MTH 222 Theory of Computation

Solutions of the exercises that appeared in the End Semester Examination

Total marks: 50 December 4, 2002 Time: $3 + \epsilon$ hours

---

1. **(a)** $L_1 := \{\alpha \in \{a,b,c\}^* \mid \nu_a(\alpha) = \nu_b(\alpha) = \nu_c(\alpha)\}$, is recursive-but-non-context-free, where $\nu_x(\alpha)$ denotes the number of occurrences of $x \in \{a,b,c\}$ in $\alpha \in \{a,b,c\}^*$. If L_1 is context-free, consider the pumping lemma constant n for L_1 (Better use the stronger version of the lemma.) and using the string $a^n b^n c^n \in L_1$ arrive at a contradiction. So L_1 is not context-free. It is R.E., since we can design a TM M_1 to accept L_1. One may go for a four-tape machine, where the numbers of occurrences of a, b and c are counted and stored in tapes 2 through 4 as the strings $0^{\nu_a(\alpha)}$, $0^{\nu_b(\alpha)}$ and $0^{\nu_c(\alpha)}$. Subsequently the lengths of these strings can be easily compared. Clearly M_1 can be so constructed as to halt on every input. Thus L_1 is recursive as well.

(b) $L_2 := \{a,b,c\}^* \setminus L_1$ is context-free-but-not-regular. If L_2 is regular, $L_1 = \bar{L}_2$ will also be regular, but by Part (a) L_1 is not even context-free. One can write $L_2 = L_{21} \cup L_{22}$, where L_{21} (resp. L_{22}) consists of all strings $\alpha \in \{a,b,c\}^*$ with $\nu_a(\alpha) \neq \nu_b(\alpha)$ (resp. $\nu_b(\alpha) \neq \nu_c(\alpha)$). Since context-free languages are closed under union, it is sufficient to show that L_{21} and L_{22} are context-free. To this end I will construct a CFG $G = (\{a,b,c\}, \{S,A,B,C,E\}, S, R)$ for L_{21} with the following rules:

$$\begin{array}{lcl}
S & \to & A \mid B \\
E & \to & C \mid CaEbC \mid CbEaC \mid EE \\
A & \to & CaE \mid CaA \mid CbAA \\
B & \to & CbE \mid CbB \mid CaBB \\
C & \to & \epsilon \mid cC
\end{array}$$

An analogous CFG for L_{22} can be written.

(c) It is an easy matter to check that $\mathcal{L}(T) = b^*$ and it follows that $L_3 = \mathcal{L}(G) = \mathcal{L}(S) = (abb^*)^*$. Thus L_3 is regular-and-infinite.

(d) L_4 is non-R.E. For a proof of this let us look at its complement $L_5 := \bar{L}_4$. Since there are R.E. languages (like Σ^*) containing ϵ and those (like $\emptyset$) not containing ϵ, it follows from Rice's theorem that L_5 is not recursive. Converting $\langle M_i \rangle$ to $\langle M_i, \epsilon \rangle = \langle M_i \rangle 111$ can be easily done by a TM (that halts eventually on every input). Now we can feed $\langle M_i, \epsilon \rangle$ to the universal TM U. If $\epsilon \in \mathcal{L}(M_i)$, U accepts $\langle M_i, \epsilon \rangle$ and halts. If $\epsilon \notin \mathcal{L}(M_i)$, then U either halts without accepting or does not halt at all on the input $\langle M_i, \epsilon \rangle$. To sum up the converter (of $\langle M_i \rangle$ to $\langle M_i, \epsilon \rangle$) in conjunction with U gives us a TM for accepting L_5. Thus L_5 is R.E.-but-not-recursive. If $L_4 = \bar{L}_5$ were R.E., then L_4 would be recursive, a contradiction.

---

2. Let $M = (Q_M, \Sigma, \Gamma, \delta_M, s_M, F_M, \sqcup)$ be a TM with the property that $\delta_M(q, X)$, if defined, is of the form (p, Y, R). Let $L := \mathcal{L}(M)$. I will design an ϵ-NFA $N = (Q_N, \Sigma \cup \{\$\}, \delta_N, s_N, F_N)$ to accept the language $L\$$ for some symbol $\$ \notin \Sigma$. But then the quotient language $L\$/\$ = L$ will also be regular.

Take $Q_N := Q_M \uplus Q'_M \uplus \{f'_N, f_N\}$. Here $|Q'_M| = |Q_M|$. If $Q_M = \{q_1, \ldots, q_m\}$, let us denote $Q'_M = \{q'_1, \ldots, q'_m\}$, i.e., there is a one-to-one correspondence of $q_i \in Q_M$ with $q'_i \in Q'_M$. In fact, both q_i and q'_i represent the same state of M, q_i stands for the case that the head of M is scanning a non-blank symbol and q'_i for the case that the head of M is scanning a blank. Also take $s_N := s_M$ and $F_N := \{f_N\}$. The transitions in δ_N are as follows:

- For every non-accepting state $q \in Q_M$ add a transition $q \to q'$ labeled ϵ. When M moves to the blank portion of the tape, N would make this move.
- For every accepting state $f \in F_M$ add a transition $f \to f'_N$ labeled ϵ. Since there are no transitions of M from the state f (a valid assumption), N can switch only to f'_N, once it reaches any accepting state of M. But at that time M need not have scanned the entire input, whereas N must see its entire input before

accepting. Thus we add a self-loop from f'_N to itself labeled by each $a \in \Sigma$. Finally N reads \$ and accepts, i.e., there is a transition $f'_N \to f_N$ labeled \$.

- M may choose to accept, when it is in the blank portion of the tape following the input. Thus for each accepting state $f \in F_M$ we add the transition $f' \to f_N$ labeled by \$. Note that once N reaches the states in Q'_M, M never reads any non-blank symbol in its tape and hence the transition $f' \to f'_N$ would be incorrect (since there is a self-loop $f'_N \to f'_N$ labeled by the input alphabet). This is the reason why I separated f_N and f'_N.
- For each transition $\delta_M(q, a) = (p, Y, R)$ of M with $a \in \Sigma$ add the transition $q \to p$ labeled a. Using these transitions N simulates the moves of M, while M is scanning the non-blank portion of the tape.
- For each transition $\delta_M(q, \sqcup) = (p, Y, R)$ of M add the transition $q' \to p'$ labeled ϵ. This simulates M's behavior, when M is scanning a blank.

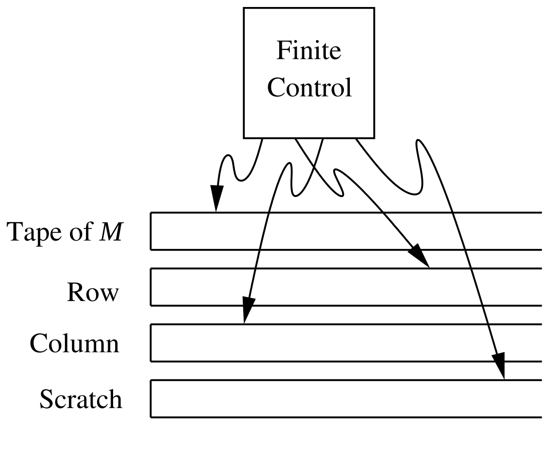
That's it! Though almost clear from the description above, let us prove formally that $\mathcal{L}(N) = L\$$. First consider some $\alpha = a_1 \ldots a_l \in L$. Since M accepts α, it goes to one of its final states while processing α. If this happens before M starts scanning a blank α, the sequence of ID's of M will look like $(p_1, \underline{a_1}a_2 \ldots a_l) \vdash_M (p_2, X_1\underline{a_2}a_3 \ldots a_l) \vdash_M \cdots \vdash_M (f, X_1X_2 \ldots X_k\underline{a_{k+1}} \ldots a_l)$, where $p_i \in Q_M \setminus F_M$, $f \in F_M$ and $k \leqslant l$. (Note that $k = l$ means that the last ID should be rewritten as $(f, X_1X_2 \ldots X_l\underline{\sqcup})$.) Then N accepts $\alpha\$$ using the moves $p_1 \overset{a_1}{\to} p_2 \overset{a_2}{\to} p_3 \overset{a_3}{\to} \cdots \overset{a_k}{\to} f \overset{\epsilon}{\to} f'_N \overset{a_{k+1}}{\to} f'_N \overset{a_{k+2}}{\to} \cdots \overset{a_l}{\to} f'_N \overset{\$}{\to} f_N$. Now assume that M accepts α after entering the blank region of the tape following the input. Then the sequence of ID's for N would look like $(p_1, \underline{a_1}a_2 \ldots a_l) \vdash_M (p_2, X_1\underline{a_2}a_3 \ldots a_l) \vdash_M \cdots \vdash_M (p_{l+1}, X_1X_2 \ldots X_l\underline{\sqcup}) \vdash_M (p_{l+1}, X_1X_2 \ldots X_lX_{l+1}\underline{\sqcup}) \vdash_M \cdots \vdash_M (f, X_1X_2 \ldots X_lX_{l+1} \ldots X_{l+k}\underline{\sqcup})$ for some $k \geqslant 2$, $p_i \in Q_M \setminus F_M$ and $f \in F_M$. In this case N accepts $\alpha\$$ following the sequence of moves $p_1 \overset{a_1}{\to} p_2 \overset{a_2}{\to} \cdots \overset{a_l}{\to} p_{l+1} \overset{\epsilon}{\to} p'_{l+1} \overset{\epsilon}{\to} p'_{l+2} \overset{\epsilon}{\to} \cdots \overset{\epsilon}{\to} f' \overset{\$}{\to} f_N$.

Now consider that M rejects α. First suppose that M does not halt on input α, i.e., M never reaches any state $f \in F_M$. Therefore N also can not reach any f or f' and nor to f_N as well. Then suppose that M halts in a non-accepting state $q \in Q_M \setminus F_M$. N, while in state $p \in Q_M \setminus F_M$, can always choose the ϵ-transition $p \to p'$. But there are no transitions of N from any $p' \in Q'_M$ to any $r \in Q_M$. Also N never consumes a real input during the transitions between the states of Q'_M, whereas it consumes only the end-of-string marker \$ for switching from some states of Q'_M to f_N. Thus N is bound to go to a stuck position (or end up in an infinite loop inside Q'_M), if it chooses the move $p \to p'$ before all symbols of α are read. Therefore if M halts before scanning all the input symbols, N also can not accept $\alpha\$$. If M moves past the input, N must also use some ϵ-transition $p \to p'$, since no ϵ-transitions of N are defined among the states of Q_M. Therefore if M halts in state q while scanning the blank, N also halts in q', because there are no ϵ-moves of N from q'. In all the cases N rejects $\alpha\$$.

Finally note that every string accepted by N is of the form $\alpha\$$ for some $\alpha \in \Sigma^*$. It follows that $\mathcal{L}(N) = L\$$.

---

3. I will design a TM N with four semi-infinite one-dimensional tapes such that $\mathcal{L}(N) = \mathcal{L}(M)$. Since $\mathcal{L}(N)$ is R.E., the result follows. The working of N is described schematically as in the adjacent diagram.



The two-dimensional tape of M is simulated in the first tape of N. Let us plan to use the bijection $\varphi : \mathbb{N} \to \mathbb{N}$, $\varphi(i, j) = 1 + \frac{1}{2}[(i + j)^2 - i - 3j]$, for this purpose. If we number the cells of the tape of M by the pair $(i, j) \in \mathbb{N}^2$, where i denotes the row number and j the column number, then the (i, j)-th cell of M is stored in the $\varphi(i, j)$-th cell of the first tape of N.

The second and third tapes of N hold the coordinates (i.e., the position (i, j)) of the head of M. A simple way to do this is to store the strings 0^i and 0^j.

A scratch tape is used for intermediate calculations. If necessary one may go for more than one (but only finitely many) scratch tapes. Let me assume that only one scratch tape is sufficient (in fact, it is!) and makes it easy enough to handle all the calculations that I will describe below.

The input string $\alpha := a_1 \ldots a_n$ is provided to N on Tape 1 starting from the leftmost cell of that tape. In order to simulate the initial configuration of M, N first relocates a_i to the $\varphi(1, i)$-th cell for all $i = 1, \ldots, n$. The scratch tape (or tapes) can be used for these relocations. All other cells of the first tape of N are made empty. Then M's first head is positioned at the first cell of the first tape. N's finite control mimics the states of M, i.e., N initializes its finite control to the start state of M.

In order to simulate a move of M, N scans its first tape and also consults its current state (which is the same as M's current state). If M does not specify a transition in this situation, N halts without accepting. Otherwise N simulates M's (unique) move as follows:

- N changes its state as M would do.
- N replaces the symbol of the cell scanned by its first head as M would do.
- Depending on the move of M from the (i, j)-th cell to the (i', j')-th cell, N adjusts its first head and the contents of its second and third tapes as follows. The new coordinates (i', j') of M's head is related to the old coordinates (i, j) as follows:

Move of M	(i', j')	$\varphi(i', j') - \varphi(i, j)$
North	$(i-1, j)$	$-(i + j - 1)$
East	$(i, j+1)$	$+(i + j - 1)$
West	$(i, j-1)$	$-(i + j - 2)$
South	$(i+1, j)$	$+(i + j)$

  The first head of N should be moved right by $\varphi(i', j') - \varphi(i, j)$ cells. (A negative value of $\varphi(i', j') - \varphi(i, j)$ indicates a left movement.) N reads i and j from its second and third tapes and computes in its scratch tape the quantity $\varphi(i', j') - \varphi(i, j)$ as shown in the third column of the above table. It then moves its first head by an appropriate amount and in the appropriate direction. It then replaces the contents of its second and third tapes from 0^i and 0^j to $0^{i'}$ and $0^{j'}$. For all the moves of M's head one of i and j remains unaltered, the other either increases or decreases by 1. Thus it is only necessary to add or delete one 0 from one of the second and third tapes.
- Finally N checks if the updated state is an accepting state for M. If so, N accepts and halts. If not, N simulates the next move of M.

It is clear from the above description that N accepts a string α, if and only if M accepts α and therefore $\mathcal{L}(N) = \mathcal{L}(M)$. One can convert N to a standard one-tape one-head TM with a one-dimensional tape (infinite in both directions) using the procedures described in the lectures. Thus $\mathcal{L}(N) = \mathcal{L}(M)$ is R.E.

---

4. (a) First let me provide a characterization of the members of L_{gt}. $\alpha\#\beta$ with $\alpha, \beta \in \{0,1\}^+$ is in L_{gt}, if and only if $\alpha = 0^i\gamma 1\mu$ and $0^k\beta = 0^j\gamma 0\nu$ for some $i, j, k \in \mathbb{Z}_+$ (the leading zeros), $\gamma, \mu, \nu \in \{0,1\}^*$ with $|\mu| = |\nu|$. The grammar for L_{gt} can be designed in a manner similar to the $\alpha\alpha$ language described in the lectures. The rules are given below. A brief explanation of the intended purpose of each rule is also provided.

$S \rightarrow CUV$	C generates 0^i of α, U is responsible for the two occurrences of γ and V for 1μ and 0ν.
$U \rightarrow D \mid UA0 \mid UB1$	D generates $\#0^j$ and also acts as the 'center'. U generates the strings $D((A0) \cup (B1))^*$. Later each A will be converted to a 0 and each B to 1, once it meets D and goes to the left of D.
$V \rightarrow B0 \mid VA0 \mid VA1 \mid VB0 \mid VB1$	V generates the strings $B0((A \cup B)(0 \cup 1))^*$. The part $B0$ makes the final terminal string of the proper form, whereas $((A \cup B)(0 \cup 1))^*$ will lead to μ and ν after A's and B's travel left, meet the center D, go to the left of D and get converted to 0's and 1's respectively.

$0A \rightarrow A0$	A can penetrate through 0's.
$1A \rightarrow A1$	A can penetrate through 1's.
$DA \rightarrow 0D$	A meets D from right, goes to the left of D after getting transformed to 0.
$0B \rightarrow B0$	B can penetrate through 0's.
$1B \rightarrow B1$	B can penetrate through 1's.
$DB \rightarrow 1D$	B meets D from right, goes to the left of D after getting transformed to 1.
$C \rightarrow C0 \mid \epsilon$	C generates the leading zeros for α and then vanishes.
$D \rightarrow D0$	D generates leading zeros (0^j) for β.
$D00 \rightarrow D0$	D can also erase leading zeros (0^k) from β, but this should not result in an
$D01 \rightarrow D1$	empty β. So delete a leading zero, if something appears after it.
$D \rightarrow \#$	Finally D gets transformed to the separator $\#$.

Since $\mathcal{L}(S)$ contains strings consisting only of terminals, each A and B generated by U and V must vanish. The only way this can be done is by an interaction with D, which puts 0's and 1's of α in the intended positions. Also D can vanish only after converting all A's and B's. C does not play any rôle in the alignment of A's and B's and hence it can vanish at any point of time (after generating the i 0's of α). It follows that $\mathcal{L}(S) = L_{gt}$.

(b) The TM M_{gt} I am going to design for accepting L_{gt} is a two-tape TM. The second tape is used as a scratch tape in order to speed up certain computations. To start with M_{gt} checks if the input is in the correct format, i.e., in the format $\alpha\#\beta$ with $\alpha, \beta \in \{0,1\}^+$. This check can be done by a single pass through the input string. If the input is not of the correct format, M_{gt} halts without accepting.

M_{gt} then copies β to the second tape and removes the suffix $\#\beta$ from the first tape. Thus the two arguments for the comparison are now separated and placed in different tapes. A single-tape TM (with both the operands α and β on the same tape) can very well do whatever M_{gt} is going to do next, but introduction of the second tape makes the computation more efficient and the description easier.

M_{gt} now deletes all leading zeros from α. The head for the first tape is positioned at the beginning of α and as long as the head scans a zero it keeps on replacing it by the blank and moving right. If the head meets a 1, then it stops removing leading zeros. If it meets the blank, then α is of the from 0^i. M_{gt} then writes a single zero to represent $\text{val}(\alpha) = 0$.

In an analogous way M_{gt} then deletes leading zeros from β. Let us denote the strings left in the tapes now by α' and β'. We clearly have $\text{val}(\alpha) = \text{val}(\alpha')$ and $\text{val}(\beta) = \text{val}(\beta')$.

M_{gt} then compares the lengths of α' and β'. If $|\alpha'| > |\beta'|$, then M_{gt} accepts and halts. If $|\alpha'| < |\beta'|$, then M_{gt} halts without accepting. This leaves us with the case $|\alpha'| = |\beta'|$. Let $\alpha' = a_1 \ldots a_m$ and $\beta' = b_1 \ldots b_m$ for some $n \in \mathbb{N}$. M_{gt} then compares a_1 with b_1, a_2 with b_2 and so on, till it detects $a_i \neq b_i$ for some i. If $a_i = 1$ and $b_i = 0$, then M_{gt} accepts and halts. If $a_i = 0$ and $b_i = 1$, then M_{gt} halts without accepting. If $a_j = b_j$ for all $j = 1, \ldots, m$, then the two operands are equal and M_{gt} halts without accepting.

(c) L_{gt} is recursive, since the machine M_{gt} of Part (2) halts on any input. More specifically if n is the length of the input ($\alpha\#\beta$), then the two-tape machine halts after $O(n)$ moves. Each logical step of the operation of M (checking the validity of the input format, copying β to the second tape, erasing $\#\beta$ from the first tape, deleting leading zeros from α and β, comparing the lengths of α' and β' and finally comparing the individual bits of α' and β') can be carried out using $O(n)$ moves. M_{gt} has to reposition its heads for only a constant number of times and each such repositioning is doable with $O(n)$ moves. One can convert this two-tape TM to a one-tape TM which requires $O(n^2)$ moves. In any case the acceptance or rejection decision of the machine comes in finite time for any input.