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 at least 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 $A = \{\langle R\rangle \mid R$ 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 $D = \{\langle x, y\rangle \mid x \in C$ and y is the computation history of M 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 By the pumping lemma, $L(G)$ is an infinite language $\iff$ $L(G)$ includes a string of at least length p. So we can first use $\mathit{INFINITE}_{\mathsf{PDA}}$ from Problem 4.11 to check whether $|L(G)| = \infty$, and then check whether $x \in L(G)$ for all x of less-than-p length, one by one.

4.30 Let E be A's enumerator which generates $\langle M_1\rangle, \langle M_2\rangle, \ldots$ in order. Construct $D = \{\langle n\rangle \mid \langle n\rangle \notin L(M_n)\}$.

4.31 First, recursively determine whether $V \stackrel{*}{\Rightarrow} w$ with some $w \in \Sigma^*$, for all variables V. Then, recursively determine whether $V \stackrel{*}{\Rightarrow} xAy$ with some $x, y \in \Sigma^*$, for all variables V.

4.32 Construct DPDA $P'_{(q,x)}$ by modifying P, such that $L(P'_{(q,x)}) = \varnothing$ if and only if (q, x) is a looping situation for P. Then we only need E_{PDA}.

5 Reducibility

5.9 Reduce from 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 from 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 from 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 Reduce from A_{TM}. To determine whether TM M accepts w, construct TM N which simulates M on N's own input w but never attempts to move its head left when its head is on the left-most tape cell unless when M accepts.

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 $\delta(q_i, a) = (q_j, b, \mathrm{L})$ to $\delta(q_i, a) = (q_{\text{accept}}, b, \mathrm{R})$. 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 (let k be sufficiently large), 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 for large n, 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 Obviously, in this case a PCP instance P always has a match unless all dominos in P have longer upper strings, or they all have longer lower strings.

5.18 Reduce from *PCP*, since any string with finite alphabet can be encoded to a binary one.

5.19 Trivial.

5.20 We can encode any language to the one over the unary alphabet.

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 It immediately follows from $A_{\mathsf{TM}}, \overline{A_{\mathsf{TM}}} \leq_{\mathrm{m}} J$.

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

5.26 *Need a solution.*

5.27 *Need a solution.*

⋆5.28 *Refer to the textbook.*

5.29 The case P is not nontrivial is trivial. As for the second condition, let $P = \{\langle M \rangle \mid \mathsf{TM}\ M \text{ has 100 states}\}$.

5.30 Trivial.

5.31 Let M be a TM which on input $\langle x \rangle$ ($x \in \mathbb{Z}^+$) calculates $x, f(x), f(f(x)), \ldots$ until it finds some $f^{(n)}(x) = 1$, and then accepts. Let N be a TM uses H to calculate whether $\langle M, x \rangle \in A_{\mathsf{TM}}$ for $x = 1, 2, \ldots$ in order, until it finds some $\langle M, x \rangle \notin A_{\mathsf{TM}}$, and then accepts. We have the positive answer to the $3x + 1$ problem if and only if $\langle N, \mathsf{0} \rangle \notin A_{\mathsf{TM}}$.

5.32 a. As hinted, reduce from *PCP*.

b. Reduce from $OVERLAP_{\mathsf{CFG}}$ by constructing a grammar whose rules are G's and H's rules and $S \to S_G\$ \mid S_H\$\$$, where S_G and S_H are G's and H's start variables.

5.33 *Need a solution.*

5.34 *Need a solution.*

5.35 *Need a solution.*

*5.36 See `https://cstheory.stackexchange.com/q/39407/46760` for two different solutions.

6 Advanced Topics in Computability Theory

6.6 Let $M = P_{\langle N \rangle}$ and N print $\langle M \rangle = q(\langle N \rangle)$.

6.7 A TM that always loops.

*6.8 Suppose for the sake of contradiction that f is a reduction from EQ_{TM} to $\overline{EQ_{\mathsf{TM}}}$. It is easy to generalize the fixed-point version of the recursion theorem to find $f(\langle M, N \rangle) = \langle M', N' \rangle$ such that M, N simulate M', N' respectively. Then $\langle M, N \rangle \in EQ_{\mathsf{TM}} \iff \langle M', N' \rangle \in \overline{EQ_{\mathsf{TM}}} \iff \langle M, N \rangle \in \overline{EQ_{\mathsf{TM}}}$. Absurd.

6.9 *Refer to the textbook.*

6.10 *Refer to the textbook.*

*6.11 $(\mathbb{R}, =, <)$.

6.12 *Refer to the textbook.*

6.13 Since $\mathbb{Z}_m$ is finite, any sentence in the language of $\mathcal{F}_m$ can be decided by brute-force checking.

6.14 Let $J = \mathtt{0}A \cup \mathtt{1}B$.

6.15 Let $B = A^A_{\mathsf{TM}} = \{\langle M^A, w \rangle \mid M^A \text{ accepts } w\}$. Then apply any classical method used in proving undecidablity of A_{TM}.

*6.16 *Need a solution.*

*6.17 Let

$$A = \{\langle M, w \rangle \mid \mathsf{TM}\ M \text{ on input } w \text{ halts with } \mathtt{0} \text{ on its tape}\}$$
$$B = \{\langle M, w \rangle \mid \mathsf{TM}\ M \text{ on input } w \text{ halts with } \mathtt{1} \text{ on its tape}\}$$

If there is a C's decider N, we can construct TM M which on input w first run N on $\langle M, w \rangle$ to know that M would not halt with $x \in \{\mathtt{0}, \mathtt{1}\}$ on M's tape, and then violates it.

6.18 *Need a solution.*

6.19 $|\{L(M^A) \mid M^A \text{ is an oracle } \mathsf{TM}\}| \le |\{M^A \mid M^A \text{ is an oracle } \mathsf{TM}\}| \le \aleph_0 < 2^{\aleph_0} = |\{L \mid L \text{ is a language}\}|$.

6.20 *Need a solution.*

6.21 *Need a solution.*

6.22 *Need a solution.*

6.23 Reduce from Problem 6.24.

6.24 Reduce from Problem 6.25.

6.25 *Need a solution.*

*6.26 Suppose for the sake of contradiction that $\mathrm{K}(xy) \le \mathrm{K}(x) + \mathrm{K}(y) + c$ always holds. Define

$$f_n = \sum_{|x|=n} 2^{-\mathrm{K}(x)}.$$

Then $\mathrm{K}(xy) \le \mathrm{K}(x) + \mathrm{K}(y) + c \implies f_{n+m} \ge 2^{-c} f_n f_m \implies f_{kn} \ge (2^{-c} f_n)^k$. On the other hand, Corollary 6.30 implies $f_n \le n + 1$. Therefore,

$$f_n \le 2^c \sqrt[k]{f_{kn}} \le 2^c \sqrt[k]{kn+1}.$$

Letting $k \to +\infty$ we obtain that $f_n \leq 2^c$ for all n. However, there is a TM M which on input $\langle p, q\rangle$ ($p, q \in \mathbb{N}$ and $q < 2^{2^p}$) halts with $r(p,q)$, a 2^p-bits binary representation of q, on its tape. So $\mathrm{K}(r(p,q)) \leq 2\log_2 p + \log_2 q + d$ with some constant d. Then, if $n = 2^p$ for some large p,

$$f_n = \sum_{|x|=n} 2^{-\mathrm{K}(x)} \geq \sum_{q<2^n} 2^{-\mathrm{K}(r(p,q))} \geq 2^{-2\log_2 p - d} \sum_{q<2^n} \frac{1}{q} \geq \frac{n}{2^d (\log n)^2},$$

which apparently contradicts with $f_n \leq 2^c$.

6.27 *Need a solution.*

6.28
a. $x = 0 \iff \forall y,\ x + y = y$
b. $x = 1 \iff \forall y,\ y = 0 \wedge x + y = 1$
c. $x = y \iff \forall z,\ z = 0 \wedge x + z = y$
d. $x < y \iff \exists z,\ \neg(z = 0) \wedge x + z = y$

7 Time Complexity

7.13 $a^b = (a^{\lfloor b/2 \rfloor})^2 \cdot a^{b \bmod 2}$, where $a, b \in \mathbb{N}$.

7.14 $q^t = (q^{\lfloor t/2 \rfloor})^2 \cdot q^{t \bmod 2}$, where $q \in S_k$ and $t \in \mathbb{N}$.

7.15 The hint in the textbook is sufficient.

7.16 *Refer to the textbook.*

7.17 A brute-force algorithm for *UNARY-SSUM* is already in P, due to the inefficiency of unary representation.

7.18 First $A \in \mathrm{P} = \mathrm{NP}$. Then since there exist $x \in A$ and $y \notin A$, for an arbitrary language $B \in \mathrm{NP} = \mathrm{P}$, f defined as follows is polynomial time computable.

$$f(w) = \begin{cases} x, & w \in B \\ y, & w \notin B \end{cases}$$

*7.19 Let the certificate of $q \in \mathbb{P}$ consist of

- $g \in \mathbb{Z}_m^*$ such that $g^{m-1} = 1$ and $g^{(m-1)/q} \neq 1$ for all prime $q \mid m-1$,
- the standard factorization of $m - 1 = \prod q_i^{r_i}$,
- certificates of $q_i \in \mathbb{P}$.

7.20 *There is something wrong with the problem.*

7.21 a. Modify the proof of *PATH* $\in$ P.

b. Reduce from *UHAMPATH* by setting k to the amount of nodes in G minus 1.

7.22 Reduce from *SAT*. In order to determine whether $\langle \phi \rangle \in SAT$, construct $\phi' = \phi \wedge (z \vee \overline{z})$.

7.23 *Refer to the textbook.*

7.24 *Need a solution.*

7.25 *Need a solution.*

7.26 *Need a solution.*

7.27