CS21004 Formal Languages and Automata Theory, Spring 2012–13

End-Semester Test

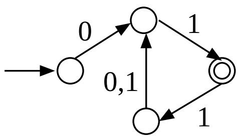
Maximum marks: 60 | Date: 19-Apr-2013 | Duration: Three hours (AN)

Roll no: ____________ **Name:** ______________________________

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

1. Answer the following six parts with brief justifications. Take $\Sigma = \{0, 1\}$ in all the parts. **(2 × 6)**

(a) Write a regular expression equivalent to the following NFA.



Solution $01(1(0+1)1)^*$ (easy inspection).

(b) Regular or not: $\{\alpha\beta\alpha\gamma \mid \alpha, \beta, \gamma \in \Sigma^*,\ |\beta| = |\gamma|\}$?

Solution Yes. We are allowed to take $\alpha = \epsilon$, so the language is the set of all even-length strings.

(c) Context-free or not: $\{\alpha\beta\alpha\gamma \mid \alpha, \beta, \gamma \in \Sigma^*,\ \alpha \neq \epsilon,\ |\beta| = |\gamma|\}$?

Solution Yes. The language consists of all strings of length $2l$ (for any $l \geqslant 1$) such that the first and the $(l+1)$-st symbols are the same. A PDA can match these two symbols (by guessing their positions and using its stack to verify $|\beta| = |\gamma|$).

(d) True or False: $\{\alpha\beta\gamma \mid \#0(\alpha) = \#1(\beta) = \#0(\gamma)\} \bigcap \mathcal{L}(0^*1^*0^*) = \{0^n1^n0^n \mid n \geqslant 0\}$?

Solution False. For example, the string $0^{n+2}1^n0^{n+1}$ is of the form $0^*1^*0^*$, and can be decomposed as $\alpha\beta\gamma$ with $\alpha = 0^n$, $\beta = 001^n0$ and $\gamma = 0^n$. Indeed, any string of the form $0^*1^*0^*$ admits a decomposition $\alpha\beta\gamma$ with $\#0(\alpha) = \#1(\beta) = \#0(\gamma)$.

(e) Recursive or not: $\{M \mid M$ is a Turing machine that accepts at least 2013 strings$\}$?

Solution No. Containing at least 2013 strings is a non-trivial (but monotone) property of r.e. sets. Now, use Rice's theorem, Part I.

(f) Recursively enumerable or not: $\{M \mid M$ is a Turing machine that accepts exactly 2013 strings$\}$?

Solution No. Containing exactly 2013 strings is a non-monotone property of r.e. sets. Now, use Rice's theorem, Part II.

2. Consider the context-free grammar G over $\{a, b\}$, with start symbol S, and with the following productions.

$$\begin{aligned} S &\rightarrow aaB \mid Abb \\ A &\rightarrow a \mid aA \\ B &\rightarrow b \mid bB \end{aligned}$$

(a) What is $\mathcal{L}(G)$? **(2)**

Solution $\mathcal{L}(G) = \{a^2b^n \mid n \geqslant 1\} \bigcup \{a^nb^2 \mid n \geqslant 1\}$.

(b) Prove that this CFG is ambiguous. **(4)**

Solution The string $aabb$ has two distinct leftmost derivations:

$$\begin{aligned} &S \Rightarrow aaB \Rightarrow aabB \Rightarrow aabb, \\ &S \Rightarrow Abb \Rightarrow aAbb \Rightarrow aabb. \end{aligned}$$

(c) Design an unambiguous context-free grammar for $\mathcal{L}(G)$. **(4)**

Solution In order to disambiguate this grammar, we separate out some small examples.

$$\begin{aligned} S &\rightarrow aab \mid abb \mid aabb \mid aaAbb \mid aaBbb \\ A &\rightarrow a \mid aA \\ B &\rightarrow b \mid bB \end{aligned}$$

Roll no: ____________________ **Name:** ____

3. Consider the unrestricted grammar over the singleton alphabet $\Sigma = \{a\}$, having the start symbol S, and with the following productions.

$$\begin{aligned} S &\rightarrow AS \mid aT \\ Aa &\rightarrow aaaA \\ AT &\rightarrow T \\ T &\rightarrow \epsilon \end{aligned}$$

(a) Show a derivation of a^9 using this grammar. **(4)**

Solution $S \Rightarrow AS \Rightarrow AAS \Rightarrow AAaT \Rightarrow AaaaAT \Rightarrow AaaaT \Rightarrow aaaAaaT \Rightarrow aaaaaaAaT \Rightarrow aaaaaaaaaAT \Rightarrow aaaaaaaaaT \Rightarrow aaaaaaaaa.$

(b) What is the language generated by this unrestricted grammar? Justify. **(4)**

Solution We have $\mathcal{L}(S) = \{a^{3^n} \mid n \geqslant 0\}$. In order to prove this, we may proceed by induction on the number of A's generated before the rule $S \rightarrow aT$ is applied. Each generated A must get in contact with T before vanishing. In the rightward journey of each A, the number of a's is tripled.

4. For two languages A and B over the same alphabet Σ, define the language

$$A/B = \{\alpha \in \Sigma^* \mid \alpha\beta \in A \text{ for some } \beta \in B\}.$$

Prove that if A and B are recursively enumerable, then so also is A/B. **(6)**

Solution Let A and B be recognized by Turing machines M and N. We design a two-track non-deterministic Turing machine K to accept A/B as follows.

K, upon input α, does the following:

(1) Non-deterministically choose a string β. Append β to α in the first track, and copy β to the initially empty second track.

(2) Simulate M on $\alpha\beta$ and N on β in parallel on the two tracks.

(3) If both the simulations accept, accept. If one or both of the simulations reject, reject. If both the simulations keep on looping, the parallel simulation in Step (2) never terminates (that is, Step (3) is never executed).

The non-deterministic choice of β can be replaced by all choices of β. In that case, the simulations on all $\alpha\beta$ and β run on a time-sharing basis.

Roll no: ________________ **Name:** ____

5. Identify the types of the following two languages (recursive / non-recursive but recursively enumerable / not recursively enumerable). Halting may be a choice of Turing machines, so do not use Rice's theorem. Construct appropriate Turing machines and/or supply appropriate reductions.

(a) $L_a = \{M \mid M$ is a Turing machine that halts on exactly 2013 input strings$\}$. **(6)**

L_a is ______________ not recursively enumerable ______________ .

Proof:

Solution We propose a reduction $\sim$ HP $\leqslant_m L_a$ which maps $M\#\alpha$ to N such that M does not halt on α if and only if N halts on exactly 2013 input strings. The reduction algorithms uses any 2013 constant strings $\gamma_1, \gamma_2, \ldots, \gamma_{2013}$ (distinct from one another). For example, we may have $\gamma_i = 0^i$ for $i = 1, 2, \ldots, 2013$.

N, upon input β, does the following:

(1) Check whether $\beta = \gamma_i$ for some $i = 1, 2, 3, \ldots, 2013$. If so, halt (after accepting or rejecting).

(2) Simulate M on α.

(3) If the simulation of Step (2) halts, halt (after accepting or rejecting).

It follows that if M halts on α, then N halts on every input β. On the other hand, if M does not halt on α, then N halts only on the 2013 input strings $\gamma_1, \gamma_2, \ldots, \gamma_{2013}$.

(b) $L_b = \{M \mid M$ is a Turing machine that halts on at least 2013 input strings$\}$. **(6)**

L_b is ______ recursively enumerable but not recursive ______ .

Proof:

Solution We can design a Turing machine K that simulates M on all possible input strings on a time-sharing basis. If any 2013 of the simulations halt, K accepts and halts. If 2013 strings are never found, the parallel simulation of K never stops. Thus, L_b is recursively enumerable. (Alternatively, K can be a non-deterministic Turing machine which simulates M on 2013 distinct choices of inputs. The simulations may proceed in parallel or one after another.)

In order to prove that L_b is not recursive, we make a reduction HP $\leqslant_m L_b$ that maps $M\#\alpha$ to N such that M halts on α if and only if N halts on at least 2013 input strings.

N, upon input β, does the following:

(1) Simulate M on α.

(2) If the simulation of Step (1) halts, halt (after accepting and rejecting).

It follows that if M halts on α, then N halts on all input strings (in particular, on at least 2013 input strings). On the other hand, if M does not halt on α, then N does not halt on any input string β.

Roll no: ________________ **Name:** ______________________________________

6. Find out (with appropriate justifications) which of the following two languages is/are recursive. Assume that some fixed input alphabet Σ is provided for CFG's and PDA's. Do not use any theorem or algorithm not covered in the lectures or tutorials.

(a) $L_c = \{G\#\alpha \mid G$ is a CFG, $\alpha \in \Sigma^*$, and $\alpha \in \mathcal{L}(G)\}$. **(6)**

L_c is ______________________ recursive ______________________ .

Proof:

Solution Here is a decider for L_c. Let S be the start symbol of G.

(1) If $\alpha = \epsilon$, find out whether $S \Rightarrow^* \epsilon$ is a valid derivation. More precisely, first mark all non-terminal symbols A for which $A \to \epsilon$ is a rule in G. Next, keep on marking non-terminal symbols B having rules $B \to C_1C_2 \ldots C_k$ with $C_1, C_2, \ldots, C_k$ already marked. Stop when no further marking of non-terminal symbols is possible. If the start symbol S is marked, accept, else reject.

(2) Convert G to the Chomsky normal form. Call this converted grammar G'.

(3) Derive all sentential forms of length $\leqslant |\alpha|$ using at most $2|\alpha| - 1$ derivation steps under the converted grammar G'. If α is ever generated, accept, else reject.

Step (1) can be completed in a finite number of steps, because there are only finitely many non-terminal symbols. Conversion of G to G' can again be done in a finite amount of time. The third step too finishes in a finite amount of time, since a grammar in the Chomsky normal form guarantees increase in length of a sentential form for each application of a production of the form $A \to BC$. Application of a production of the form $A \to a$, on the other hand, replaces a non-terminal symbol by a terminal symbol. Therefore, a sentence of length n does not require more than $2n - 1$ steps in the derivation.

(b) $L_d = \{G\#P \mid G$ is a CFG, P is a PDA, and $\mathcal{L}(G) = \mathcal{L}(P)\}$. **(6)**

L_d is ______not recursive______ .

Proof:

Solution We propose a reduction $\mathrm{ALL}_{\mathrm{CFG}} \leqslant_m L_d$. Given an input G for $\mathrm{ALL}_{\mathrm{CFG}}$, the reduction algorithm generates a PDA P with $\mathcal{L}(P) = \Sigma^*$, and outputs $G\#P$. Clearly, $G \in \mathrm{ALL}_{\mathrm{CFG}}$ if and only if $\mathcal{L}(G) = \mathcal{L}(P) = \Sigma^*$.

Roll no: ____________ **Name:** ______________________________

[Space for leftover answers and rough work]

[Space for leftover answers and rough work]