CS21004 Formal Languages and Automata Theory, Spring 2010–11

End-Semester Test

Maximum marks: 50 Date: April 2011 Duration: 3 hours

Roll no: ____________ **Name:** ______________________________

[*Write your answers in the question paper itself. Be brief and precise. Answer* <u>*all*</u> *questions.*]

1. Let L be the language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains an equal number of occurrences of } ab \text{ and } ba\}.$$

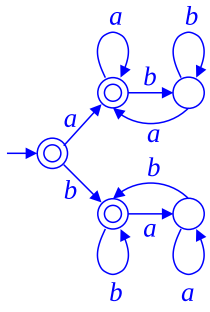
For example, $ababa \in L$ (two occurrences of ab, and two of ba), whereas $bbaba \notin L$ (one occurrence of ab, and two of ba).

(a) Give a regular expression whose language is L. **(5)**

Solution L is the language of the regular expression $\epsilon + aa^*(bb^*aa^*)^* + bb^*(aa^*bb^*)^*$.

(b) Design a DFA/NFA/ϵ-NFA to accept L. **(5)**

Solution The following DFA accepts L.

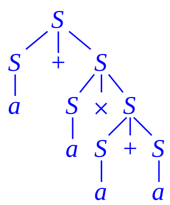


2. Consider the following context-free grammar to generate arithmetic expressions in one variable a, involving addition and multiplication operations only. Here, S is the start symbol.

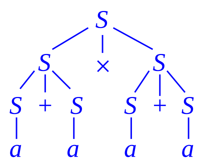
$$S \rightarrow a \mid S + S \mid S \times S$$

(a) Draw *all* the parse trees for the string $a + a \times a + a$ following this grammar. **(5)**

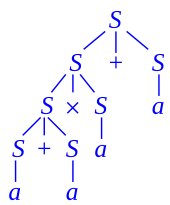
Solution There are five parse trees for $a + a \times a + a$. These trees are shown below. They respectively stand for the following interpretations of the given expression: $a + (a \times (a + a))$, $(a + a) \times (a + a)$, $((a + a) \times a) + a$, $a + ((a \times a) + a)$ and $(a + (a \times a)) + a$.



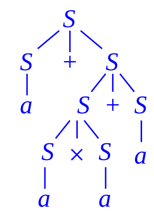
(a) First tree



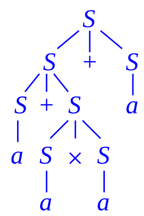
(b) Second tree



(c) Third tree



(d) Fourth tree



(e) Fifth tree

(b) Design an unambiguous grammar to generate the same language. Force the operations to be evaluated from left to right (like in some digital calculators). This means that $+$ and $\times$ are given the same precedence and left-to-right associativity. For example, $a + a \times a + a$ is to be interpreted as $((a + a) \times a) + a$. **(5)**

Solution The following grammar generates the same language, but forces left-to-right evaluation of the operations.

$$S \rightarrow a \mid S + a \mid S \times a.$$

3. Consider the following unrestricted (Type 0) grammar with the start symbol S, with non-terminal symbols S, A, B, C, and with terminal symbols a, b, c.

$$\begin{array}{llllll} S \to ABCS \mid \epsilon, & & & & & \\ AB \to BA, & BA \to AB, & AC \to CA, & CA \to AC, & BC \to CB, & CB \to BC, \\ A \to a, & B \to b, & C \to c. & & & \end{array}$$

(a) Show a derivation of the string $babacc$ using this grammar. Show each individual step in the derivation, and mention the rule used in each step. **(5)**

Solution A derivation of $babacc$ (along with the rules used) is shown below.

$$\begin{array}{lll} S & \xrightarrow{1} ABCS & [S \to ABCS] \\ & \xrightarrow{1} ABCABCS & [S \to ABCS] \\ & \xrightarrow{1} ABCABC & [S \to \epsilon] \\ & \xrightarrow{1} BACABC & [AB \to BA] \\ & \xrightarrow{1} BAACBC & [CA \to AC] \\ & \xrightarrow{1} BAABCC & [CB \to BC] \\ & \xrightarrow{1} BABACC & [AB \to BA] \\ & \xrightarrow{1} bABACC & [B \to b] \\ & \xrightarrow{1} baBACC & [A \to a] \\ & \xrightarrow{1} babACC & [B \to b] \\ & \xrightarrow{1} babaCC & [A \to a] \\ & \xrightarrow{1} babacC & [C \to c] \\ & \xrightarrow{1} babacc & [C \to c] \end{array}$$

(b) What language over $\{a, b, c\}$ is generated by this grammar? Justify. **(5)**

Solution The grammar generates all strings over $\{a, b, c\}$ with equally many a's, b's and c's.

S generates an equal number of A's, B's and C's before becoming ϵ. Interchanging two of the adjacent symbols A, B, C (like in the rule $AB \to BA$) does not change the counts of A's, B's and C's. Finally, each A has to be converted to an a, each B to a b, and each C to a c. Therefore, each string in $\{a, b, c\}^*$ generated by this grammar has equally many a's, b's and c's.

Conversely, given any string $\alpha \in \{a, b, c\}^*$ with i occurrences of each of a, b and c, one can generate $(ABC)^i$ from S. Subsequently, the upper-case version of α can be obtained from $(ABC)^i$ by interchanging adjacent pairs of symbols. Finally, the rules $A \to a$, $B \to b$ and $C \to c$ convert the upper-case version of α to α.

4. Exactly one of the following languages is recursive, exactly one is not recursive but r.e., and exactly one is not r.e. Identify which one is what, and supply a corroborating proof in each case.

$$
\begin{aligned}
L_1 &= \{M_1\#M_2 \mid \text{NSTEPS}(M_1,\epsilon) < \text{NSTEPS}(M_2,\epsilon)\},\\
L_2 &= \{M_1\#M_2 \mid \text{NSTEPS}(M_1,\epsilon) \leqslant \text{NSTEPS}(M_2,\epsilon)\},\\
L_3 &= \{M_1\#M_2 \mid \mathcal{L}(M_1) \cap \mathcal{L}(M_2) \text{ is r.e.}\}.
\end{aligned}
$$

Here, M_1 and M_2 are (encoding of) Turing machines. For a Turing machine M and for an input w of M, the symbol NSTEPS(M, w) stands for the number of steps that M takes before halting, upon input w. If M loops (that is, does not halt) on w, we take NSTEPS$(M, w) = \infty$. If M' also loops on w', we make the convention that NSTEPS$(M, w) =$ NSTEPS(M', w') (that is, two infinities in this context are equal).

(a) L_1 is r.e. but not recursive .

Proof **(5)**

Solution A two-tape TM N can simulate M_1 on ϵ on one tape, and M_2 on ϵ on the other tape in a round-robin fashion. If the simulation of M_1 halts before that of M_2, N accepts. If the simulation of M_2 halts before that of M_1, N rejects. If the two simulations halt after the same number of steps, N rejects. If neither simulation halts, N continues with the simulation (that is, loops) for ever, and never accepts. This shows that L_1 is r.e.

In order to show that L_1 is not recursive, we propose a reduction HP $\leqslant_m L_1$. Upon input $M\#x$ (an instance of HP), the reduction algorithm outputs $M_1\#M_2$ (an instance of L_1) such that M halts on x if and only if M_1 halts in fewer steps than M_2 upon input ϵ.

M_1 upon input y_1 first checks whether $y_1 = \epsilon$. If not, M_1 enters an infinite loop. If $y_1 = \epsilon$, M_1 simulates M on x, and accepts if M halts.

Upon any input y_2, the other machine M_2 enters an infinite loop.

If M halts on x, M_1 halts on ϵ (that is, takes a finite number of steps before halting), whereas M_2 loops on ϵ (infinite steps), that is, NSTEPS$(M_1, \epsilon) <$ NSTEPS(M_2, ϵ). On the other hand, if M does not halt on x, both M_1 and M_2 loop on input ϵ, that is, NSTEPS$(M_1, \epsilon) =$ NSTEPS$(M_2, \epsilon) = \infty$.

(b) L_2 is ________ not r.e. ________ .

Proof **(5)**

Solution The language $\sim L_1 = \{M_1\#M_2 \mid \text{NSTEPS}(M_1, \epsilon) \geqslant \text{NSTEPS}(M_2, \epsilon)\}$ is not r.e., since if both L_1 and $\sim L_1$ are r.e., L_1 is recursive. The simple reduction $\sim L_1 \leqslant_m L_2$ converting $M_1\#M_2$ to $M_2\#M_1$ shows that L_2 is not r.e.

Alternatively, one can propose a direct reduction $\sim$HP $\leqslant_m L_2$ (or equivalently, HP $\leqslant_m \sim L_2$) to prove that L_2 is not r.e. Since $\sim L_2 = \{M_1\#M_2 \mid \text{NSTEPS}(M_1, \epsilon) > \text{NSTEPS}(M_2, \epsilon)\}$, the same reduction as in Part (a) (with the roles of M_1 and M_2 interchanged) works.

(c) L_3 is recursive .

Proof **(5)**

Solution Let $K_1 = \mathcal{L}(M_1)$ and $K_2 = \mathcal{L}(M_2)$ be r.e. languages. One can simulate M_1 and M_2 sequentially (or in parallel on two separate tapes/tracks) on the (same) input y, and accepts y if both the simulations accept y. This shows that r.e. languages are closed under intersection. Therefore, given a valid encoding $M_1\#M_2$ of two TM's M_1 and M_2, the decision whether $\mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ is r.e. is trivial (always "yes").

5. Which of the following are properties of r.e. sets? First, supply an answer Yes/No. If the answer is *Yes*, mention whether the property is trivial and/or monotone. If the answer is *No*, supply a one-line justification. In each case, the property is specified by a Turing machine M. **(5)**

(a) M accepts ϵ.

Yes, non-trivial, monotone

(b) M (explicitly) rejects ϵ.

No. We can construct examples of $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ with M_1 rejecting ϵ, and M_2 looping on ϵ.

(c) M halts on ϵ.

No. We can construct examples of $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ with M_1 halting on ϵ (rejection), and M_2 not.

(d) $\mathcal{L}(M)$ is a context-free language.

Yes, non-trivial, non-monotone

(e) $\mathcal{L}(M)$ contains a context-free language.

Yes, trivial, monotone