

# Solution to Introduction to the Theory of Computation

Lwins\_Lights

## Contents

<b>1</b>	<b>Regular Languages</b>	<b>2</b>
<b>2</b>	<b>Context-Free Languages</b>	<b>3</b>
<b>3</b>	<b>The Church–Turing Thesis</b>	<b>4</b>
<b>4</b>	<b>Decidability</b>	<b>5</b>
<b>5</b>	<b>Reducibility</b>	<b>7</b>
<b>6</b>	<b>Advanced Topics in Computability Theory</b>	<b>9</b>
<b>7</b>	<b>Time Complexity</b>	<b>11</b>

# 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  $E_{\text{PDA}}$ 's decider to decide whether  $L(M) \cap R = \emptyset$ .
- 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  $EQ_{\text{DFA}}$ 's decider.
- 4.14 *Refer to the textbook.*
- \*4.15 By the pumping lemma,  $1^k \in L(G) \implies 1^{k+p!} \in L(G)$  for every  $k \geq p$ . Therefore  $1^* \subseteq L(G) \iff \{1^k \mid k \leq p + p!\} \subseteq L(G)$ , which can be easily checked by a TM in finite time.
- 4.16 Since  $A = \{\langle R \rangle \mid R \text{ is a regular expression, } R \cap \Sigma^* 111 \Sigma^* \neq \emptyset\}$ , we only need an  $E_{\text{DFA}}$ '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,  $L(D) \cap (\Sigma \cup \epsilon)^p = \emptyset \implies L(D) = \emptyset$ , 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.,  $A_{\text{TM}}$ . 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  $\bar{A}$ 's recognizer and  $\bar{B}$ 's recognizer on its own input.  $M$  accepts when  $\bar{B}$ 's recognizer accepts and rejects when  $\bar{A}$ '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  $EQ_{\text{DFA}}$ '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  $\emptyset$ . Therefore we can use  $E_{\text{PDA}}$ 's decider to solve it.
- \*4.25 *Refer to the textbook.*
- \*4.26  $M$  is a DFA that accepts some palindrome  $\iff \text{CFL } \{w \mid w = w^{\mathcal{R}}\} \cap L(M) \text{ is not empty, which can be decided by } E_{\text{PDA}} \text{'s decider.}$
- \*4.27 Refer to the solution to Problem 4.26.
- 4.28  $x$  is a substring of some  $y \in L(G) \iff L(G) \cap \Sigma^* x \Sigma^* \neq \emptyset$ , which can be decided by  $E_{\text{PDA}}$ '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  $INFINITE_{\text{PDA}}$  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  $\langle M_1 \rangle, \langle M_2 \rangle, \dots$  in order. Construct  $D = \{\langle n \rangle \mid \langle n \rangle \notin L(M_n)\}$ .

- 4.31 First, recursively determine whether  $V \xRightarrow{*} w$  with some  $w \in \Sigma^*$ , for all variables  $V$ . Then, recursively determine whether  $V \xRightarrow{*} xAy$  with some  $x, y \in \Sigma^*$ , for all variables  $V$ .
- 4.32 Construct DPDA  $P'_{(q,x)}$  by modifying  $P$ , such that  $L(P'_{(q,x)}) = \emptyset$  if and only if  $(q, x)$  is a looping situation for  $P$ . Then we only need  $E_{\text{PDA}}$ .

## 5 Reducibility

- 5.9 Reduce from  $A_{\text{TM}}$ . 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  $E_{\text{TM}}$ . 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  $E_{\text{TM}}$ . 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  $q_{\text{accept}}$ .
- 5.14 Reduce from  $A_{\text{TM}}$ . 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  $\delta(q_i, a) = (q_j, b, L)$  to  $\delta(q_i, a) = (q_{\text{accept}}, b, R)$ . 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  $BB(n) + 1$  1s 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  $BB(n) + 1$  1s on the tape when started with a blank tape. However, then  $M_{2k}$ , as a  $2k$ -state TM, would write  $BB(2k) + 1$  1s. Absurd.
- 5.17 Obviously, in this case a PCP instance  $P$  always has a match unless all dominos in  $P$  have longer top strings, or they all have longer bottom 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.
- $$f(x) = \begin{cases} 01, & x \in A \\ 10, & x \notin A \end{cases}$$
- 5.24 It immediately follows from  $A_{\text{TM}}, \overline{A_{\text{TM}}} \leq_m J$ .
- 5.25  $\overline{E_{\text{TM}}} \leq_m A_{\text{TM}}$  by Problem 5.22 and it is well known that  $A_{\text{TM}} \leq_m E_{\text{TM}}$ . By the way, it is not difficult to construct an undecidable language  $B$  such that  $B =_m \overline{B}$ .
- 5.26 The idea is the same as how we deal with  $A_{\text{LBA}}$  and  $E_{\text{LBA}}$ .
- 5.27 Prove that  $A_{\text{TM}} \leq_m E_{2\text{DIM-DFA}} \leq_m EQ_{2\text{DIM-DFA}}$ , in which the former reduction can be done by computation history method.

\*5.28 Refer to the textbook.

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

5.30 Trivial.

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

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

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

5.33 Refer to the proof of undecidability of  $ALL_{\text{CFG}}$ . Let  $w = \#C_1\#C_3\#C_5\#\dots\#C_6^R\#C_4^R\#C_2^R\#$ .

5.34 Reduce from  $A_{\text{TM}}$ . To determine whether TM  $N$  accepts  $w$ , construct TM  $M$  which simulates  $N$  on  $M$ 's own input  $w$  but never modifies the portion of the tape that contains the input  $w$  unless when  $N$  accepts.

5.35 a. Just enumerate  $w$ .

b. Reduce from  $ALL_{\text{CFG}}$ . In order to determine whether  $L(G) = \Sigma^*$ , construct a grammar whose rules are  $G$ 's rules and  $S \rightarrow S_G \mid T$ ;  $T \rightarrow aT \mid \epsilon$  ( $a \in \Sigma$ ), where  $S_G$  are  $G$ 's start variable.

\*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  $q(\langle N \rangle) = \langle M \rangle$ .

6.7 A TM that always loops.

\*6.8 Suppose for the sake of contradiction that  $f$  is a reduction from  $EQ_{TM}$  to  $\overline{EQ_{TM}}$ . 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  $\langle M, N \rangle \in EQ_{TM} \iff \langle M', N' \rangle \in \overline{EQ_{TM}} \iff \langle M, N \rangle \in \overline{EQ_{TM}}$ . 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  $J = 0A \cup 1B$ .

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

\*6.16 (**Kleene–Post**) For convenience let any language  $L$  be a subset of  $\mathbb{N}$  instead of  $\Sigma^*$ . Denote  $\{0, 1, \dots, m\}$  by  $[m]$ . Define

$$\mathcal{L}_m(A) = \{L \subseteq \mathbb{N} \mid L \cap [m] = A\} \quad (A \subseteq [m])$$

Let  $M_0, M_1, M_2, \dots$  be all possible oracle TMs. We will give two series of families of languages  $\mathcal{A}_0 \supseteq \mathcal{A}_1 \supseteq \mathcal{A}_2 \supseteq \dots$  and  $\mathcal{B}_0 \supseteq \mathcal{B}_1 \supseteq \mathcal{B}_2 \supseteq \dots$  such that

$$\forall A \in \mathcal{A}_n, B \in \mathcal{B}_n, M_n^A \text{ is not } B\text{'s decider and } M_n^B \text{ is not } A\text{'s decider.}$$

Then taking arbitrary  $A \in \mathcal{A} = \bigcap_{n \in \mathbb{N}} \mathcal{A}_n$  and  $B \in \mathcal{B} = \bigcap_{n \in \mathbb{N}} \mathcal{B}_n$  we have  $A \not\leq_T B$  and  $B \not\leq_T A$ . We build them by induction. Given  $\mathcal{A}_{n-1} = \mathcal{L}_m(X)$  and  $\mathcal{B}_{n-1} = \mathcal{L}_m(Y)$ , we first find  $\mathcal{A}'_n \subseteq \mathcal{A}_{n-1}$  and  $\mathcal{B}'_n \subseteq \mathcal{B}_{n-1}$  such that  $\forall A \in \mathcal{A}'_n, B \in \mathcal{B}'_n, M_n^A$  is not  $B$ 's decider.

- If there is no  $A \in \mathcal{A}_{n-1}$  such that  $M_n^A$  is a decider, let  $\mathcal{A}'_n = \mathcal{A}_{n-1}$  and  $\mathcal{B}'_n = \mathcal{B}_{n-1}$ .
- Suppose  $M_n^A$  is a decider with some  $A \in \mathcal{A}_{n-1}$ , then there exists an  $m' > m$  such that

$$\forall A' \in \mathcal{L}_{m'}(A \cap [m']), m+1 \in L(M_n^A) \iff m+1 \in L(M_n^{A'}).$$

Then, let  $\mathcal{A}'_n = \mathcal{L}_{m'}(A \cap [m'])$ ,  $\mathcal{B}'_n = \mathcal{L}_{m'}(Y)$  or  $\mathcal{L}_{m'}(Y \cup \{m+1\})$ , depending on whether  $m+1 \in L(M_n^A)$ .

The same method can be also used to find  $\mathcal{A}_n \subseteq \mathcal{A}'_n$  and  $\mathcal{B}_n \subseteq \mathcal{B}'_n$  such that  $\forall A \in \mathcal{A}_n, B \in \mathcal{B}_n, M_n^B$  is not  $A$ 's decider.

\*6.17 Let

$$\begin{aligned} A &= \{\langle M, w \rangle \mid \text{TM } M \text{ on input } w \text{ halts with 0 on its tape}\} \\ B &= \{\langle M, w \rangle \mid \text{TM } M \text{ on input } w \text{ halts with 1 on its tape}\} \end{aligned}$$

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  $x \in \{0, 1\}$  on  $M$ 's tape, and then violates it.

6.18 Suppose  $L(M) \neq L(N)$ , we can enumerate  $x$  to find one such that  $\langle M, x \rangle \in A_{TM} \oplus \langle N, x \rangle \in A_{TM}$  holds, where  $\oplus$  means exclusive or.

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

- 6.20 Let  $M$  be  $PCP$ 's recognizer. Then check whether  $\langle M, \langle P \rangle \rangle \in A_{\text{TM}}$  to know if instance  $P$  has a match.
- 6.21 Since  $K(x) \leq |x| + c$ , we can check all possible minimal description  $\langle M, w \rangle$  to see if  $M$  on input  $w$  halts with  $x$  on its tape by simulating  $M$ , where “possible” means that  $|\langle M, w \rangle| < |x| + c$  and  $\langle M, w \rangle \in A_{\text{TM}}$ .
- 6.22 Trivial.
- 6.23 Reduce from Problem 6.24.
- 6.24 Reduce from Problem 6.25.
- 6.25 If not, there is an enumerator  $E$  which would print infinite many incompressible strings one by one:  $s_1, s_2, \dots$ . By using  $E$  we can construct TM  $M$ , which prints an incompressible string  $s_i$ , such that  $|s_i| > |\langle M, 0 \rangle|$ , on its tape. Then we have  $K(s_i) \leq |\langle M, 0 \rangle| < |s_i|$ . Absurd.
- \*6.26 Suppose for the sake of contradiction that  $K(xy) \leq K(x) + K(y) + c$  always holds. Define

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

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

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

Letting  $k \rightarrow +\infty$  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  $K(r(p, q)) \leq 2 \log_2 p + \log_2 q + d$  with some constant  $d$ . Then, if  $n = 2^p$  for some large  $p$ ,

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

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

6.27 Show that  $\overline{HALT_{\text{TM}}} \leq_m S, \bar{S}$ .

- 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 Use dynamic programming. Denote  $dp[i][j] = \mathbf{1}\{\langle \{x_1, \dots, x_i\}, j \rangle \in SUBSET-SUM\}$ , where  $\alpha \implies \mathbf{1}\{\alpha\} = 1$  and  $\neg\alpha \implies \mathbf{1}\{\alpha\} = 0$ .

7.18 First  $A \in P = NP$ . Then since there exist  $x \in A$  and  $y \notin A$ , for an arbitrary language  $B \in NP = P$ ,  $f$  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 It follows from the result of Problem 7.18.

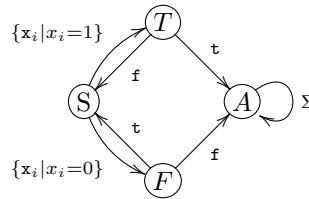
7.21 a. Modify the proof of  $PATH \in P$ .

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  $\phi' = \phi \wedge (z \vee \bar{z})$ .

7.23 Refer to the textbook.

\*7.36 Reduce from  $3SAT$ . If there is no limitation on  $\Sigma$ , let the following DFA correspond to an assignment to variables  $x_1, x_2, \dots, x_n$  in 3cnf-formula  $\phi$ . You may need some extra states and transitions.



Once you solved the problem without regard to the limitation on  $\Sigma$ , based on your solution, consider how to build a reduction where  $\Sigma = \{0, 1\}$ .

7.37 Using the computation history as certificate we easily obtain  $U \in NP$ . And it is easy to show that  $3SAT \leq_P U$ , by designing an NTM  $M$ , which accepts  $\langle \phi \rangle$  in polynomial time on at least one branch if and only if  $\langle \phi \rangle \in 3SAT$ .

\*7.38 Let  $\phi(x/t)$  denote the Boolean formula  $\phi$  after replacing every existence of  $x$  in  $\phi$  with  $t$ . Suppose there are  $n$  variables  $x_1, \dots, x_n$  in  $\phi$ .  $\langle \phi \rangle \in SAT \implies \langle \phi(x_1/0) \rangle \in SAT$  or  $\langle \phi(x_1/1) \rangle \in SAT$ , so we can directly assign 0 or 1 to  $x_1$ . Recursively assigning  $x_2, \dots, x_n$  in this way we have done.

\*7.39 Construct language  $L = \{\langle n, x, y \rangle \mid n \text{ has a nontrivial factor in interval } [x, y]\}$ , which is apparently in  $NP = P$ . Then refer to the solution to Problem 7.38.

\*7.40 Refer to the textbook.