Solution to Introduction to the Theory of Computation

Lwins_Lights

Contents

1 Regular Languages

2 Context-Free Languages

3 The Church–Turing Thesis

4 Decidability

4.10 *Refer to the textbook.*

4.11 By the pumping lemma, $L(M)$ is an infinite language $\iff$ $L(M)$ includes a string of length p. Since $R = \Sigma^p\Sigma^*$ is a regular language, it is easy to design a PDA P recognizing $L(M) \cap R$. Then use E_{PDA}'s decider to decide whether $L(M) \cap R = \varnothing$.

4.12 *Refer to the textbook.*

4.13 $L(R) \subseteq L(S) \iff L(R) \cup L(S) = L(S)$, which can be decided by EQ_{DFA}'s decider.

4.14 *Refer to the textbook.*

*4.15 By the pumping lemma, $\mathtt{1}^k \in L(G) \implies \mathtt{1}^{k+p!} \in L(G)$ for every $k \geq p$. Therefore $\mathtt{1}^* \subseteq L(G) \iff \{\mathtt{1}^k \mid k \leq p + p!\} \subseteq L(G)$, which can be easily checked by a TM in finite time.

4.16 Since $A = \{\langle R\rangle \mid R \text{ is a regular expression}, R \cap \Sigma^*\mathtt{111}\Sigma^* \neq \varnothing\}$, we only need an E_{DFA}'s decider.

4.17 Suppose we have two DFAs D_1 and D_2. Let D be a DFA recognizing $L(D_1) \oplus L(D_2)$, where $\oplus$ means symmetric difference. By the pumping lemma, $L(D) \cap (\Sigma \cup \epsilon)^p = \varnothing \implies L(D) = \varnothing$, thus p can be the required length.

*4.18 $\Leftarrow$: It is enough to design a TM, which recognizes C by checking whether $\langle x, y\rangle \in D$ for all possible y one by one.

$\Rightarrow$: Suppose TM M recognizes C. Let $D = \{\langle x, y\rangle \mid x \in C \text{ and } y \text{ is the computation history of } M \text{ on input } x\}$, which is obviously decidable.

*4.19 Let C be a recognizable but undecidable language, e.g., A_{TM}. Construct D provided by problem 4.18. Letting homomorphism f satisfy $f(\langle x, y\rangle) = x$ for every x, y we obtain $f(D) = C$.

4.20 Let M be a TM which runs both $\overline{A}$'s recognizer and $\overline{B}$'s recognizer on its own input. M accepts when $\overline{B}$'s recognizer accepts and rejects when $\overline{A}$'s recognizer accepts. Clearly $C = L(M)$ separates A and B.

4.21 M is a DFA that accepts $w^{\mathcal{R}}$ whenever it accepts $w \iff L(M) = L(M)^{\mathcal{R}}$, which can be decided by EQ_{DFA}'s decider.

4.22 In order to determine whether $L(R)$ is prefix-free, it suffices to check whether the DFA recognizing $L(R)$ has the property that from a reachable accept state we could arrive an accept state again by several transitions.

*4.23 *Refer to the textbook.*

4.24 A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ has an useless state $q \in Q$ if and only if PDA $P' = (Q, \Sigma, \Gamma, \delta, q_0, \{q\})$ recognizes $\varnothing$. Therefore we can use E_{PDA}'s decider to solve it.

*4.25 *Refer to the textbook.*

*4.26 M is a DFA that accepts some palindrome $\iff$ CFL $\{w \mid w = w^{\mathcal{R}}\} \cap L(M)$ is not empty, which can be decided by E_{PDA}'s decider.

*4.27 Refer to the solution to problem 4.26.

4.28 x is a substring of some $y \in L(G) \iff L(G) \cap \Sigma^* x \Sigma^* \neq \varnothing$, which can be decided by E_{PDA}'s decider.

4.29 *Need a solution.*

4.30 *Need a solution.*

4.31 *Need a solution.*

4.32 *Need a solution.*

5 Reducibility

5.9 Reduce to A_{TM}. To determine whether TM M accepts w, construct TM N which always accepts `01` but accepts `10` if and only if M accepts w.

5.10 *Refer to the textbook.*

5.11 *Refer to the textbook.*

5.12 Reduce to E_{TM}. To determine whether TM M accepts nothing, construct TM N which simulates M on N's own input w but never writes a blank symbol over a nonblank symbol unless when M accepts.

5.13 Reduce to E_{TM}. To determine whether TM M accepts nothing, construct TM N which simulates M on N's own input w and obviously has no useless state except the one q_{accept}.

5.14 *Need a solution.*

5.15 Let M' be M after modifying all its transitions $\delta(q_i, a) = (q_{\text{accept}}, b, X)$ to $\delta(q_i, a) = (q_{\text{reject}}, b, X)$, and then modifying all $\delta(q_i, a) = (q_j, b, \text{L})$ to $\delta(q_i, a) = (q_{\text{accept}}, b, \text{R})$. The problem is now reduced to checking whether M', as a *TM with stay put instead of left* described in problem 3.13, accepts w. It is easy since $L(M')$ is regular by the solution to problem 3.13.

5.16 Suppose for the sake of contradiction that BB is computable. Then obviously there exists a TM M having k states, which writes $BB(n) + 1$ `1`s on the tape, when given input $\langle n \rangle$. Further, using M we can construct a series of TMs M_n having exactly $k + n/2$ states, which writes $BB(n) + 1$ `1`s on the tape when started with a blank tape. However, then M_{2k}, as a $2k$-state TM, would write $BB(2k) + 1$ `1`s. Absurd.

5.17 *Need a solution.*

5.18 *Need a solution.*

5.19 *Need a solution.*

5.20 *Need a solution.*

5.21 The hint in the textbook is sufficient.

5.22 $\Leftarrow$: Trivial.

$\Rightarrow$: Let $f(x) = \langle M, x \rangle$, where TM M is A's recognizer.

5.23 $\Leftarrow$: Trivial, since `0*1*` is surely decidable.

$\Rightarrow$: Suppose there is an A's decider, then f defined as follows is computable.

$$f(x) = \begin{cases} \texttt{01}, & x \in A \\ \texttt{10}, & x \notin A \end{cases}$$

5.24 *Need a solution.*

5.25 $\overline{E_{\mathsf{TM}}} \leq_{\text{m}} A_{\mathsf{TM}}$ by problem 5.22 and it is well known that $A_{\mathsf{TM}} \leq_{\text{m}} E_{\mathsf{TM}}$. By the way, it is not difficult to construct an undecidable language B such that $B =_{\text{m}} \overline{B}$.