For the other direction, if both A and $\overline{A}$ are Turing-recognizable, we let M_1 be the recognizer for A and M_2 be the recognizer for $\overline{A}$. The following Turing machine M is a decider for A.

$M =$ "On input w:
1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, *accept*; if M_2 accepts, *reject*."

Running the two machines in parallel means that M has two tapes, one for simulating M_1 and the other for simulating M_2. In this case, M takes turns simulating one step of each machine, which continues until one of them accepts.

Now we show that M decides A. Every string w is either in A or $\overline{A}$. Therefore, either M_1 or M_2 must accept w. Because M halts whenever M_1 or M_2 accepts, M always halts and so it is a decider. Furthermore, it accepts all strings in A and rejects all strings not in A. So M is a decider for A, and thus A is decidable.

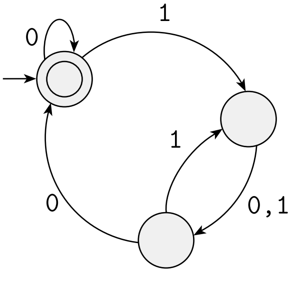
COROLLARY 4.23

$\overline{A_{\mathsf{TM}}}$ is not Turing-recognizable.

PROOF We know that A_{TM} is Turing-recognizable. If $\overline{A_{\mathsf{TM}}}$ also were Turing-recognizable, A_{TM} would be decidable. Theorem 4.11 tells us that A_{TM} is not decidable, so $\overline{A_{\mathsf{TM}}}$ must not be Turing-recognizable.

EXERCISES

[A]**4.1** Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, \texttt{0100}\rangle \in A_{\mathsf{DFA}}$?
b. Is $\langle M, \texttt{011}\rangle \in A_{\mathsf{DFA}}$?
c. Is $\langle M\rangle \in A_{\mathsf{DFA}}$?
d. Is $\langle M, \texttt{0100}\rangle \in A_{\mathsf{REX}}$?
e. Is $\langle M\rangle \in E_{\mathsf{DFA}}$?
f. Is $\langle M, M\rangle \in EQ_{\mathsf{DFA}}$?

4.2 Consider the problem of determining whether a DFA and a regular expression are equivalent. Express this problem as a language and show that it is decidable.

4.3 Let $ALL_{\mathsf{DFA}} = \{\langle A\rangle \mid A$ is a DFA and $L(A) = \Sigma^*\}$. Show that ALL_{DFA} is decidable.

4.4 Let $A\varepsilon_{\mathsf{CFG}} = \{\langle G\rangle \mid G$ is a CFG that generates $\varepsilon\}$. Show that $A\varepsilon_{\mathsf{CFG}}$ is decidable.

[A]**4.5** Let $E_{\mathsf{TM}} = \{\langle M\rangle \mid M$ is a TM and $L(M) = \emptyset\}$. Show that $\overline{E_{\mathsf{TM}}}$, the complement of E_{TM}, is Turing-recognizable.

4.6 Let X be the set $\{1, 2, 3, 4, 5\}$ and Y be the set $\{6, 7, 8, 9, 10\}$. We describe the functions $f\colon X\longrightarrow Y$ and $g\colon X\longrightarrow Y$ in the following tables. Answer each part and give a reason for each negative answer.

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

[A]**a.** Is f one-to-one?
b. Is f onto?
c. Is f a correspondence?
[A]**d.** Is g one-to-one?
e. Is g onto?
f. Is g a correspondence?

4.7 Let $\mathcal{B}$ be the set of all infinite sequences over $\{\mathtt{0,1}\}$. Show that $\mathcal{B}$ is uncountable using a proof by diagonalization.

4.8 Let $T = \{(i, j, k) \mid i, j, k \in \mathcal{N}\}$. Show that T is countable.

4.9 Review the way that we define sets to be the same size in Definition 4.12 (page 203). Show that "is the same size" is an equivalence relation.

PROBLEMS

[A]**4.10** Let *INFINITE*$_{\mathsf{DFA}} = \{\langle A\rangle \mid A$ is a DFA and $L(A)$ is an infinite language$\}$. Show that *INFINITE*$_{\mathsf{DFA}}$ is decidable.

4.11 Let $INFINITE_{\mathsf{PDA}} = \{\langle M\rangle \mid M$ is a PDA and $L(M)$ is an infinite language$\}$. Show that $INFINITE_{\mathsf{PDA}}$ is decidable.

[A]**4.12** Let $A = \{\langle M\rangle \mid M$ is a DFA that doesn't accept any string containing an odd number of $\mathtt{1}$s$\}$. Show that A is decidable.

4.13 Let $A = \{\langle R, S\rangle \mid R$ and S are regular expressions and $L(R) \subseteq L(S)\}$. Show that A is decidable.

[A]**4.14** Let $\Sigma = \{\mathtt{0,1}\}$. Show that the problem of determining whether a CFG generates some string in $\mathtt{1}^*$ is decidable. In other words, show that

$$\{\langle G\rangle \mid G \text{ is a CFG over } \{\mathtt{0,1}\} \text{ and } \mathtt{1}^* \cap L(G) \neq \emptyset\}$$

is a decidable language.

*4.15 Show that the problem of determining whether a CFG generates all strings in $\texttt{1}^*$ is decidable. In other words, show that $\{\langle G\rangle |\ G$ is a CFG over $\{\texttt{0,1}\}$ and $\texttt{1}^* \subseteq L(G)\}$ is a decidable language.

4.16 Let $A = \{\langle R\rangle |\ R$ is a regular expression describing a language containing at least one string w that has $\texttt{111}$ as a substring (i.e., $w = x\texttt{111}y$ for some x and y)$\}$. Show that A is decidable.

4.17 Prove that EQ_{DFA} is decidable by testing the two DFAs on all strings up to a certain size. Calculate a size that works.

*4.18 Let C be a language. Prove that C is Turing-recognizable iff a decidable language D exists such that $C = \{x |\ \exists y\ (\langle x, y\rangle \in D)\}$.

*4.19 Prove that the class of decidable languages is not closed under homomorphism.

4.20 Let A and B be two disjoint languages. Say that language C ***separates*** A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-Turing-recognizable languages are separable by some decidable language.

4.21 Let $S = \{\langle M\rangle |\ M$ is a DFA that accepts $w^{\mathcal{R}}$ whenever it accepts $w\}$. Show that S is decidable.

4.22 Let $\mathit{PREFIX\text{-}FREE}_{\mathsf{REX}} = \{\langle R\rangle |\ R$ is a regular expression and $L(R)$ is prefix-free$\}$. Show that $\mathit{PREFIX\text{-}FREE}_{\mathsf{REX}}$ is decidable. Why does a similar approach fail to show that $\mathit{PREFIX\text{-}FREE}_{\mathsf{CFG}}$ is decidable?

^{A*}4.23 Say that an NFA is ***ambiguous*** if it accepts some string along two different computation branches. Let $\mathit{AMBIG}_{\mathsf{NFA}} = \{\langle N\rangle |\ N$ is an ambiguous NFA$\}$. Show that $\mathit{AMBIG}_{\mathsf{NFA}}$ is decidable. (Suggestion: One elegant way to solve this problem is to construct a suitable DFA and then run E_{DFA} on it.)

4.24 A ***useless state*** in a pushdown automaton is never entered on any input string. Consider the problem of determining whether a pushdown automaton has any useless states. Formulate this problem as a language and show that it is decidable.

^{A*}4.25 Let $\mathit{BAL}_{\mathsf{DFA}} = \{\langle M\rangle |\ M$ is a DFA that accepts some string containing an equal number of 0s and 1s$\}$. Show that $\mathit{BAL}_{\mathsf{DFA}}$ is decidable. (Hint: Theorems about CFLs are helpful here.)

*4.26 Let $\mathit{PAL}_{\mathsf{DFA}} = \{\langle M\rangle |\ M$ is a DFA that accepts some palindrome$\}$. Show that $\mathit{PAL}_{\mathsf{DFA}}$ is decidable. (Hint: Theorems about CFLs are helpful here.)

*4.27 Let $E = \{\langle M\rangle |\ M$ is a DFA that accepts some string with more 1s than 0s$\}$. Show that E is decidable. (Hint: Theorems about CFLs are helpful here.)

4.28 Let $C = \{\langle G, x\rangle |\ G$ is a CFG x is a substring of some $y \in L(G)\}$. Show that C is decidable. (Hint: An elegant solution to this problem uses the decider for E_{CFG}.)

4.29 Let $C_{\mathsf{CFG}} = \{\langle G, k\rangle |\ G$ is a CFG and $L(G)$ contains exactly k strings where $k \geq 0$ or $k = \infty\}$. Show that C_{CFG} is decidable.

4.30 Let A be a Turing-recognizable language consisting of descriptions of Turing machines, $\{\langle M_1\rangle, \langle M_2\rangle, \ldots\}$, where every M_i is a decider. Prove that some decidable language D is not decided by any decider M_i whose description appears in A. (Hint: You may find it helpful to consider an enumerator for A.)

4.31 Say that a variable A in CFL G is ***usable*** if it appears in some derivation of some string $w \in G$. Given a CFG G and a variable A, consider the problem of testing whether A is usable. Formulate this problem as a language and show that it is decidable.

4.32 The proof of Lemma 2.41 says that (q, x) is a *looping situation* for a DPDA P if when P is started in state q with $x \in \Gamma$ on the top of the stack, it never pops anything below x and it never reads an input symbol. Show that F is decidable, where $F = \{\langle P, q, x\rangle |\ (q, x) \text{ is a looping situation for } P\}$.

SELECTED SOLUTIONS

4.1 **(a)** Yes. The DFA M accepts `0100`.

(b) No. M doesn't accept `011`.

(c) No. This input has only a single component and thus is not of the correct form.

(d) No. The first component is not a regular expression and so the input is not of the correct form.

(e) No. M's language isn't empty.

(f) Yes. M accepts the same language as itself.

4.5 Let $s_1, s_2, \ldots$ be a list of all strings in Σ^*. The following TM recognizes $\overline{E_{\text{TM}}}$.

"On input $\langle M\rangle$, where M is a TM:
1. Repeat the following for $i = 1, 2, 3, \ldots$.
2. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
3. If M has accepted any of these, *accept*. Otherwise, continue."

4.6 **(a)** No, f is not one-to-one because $f(1) = f(3)$.

(d) Yes, g is one-to-one.

4.10 The following TM I decides $\mathit{INFINITE}_{\text{DFA}}$.

I = "On input $\langle A\rangle$, where A is a DFA:
1. Let k be the number of states of A.
2. Construct a DFA D that accepts all strings of length k or more.
3. Construct a DFA M such that $L(M) = L(A) \cap L(D)$.
4. Test $L(M) = \emptyset$ using the E_{DFA} decider T from Theorem 4.4.
5. If T accepts, *reject*; if T rejects, *accept*."

This algorithm works because a DFA that accepts infinitely many strings must accept arbitrarily long strings. Therefore, this algorithm accepts such DFAs. Conversely, if the algorithm accepts a DFA, the DFA accepts some string of length k or more, where k is the number of states of the DFA. This string may be pumped in the manner of the pumping lemma for regular languages to obtain infinitely many accepted strings.

4.12 The following TM decides A.

"On input $\langle M \rangle$:

1. Construct a DFA O that accepts every string containing an odd number of 1s.
2. Construct a DFA B such that $L(B) = L(M) \cap L(O)$.
3. Test whether $L(B) = \emptyset$ using the E_{DFA} decider T from Theorem 4.4.
4. If T accepts, *accept*; if T rejects, *reject*."

4.14 You showed in Problem 2.18 that if C is a context-free language and R is a regular language, then $C \cap R$ is context free. Therefore, $1^* \cap L(G)$ is context free. The following TM decides the language of this problem.

"On input $\langle G \rangle$:

1. Construct CFG H such that $L(H) = 1^* \cap L(G)$.
2. Test whether $L(H) = \emptyset$ using the E_{CFG} decider R from Theorem 4.8.
3. If R accepts, *reject*; if R rejects, *accept*."

4.23 The following procedure decides $AMBIG_{\mathsf{NFA}}$. Given an NFA N, we design a DFA D that simulates N and accepts a string iff it is accepted by N along two different computational branches. Then we use a decider for E_{DFA} to determine whether D accepts any strings.

Our strategy for constructing D is similar to the NFA-to-DFA conversion in the proof of Theorem 1.39. We simulate N by keeping a pebble on each active state. We begin by putting a red pebble on the start state and on each state reachable from the start state along ε transitions. We move, add, and remove pebbles in accordance with N's transitions, preserving the color of the pebbles. Whenever two or more pebbles are moved to the same state, we replace its pebbles with a blue pebble. After reading the input, we accept if a blue pebble is on an accept state of N or if two different accept states of N have red pebbles on them.

The DFA D has a state corresponding to each possible position of pebbles. For each state of N, three possibilities occur: It can contain a red pebble, a blue pebble, or no pebble. Thus, if N has n states, D will have 3^n states. Its start state, accept states, and transition function are defined to carry out the simulation.

4.25 The language of all strings with an equal number of 0s and 1s is a context-free language, generated by the grammar $S \rightarrow 1S0S \mid 0S1S \mid \varepsilon$. Let P be the PDA that recognizes this language. Build a TM M for BAL_{DFA}, which operates as follows. On input $\langle B \rangle$, where B is a DFA, use B and P to construct a new PDA R that recognizes the intersection of the languages of B and P. Then test whether R's language is empty. If its language is empty, *reject*; otherwise, *accept*.