Formal Languages and Automata Theory

Third Long Test

Time: 50 minutes | 14–April–2021 | Maximum marks: 40

---

1. Consider the language

$$L = \left\{ wc^m d^n \mid w \in \{a,b\}^*,\ m = \text{the number of } a\text{'s in } w, \text{ and } n = \text{the number of } b\text{'s in } w \right\}.$$

(a) Design a Turing machine that accepts L. **(4)**

Solution [Sketch] First check whether the input is in the correct format. If not (for example, if an a or a b appears after a c or a d, or if a d appears after a c), reject and halt. Match the a's in w with the c's. If the matching fails, reject and halt. Then, match the b's in w with the d's. If the matching fails, reject and halt. Accept and halt if the reject decision has not yet been taken so far.

(b) Give an unrestricted grammar for L. **(6)**

Solution The following grammar with the start symbol S generates L.

$$\begin{array}{rcl}
S & \rightarrow & T\# \\
T & \rightarrow & aTC \mid bTD \mid \varepsilon \\
DC & \rightarrow & CD \\
D\# & \rightarrow & d \\
Dd & \rightarrow & dd \\
C\# & \rightarrow & c \\
Cd & \rightarrow & cd \\
Cc & \rightarrow & cc \\
\# & \rightarrow & \varepsilon
\end{array}$$

2. **(a)** Let $f : \{1,2,\ldots,n\} \times \{1,2,\ldots,n\} \rightarrow \{1,2,\ldots,n^2\}$ be the bijective function $f(i,j) = (i-1)n + j$ for $1 \leqslant i,j \leqslant n$. Consider the language L that consists of all strings $x\,\#\,0^k$ of the following form.

1. $x \in \{0,1\}^*$, and $|x| = n^2$ for some integer $n \geqslant k \geqslant 1$. Denote $x = x_1 x_2 \ldots x_{n^2}$.
2. For each i,j satisfying $1 \leqslant i < j \leqslant n$, we have $x_{f(i,j)} = x_{f(j,i)}$, and for each i in the range $1 \leqslant i \leqslant n$, we have $x_{f(i,i)} = 0$.
3. There exists a set $S \subseteq \{1,2,\ldots,n\}$ with $|S| \geqslant k$ such that for each $i,j \in S$, we have $x_{f(i,j)} = 0$.

Design a nondeterministic Turing machine (NTM) accepting L. (**Hint:** Think of undirected graphs.) **(6)**

Solution The strings in L are encodings of adjacency matrices of graphs with $n \geqslant 1$ vertices and an integer k, with the additional property that the graph has an independent set of size at least k.

On the work tape of the NTM, guess n, and check for Condition 1.

Then, check for Condition 2.

Condition 3 requires the graph to have an independent set S of size at least k. Guess a binary vector of length n (first write down n 0's, where the square of the length should match the length of x, then nondeterministically change some of the 0's to 1's) which is an indicator vector I for the vertices of S. Check if the indicator vector I has at least k 1's (the number of 1's should be at least the number of 0's after the # symbol in the input), otherwise reject. For each i,j in the range $1 \leq i,j \leq n$ such that both the i-th and the j-th bits in I are 1, check if $x_{f(i,j)} = 0$. If this is not true for some pair i,j, then reject. Otherwise, accept the input string.

(b) A Jump Turing machine (JTM) $J = (Q, \Sigma, \Gamma, \delta, \vdash, \Box, s, t, r)$ is like a standard one-tape Turing machine (TM) with the only exception that each transition of J is of the form $\delta(p,A) = (q,B,m)$, where $p,q \in Q$, and

$A, B \in \Gamma$, and $m \in \mathbb{Z}$. This means that if the finite control of J is in the state p and the head of J scans the tape symbol A, then the state changes to q, the content of the tape cell is changed from A to B, and the head jumps by m cells relative to the current position. If $m = 0$, the head stays at the current cell. If $m > 0$, the head makes a right jump. If $m < 0$, the head makes a left jump with the understanding that if the head is at position i on the tape and $|m| > i$, then the head goes to the leftmost cell (which stores the left end-marker $\vdash$). Also assume that if A is $\vdash$, then $m \geqslant 0$. Prove that a JTM is equivalent to a TM. **(4)**

Solution A TM is a special case of a JTM where $m = \pm 1$ in each transition.

Conversely, we show that a transition $\delta_J(p, A) = (q, B, m)$ of J can be simulated by multiple transitions of an ordinary TM M. Q_M consists of all the states in Q_J along with some additional temporary states. Let us see how different values of m can be handled.

$m > 0$: Introduce $m - 1$ temporary states $u_1, u_2, \ldots, u_{m-1}$ and the transitions $\delta_M(p, A) = (u_1, B, R)$, $\delta_M(u_1, *) = (u_2, *, R)$, $\delta_M(u_2, *) = (u_3, *, R)$, $\ldots$, $\delta_M(u_{m-1}, *) = (q, *, R)$. Here, $*$ is any tape symbol which is not changed during the last $m - 1$ transitions of M.

$m < 0$: Write $m = -n$ with $n > 0$. Introduce n temporary states $v_1, v_2, \ldots, v_n$ and the transitions $\delta_M(p, A) = (v_1, B, L)$, $\delta_M(v_1, *) = (v_2, *, L)$, $\delta_M(v_2, *) = (v_3, *, L)$, $\ldots$, $\delta_M(v_{n-1}, *) = (q, *, L)$. Here, $*$ is any tape symbol other than $\vdash$. In order to ensure that M never moves to the left of the left end-marker, add the transitions $\delta_M(v_i, \vdash) = \delta_M(v_n, \vdash, R)$ for all $i \in [1, n-1]$, and $\delta_M(v_n, *) = (q, *, L)$.

$m = 0$: Add a temporary state w and the transitions $\delta_M(p, A) = (w, B, R)$ and $\delta_M(w, *) = (q, *, L)$.

Notice that since each m in a transition is finite, and there are only finitely many entries in the transition table of J, only finitely many temporary states need to be added.

3. Let N be a nondeterministic Turing machine (NTM). We say that N faces a dilemma if at some point in its working, it encounters a situation where the finite control is in the state p, the head scans the tape symbol a, and $\delta(p, a)$ offers multiple (two or more) possibilities, where p is neither the accept nor the reject state. Consider the following two languages.

$$\mathrm{DILEMMA}_\varepsilon = \Big\{ N \mid N \text{ is an NTM which faces a dilemma at least once on input } \varepsilon \Big\},$$
$$\mathrm{DILEMMA}_{\mathrm{ALL}} = \Big\{ N \mid N \text{ is an NTM which faces a dilemma at least once on each input} \Big\}.$$

(a) Prove that $\mathrm{DILEMMA}_\varepsilon$ is recursively enumerable but not recursive. **(5)**

Solution [RE] Here is a Turing machine T that recognizes $\mathrm{DILEMMA}_\varepsilon$. T simulates N on ε. The simulation of an NTM in the current context goes as follows. Before simulating each move of N, T first finds out the number m of transition possibilities that apply to the current situation of N. If $m = 0$, T rejects and halts. If $m = 1$, T simulates the unique transition applicable. If the resulting state is t or r, T rejects and halts. If $m \geqslant 2$, T accepts and halts.

[Not recursive] We propose a reduction $\mathrm{HP} \leqslant \mathrm{DILEMMA}_\varepsilon$ that takes $M \# w$ to N such that N faces a dilemma on input ε if and only if M halts on w. Here, M is considered to be a DTM (this is how HP was defined).

N, on input v, does the following.

1. Simulate M on w.
2. If the simulation halts, make a nondeterministic choice to jump to the accept or to the reject state.
3. Accept/Reject depending on the choice, and halt.

Since M is a DTM, the simulation of Step 2 never faces a dilemma. Therefore if M does not halt on w, N never faces a dilemma. Conversely, if M halts on w, N faces a dilemma in Step 3 on all inputs, and in particular on ε.

(b) Prove that $\mathrm{DILEMMA}_{\mathrm{ALL}}$ is not recursively enumerable. **(5)**

Solution We propose a reduction $\overline{\mathrm{HP}} \leqslant \mathrm{DILEMMA}_{\mathrm{ALL}}$ taking $M \# w$ to N such that N faces a dilemma on all inputs if and only if M does not halt on w. Here again, we take M to be DTM.

N, on input v, does the following.

1. Simulate M on w for $|v|$ steps.

2. If the simulation of Step 1 does not halt, make a nondeterministic choice to jump to the accept or to the reject state. Accept/Reject depending on the choice, and halt.
3. If the simulation of Step 1 halts, reject and halt.

Since M is a DTM, the simulation of Step 1 never faces a dilemma. Any dilemma that N faces must be in Step 2. If M does not halt on w, then it does not halt in any finite number (like $|v|$) steps, so a nondeterministic choice is made by N in Step 2, that is, N faces a dilemma on any input v. On the other hand, if M halts on w in s steps, then Step 2 is executed if and only if $|v| < s$. If $|v| \geqslant s$, then Step 3 is executed, and N never faces a dilemma. Therefore in this case N faces a dilemma not on all inputs.

4. Let G be a context-free grammar over an input alphabet Σ, accepting the language $L = \mathscr{L}(G)$. Also, let F be a *finite non-empty* subset of Σ^*. Prove/Disprove whether each of the following two problems is decidable.

(a) Given G and F, determine whether $L = F$. **(5)**

Solution [*Decidable*]

First, note that the membership problem for a CFG is decidable. So for each $w \in F$, determine whether $w \in L$. If some $w \in F$ is not in L, reject.

Convert G to CNF, and derive a pumping-lemma constant k for L. If some string in F is of length $\geqslant k$, then L is infinite (you can pump in), so reject. Otherwise, check whether G can generate any string of length $< k$ other than those in F. If yes, reject. Finally, check whether G can generate any string of length in the range $[k, 2k)$. If yes, reject.

If the reject decision is not yet taken, accept.

(b) Given G and F, determine whether $L = \Sigma^* - F$. **(5)**

Solution [*Undecidable*]

[Proof based on reduction]

Assume that the given problem has a decider D. Using this, we prepare a decider D' for the problem whether $L' = \mathscr{L}(G') = \Sigma^*$, where G' is a CFG over Σ. D' runs the following steps.

1. Decide whether $\varepsilon \in L'$. If not, reject.
2. Invoke the decider D with input $G = \text{CNF}(G')$ and $F = \{\varepsilon\}$.
3. Accept if D accepts, or reject if D rejects.

Step 1 is decidable, because we have seen a marking algorithm for the membership of ε in the language of a CFG. We have also seen that the general membership problem whether a CFG G can generate a given string w is decidable.

Step 2 is executed if and only if $\varepsilon \in L'$. In that case, $L = \mathscr{L}(G) = \mathscr{L}(G') - \{\varepsilon\} = L' - F$. If $L' = \Sigma^*$, we have $L = \Sigma^* - F$. Conversely, if $L' \neq \Sigma^*$, there exists a non-empty $w \in \Sigma^*$ such that $w \notin L'$ (we have $\varepsilon \in L'$). But then $w \notin L' - F = L$, that is, $L \neq \Sigma^* - F$. Therefore the above three steps decide the full-ness of G', a contradiction to the fact the CFL full-ness is undecidable.

[Proof based on valid computation histories]

This is similar to the reduction from $\overline{\text{HP}}$ to the given language $\{G \# F \mid \mathscr{L}(G) = \Delta^* - F\}$, where G is a CFG over Δ. Given $M \# w$, a CFG G is to be prepared such that $L = \mathscr{L}(G) = \Delta^* - F$ if and only if M does not halt on w. Take $F = \{\varepsilon\}$,

$$\text{VALCOMP}^+(M,w) = \{\varepsilon\} \bigcup \text{VALCOMP}(M,w),$$

and

$$L = \overline{\text{VALCOMP}^+(M,w)} = \overline{\text{VALCOMP}(M,w)} \bigcap \Delta^+.$$

First, note that L is a CFL because it is the intersection of a CFL $\overline{\text{VALCOMP}(M,w)}$ and a regular set Δ^+. A total TM can design a DFA for Δ^+, and then a PDA for L using a product construction on this DFA and a PDA for $\overline{\text{VALCOMP}(M,w)}$. The TM then uses the PDA-to-CFG conversion procedure to generate a CFG G for L. This completes the reduction $M \# w \mapsto G \# F$.

If M does not halt on w, then $\text{VALCOMP}(M,w) = \emptyset$, so $L = \overline{\emptyset} \cap \Delta^+ = \Delta^* \cap \Delta^+ = \Delta^+ = \Delta^* - \{\varepsilon\} = \Delta^* - F$. Conversely, if M halts on w, there are infinitely many (non-empty) computation histories of M on w, so L is a proper subset of (and so not equal to) $\Delta^+ = \Delta^* - F$.
