tree of height $h + 1$ only by adding an application of rule (5), (6) or (7) above the root. In each case the fringe of the new tree is $x\alpha x$, where $x \in \{a, b, c\}$, which is a palindrome since α is a palindrome.

   Thus, every parse tree produces a palindrome.

   (ii) We prove every palindrome can be parsed with the above grammar by induction on the "half-length" of the palindrome, where the "half-length" of a string α is $|\alpha| \bmod 2$ (or $\lfloor |\alpha|/2 \rfloor$).

   Base: If α is a palindrome with half-length 0, then $|\alpha|$ is 0 or 1, and α is either λ, a, b or c. We can construct a parse tree for α with one application

of rule (1), (2), (3) or (4). Thus, every palindrome with half-length 0 can be parsed with the grammar.

Step: Assume that every palindrome with half-length k (for $k \geq 0$) can be parsed with the grammar. If α is a palindrome with half-length $k+1$, there must be a string β and a symbol $x \in \{a, b, c\}$ such that $\alpha = x\beta x$, where β is a palindrome. By the inductive hypothesis, there must be a parse tree, T, for β. We can therefore construct a parse tree for α by adding an application of rule (5), (6) or (7) above the root (according to whether x is a, b or c).

Thus, every palindrome can be parsed with the grammar.

I didn't expect a proof in as much detail as this. Note that it is important to show both parts of the proof: just showing that every sentence produced by the grammar is a palindrome does not guarantee that the grammar can produce all palindromes!

4. Well-formed formulas of propositional logic (known as *wff*s) can be defined as follows:

   (i) The propositional constants, $true$ and $false$, are both wffs.

   (ii) Every propositional variable is a wff.

   (iii) If X and Y are wffs, then $\neg X$, $X \wedge Y$, $X \vee Y$, $X \Rightarrow Y$, $X \equiv Y$ and (X) are also wffs.

   (iv) Nothing else is a wff.

   In order to avoid ambiguity, we adopt the following conventions for the precedence and associativity of the propositional connectives:

Connective(s)	Precedence	Associativity
$\Rightarrow$, $\equiv$	1	Non-associative
$\vee$	2	Left-associative
$\wedge$	3	Left-associative
$\neg$	4	Associative

   As usual, parentheses are treated as having higher precedence that any of the propositional connectives.

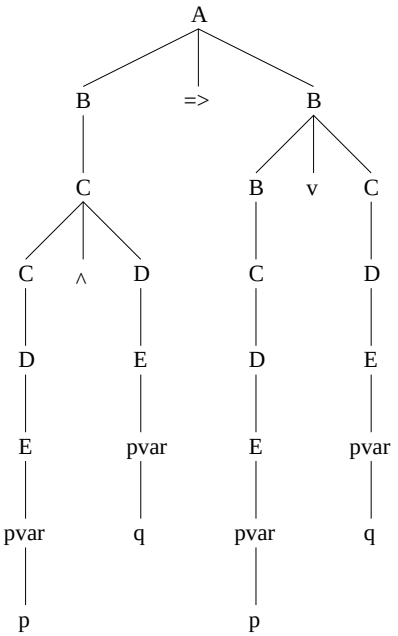
   (a) Write a Context Free Grammar for wffs, which will give every wff a unique parse tree reflecting these conventions for precedence and associativity. [8 marks]

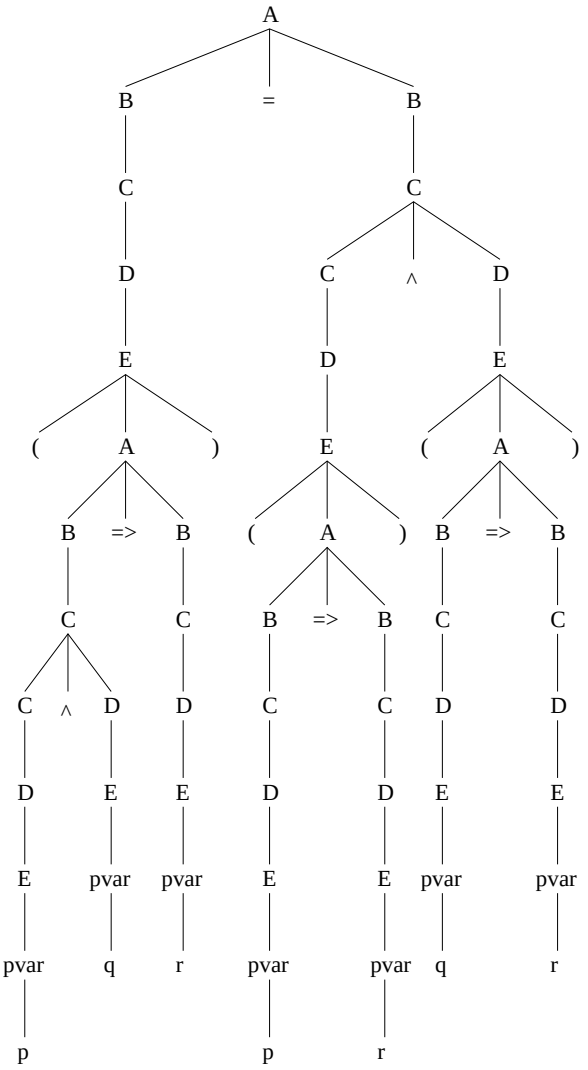
   $A \rightarrow B \mid B \Rightarrow B \mid B \equiv B$
   $B \rightarrow C \mid B \vee C$
   $C \rightarrow D \mid C \wedge D$
   $D \rightarrow E \mid \neg D$
   $E \rightarrow pvar \mid true \mid false \mid (A)$

   (b) Draw a parse tree for each of the following wffs (where p, q and r are assumed to be propositional variables): [4 marks]

   (i) $p \wedge q \Rightarrow p \vee q$

   (ii) $(p \wedge q \Rightarrow r) \equiv (p \Rightarrow r) \wedge (q \Rightarrow r)$


A
B
=>
B
C
B
v
C
C
^
D
C
D
D
E
D
E
E
pvar
E
pvar
pvar
q
pvar
q
p
p


A
B
=
B
C
C
D
C
^
D
E
D
E
(
A
)
E
(
A
)
B
=>
B
(
A
)
B
=>
B
C
C
B
=>
B
C
C
C
^
D
D
C
C
D
D
D
E
E
D
D
E
E
E
pvar
pvar
E
E
pvar
pvar
pvar
q
r
pvar
pvar
q
r
p
p
r