

- 4.28 Let  $A$  be a Turing-recognizable language consisting of descriptions of Turing machines,  $\{\langle M_1 \rangle, \langle M_2 \rangle, \dots\}$ , 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  $A$ .)

**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 (a) No,  $f$  is not one-to-one because  $f(1) = f(3)$ .  
 (d) Yes,  $g$  is one-to-one.
- 4.9 The following TM  $I$  decides  $INFINITE_{DFA}$ .

$I =$  "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 which 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.

- 4.11 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 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*."

- 4.13 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  $A$ .

"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*.”

- 4.21 The following procedure decides  $AMBIG_{NFA}$ . 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 along  $\epsilon$  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 states of  $N$ .

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.

- 4.23 The language of all strings with an equal number of 0s and 1s is a context free language, generated by the grammar  $S \rightarrow 1S0S \mid 0S1S \mid \epsilon$ . 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*.