TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2007

MID-TERM TEST

COMP/SWEN 202
Formal Foundations of Computer Science and Software Engineering
WITH ANSWERS

Time Allowed: 90 minutes

Instructions: There are **four** (4) questions.
Answer **all** the questions.
Show **all** your working.

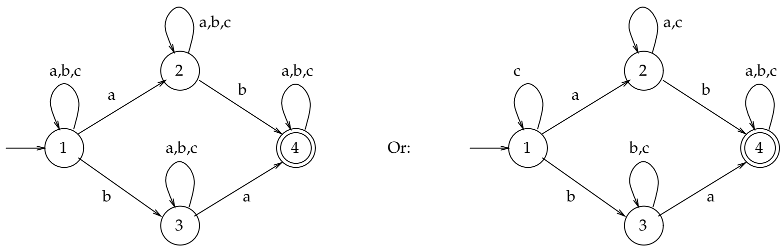
Question 1. [15 marks]

For each the following languages described below, **(i)** write a regular expression that defines the language, and **(ii)** draw a transition diagram for a (nondeterministic) finite acceptor that recognises the language:

(a) The set of all strings over $\{a, b, c\}$ containing at least one a and at least one b.

(i) $(a|b|c)^*(a(a|b|c)^*b \mid b(a|b|c)^*a)(a|b|c)^*$ or: $c^*(a(a|c)^*b \mid b(b|c)^*a)(a|b|c)^*$

(ii)



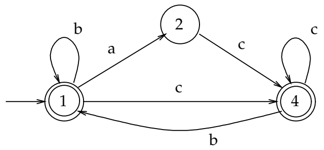
The first RE, and the first FA, can be understood as encoding a description of this language as the set of all strings $\alpha \in \{a, b, c\}^*$ such that $\alpha_i = a$ and $\alpha_j = b$ for some $i, j \in 1 .. |\alpha|$. Obviously i and j must be different (since a and b are different), so we must have either $i < j$ or $j < i$, which means either $\alpha = \beta\, a\, \gamma\, b\, \delta$ or $\alpha = \beta\, b\, \gamma\, a\, \delta$, for some strings $\beta, \gamma, \delta \in \{a, b, c\}^*$.

The second RE, and the second FA, which is deterministic, are simpler in that they have fewer symbols/transitions, but it is no so obvious that they are correct — to understand them you have to think about traversing the string from left to right remembering whether you've seen an a and/or a b.

(b) The set of all strings over $\{a, b, c\}$ in which every occurrence of a is immediately followed by a c, and no occurrence of c is immediately followed by an a.

(i) $(b|(ac|c)c^*b)^*(\lambda|(ac|c)c^*)$

(ii)



This is a case where drawing a transition diagram is far easier that writing an RE, and the best way to obtain an RE is to drawing a transition diagram first and derive the RE from it. The above RE is equivalent to the one that JFLAP produces for this NFA.

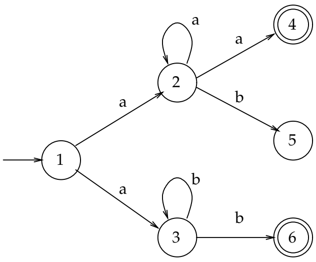
Question 2.

[15 marks]

Consider the NFA $M = (Q, q_I, A, N, F)$, where:

- $Q = \{1, 2, 3, 4, 5, 6\}$
- $q_I = 1$
- $A = \{a, b\}$
- $N(1, a) = \{2, 3\}$,
  $N(2, a) = \{2, 4\}$,
  $N(2, b_) = \{5\}$,
  $N(3, b) = \{3, 6\}$,
  $N(q, x) = \{\}$, otherwise
- $F = \{4, 6\}$

(a) Draw a transition diagram for M.



(Note that 5 is a useless state.)

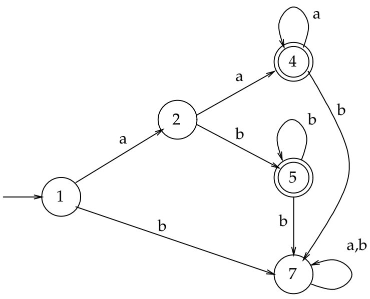
(b) Describe, in English, the language recognised by M.

All strings consisting of either two or more a's or an a followed by one or more b's.

Or:

All strings consisting of an a followed by either one or more a's or one or more b's.

(c) Draw a transition diagram for a complete DFA equivalent to M.



(d) Write a regular expression which defines the language recognised by M.

$a\,(a\,a^* \mid b\,b^*)$

Question 3. [20 marks]

Let M_1 and M_2 be two NFAs, where $M_i = (Q_i, q_{I_i}, A_i, N_i, F_i)$ for $i = 1, 2$.

(a) Explain, in English, how M_1 and M_2 can be combined to obtain an NFA that recognises $L_1 \cup L_2$, where L_1 and L_2 are the languages recognised by M_1 and M_2, respectively.

Add a new state, say q_0, make it the initial states, and add null transitions from it to the initial states of M_1 and M_2.

(b) Give a mathematical definition of the NFA described in part (a), and give a brief argument explaining why this NFA recognises $L_1 \cup L_2$.

Let the new NFA be $M_3 = (Q_3, q_{I_2}, A_2, N_3, F_3)$, then we define the components of M_2 as follows:

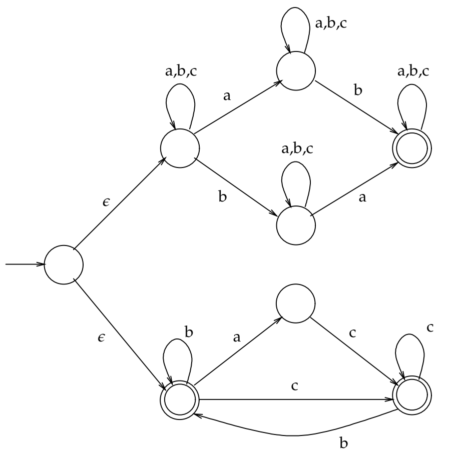
- $Q_3 = Q_1 \cup Q_2 \cup \{q_0\}$, where $q_0 \notin Q_1 \cup Q_2$
- $q_{I_3} = q_0$
- $A_3 = A_1 \cup A_2$
- $N_3 = N_1 \cup N_2 \cup \{(q_0, \epsilon, q_{I_i}) \mid i = 1, 2\}$
- $F_3 = F_1 \cup F_2$

Proof:

If a string α is in $L_1 \cup L_2$, then α is accepted by M_i for $i = 1$ or 2, which means that M_i can move from q_{I_i} to a state, say q_F in F_i while consuming α. But in that case, M_3 can move from q_0 to q_F in F_i while consuming α (since it can take a null transition from q_0 to q_{I_i} and then move from q_{I_i} to q_F while consuming α, by taking exactly the same transitions as M_i), so M_3 accepts α.

If M_3 accepts a string α, then M_3 can move from q_0 to a state, say q_F in F_3 while consuming α. Now the first transition that M_3 takes must be a null transition to q_{I_i}, for $i = 1$ or 2. The rest of the transitions that M_3 takes must all be transitions of M_i and q_F must be in F_i. Thus, M_i can move from q_{I_i} to a state in F_i while accepting α, so M_i accepts α. Since α is accepted by either M_1 or M_2, we have $\alpha \in L_1 \cup L_2$.

(c) Draw a transition diagram for the NFA obtained by applying this construction to the two NFAs you drew for Question **1**.



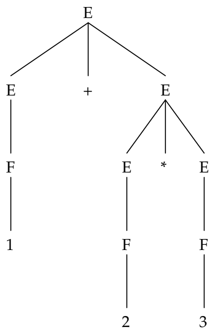
Question 4. [20 marks]

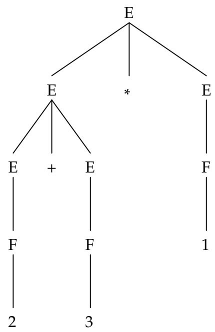
Consider the following grammar:

$$
\begin{aligned}
E &\rightarrow F \mid E * E \mid E + E \\
F &\rightarrow 1 \mid 2 \mid 3 \mid (\ E - E\ )
\end{aligned}
$$

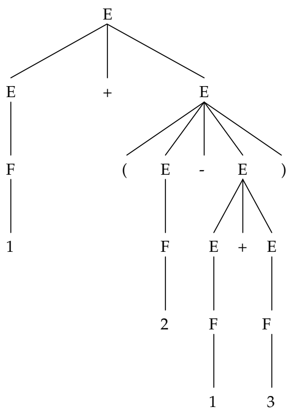
(a) Draw a parse tree for the string $1 + 2 * 3$.

We can construct two parse trees:





(b) Draw a parse tree for the string $1 + (2 - 1 + 3)$.



(c) Explain, giving an example, why this grammar is ambiguous.

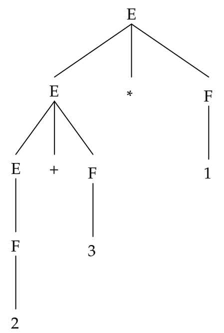
A grammar is ambiguous it is it possible to construct two distinct parse trees for any string. In part (a), we have shown two different parse trees for $1 + (2 - 1 + 3)$, so this demonstrates that the grammar is ambiguous. In general, and grammar with a rule of the form $N \rightarrow N \oplus N$ is ambiguous.

(d) Show how you would construct an equivalent unambiguous grammar by treating $*$ and $+$ as *left associative*.

To make an operator $\oplus$ left associative, we use a rule of the form $N \rightarrow M \mid N \oplus M$, which means that in an expression of the form $x \oplus y \oplus z$, the subexpression $x \oplus y$ is deeper in the tree, so the expression in naturally interpreted in the same way as $(x \oplus y) \oplus z$. Thus, we rewrite the grammar as:

$$
\begin{array}{lcl}
E & \rightarrow & F \mid E * F \mid E + F \\
F & \rightarrow & 1 \mid 2 \mid 3 \mid (\ E - E\ )
\end{array}
$$

Now, the only parse tree we can construct for $1 + 2 * 3$ is:

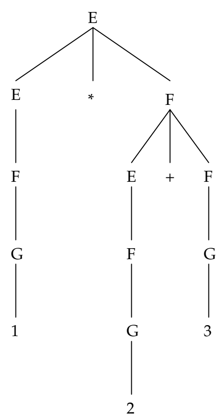


(e) Show how you would construct an equivalent unambiguous grammar by treating $*$ as having *higher precedence* (i.e. as binding more tightly) than $+$.

To give $*$ higher precedence than $+$, we add a new "layer" in the grammar, along with a new nonterminal. This forces $*$ to appear deeper in a parse tree, so expressions of the form $1 + 2 * 3$ and $1 * 2 + 3$ are naturally interpreted in the same way as $1 + (2 * 3)$ and $(1 * 2) + 3$. Thus, we rewrite the grammar as:

$$
\begin{array}{lcl}
E & \rightarrow & F \mid E + F \\
F & \rightarrow & G \mid F * G \\
G & \rightarrow & 1 \mid 2 \mid 3 \mid (\ E - E\ )
\end{array}
$$

Now, the only parse tree we can construct for $1 + 2 * 3$ is:



In parts (d) and (e), you should give the new grammar, explain how and why you have changed it, and show how the new grammar addresses the example you used in part (c).

****