

- 3.18 Prove that there exists an infinite decidable subset of  $A_{TM}$ .
- \*3.19 Show that a language is decidable iff some enumerator enumerates the language in lexicographic order.
- \*3.20 Show that single-tape TMs that cannot write on the portion of the tape containing the input string recognize only regular languages.
- 3.21 Let  $c_1x^n + c_2x^{n-1} + \dots + c_nx + c_{n+1}$  be a polynomial with a root at  $x = x_0$ . Let  $c_{\max}$  be the largest absolute value of a  $c_i$ . Show that

$$|x_0| < (n + 1) \frac{c_{\max}}{|c_1|}.$$

- <sup>A</sup>3.22 Let  $A$  be the language containing only the single string  $s$ , where

$$s = \begin{cases} 0 & \text{if life never will be found on Mars.} \\ 1 & \text{if life will be found on Mars someday.} \end{cases}$$

Is  $A$  decidable? Why or why not? For the purposes of this problem, assume that the question of whether life will be found on Mars has an unambiguous YES or NO answer.

### SELECTED SOLUTIONS

3.1 (b)  $q_1 00, \sqcup q_2 0, \sqcup x q_3 \sqcup, \sqcup q_5 x \sqcup, q_5 \sqcup x \sqcup, \sqcup q_2 x \sqcup, \sqcup x q_2 \sqcup, \sqcup x \sqcup q_{\text{accept}}$

3.2 (a)  $q_1 11, \sqcup q_3 1, \sqcup 1 q_3 \sqcup, \sqcup 1 \sqcup q_{\text{reject}}$ .

3.3 We prove both directions of the “iff.” First, if a language  $L$  is decidable, it can be decided by a deterministic Turing machine, and that is automatically a nondeterministic Turing machine.

Second, if a language  $L$  is decided by a nondeterministic TM  $N$ , we construct a deterministic TM  $D_2$  that decides  $L$ . Machine  $D_2$  runs the same algorithm that appears in the TM  $D$  described in the proof of Theorem 3.3, with an additional Stage 5: *Reject* if all branches of the nondeterminism of  $N$  are exhausted.

We argue that  $D_2$  is a decider for  $L$ . If  $N$  accepts its input,  $D_2$  will eventually find an accepting branch and accept, too. If  $N$  rejects its input, all of its branches halt and reject because it is a decider. Hence each of the branches has finitely many nodes, where each node represents one step of  $N$ 's computation along that branch. Therefore  $N$ 's entire computation tree on this input is finite, by virtue of the theorem about trees given in the statement of the exercise. Consequently  $D_2$  will halt and reject when this entire tree has been explored.

3.5 (a) Yes. The tape alphabet  $\Gamma$  contains  $\sqcup$ . A Turing machine can write any characters in  $\Gamma$  on its tape.

(b) No.  $\Sigma$  never contains  $\sqcup$ , but  $\Gamma$  always contains  $\sqcup$ . So they cannot be equal.

(c) Yes. If the Turing machine attempts to move its head off the left-hand end of the tape, it remains on the same tape cell.

(d) No. Any Turing machine must contain two distinct states  $q_{\text{accept}}$  and  $q_{\text{reject}}$ . So, a Turing machine contains at least two states.

3.8 (a)

“On input string  $w$ :

1. Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, go to stage 4. Otherwise, move the head back to the front of the tape.
2. Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, *reject*.
3. Move the head back to the front of the tape and go to stage 1.
4. Move the head back to the front of the tape. Scan the tape to see if any unmarked 1s remain. If none are found, *accept*; otherwise, *reject*.”

**3.10** We first simulate an ordinary Turing machine by a write-twice Turing machine. The write-twice machine simulates a single step of the original machine by copying the entire tape over to a fresh portion of the tape to the right-hand side of the currently used portion. The copying procedure operates character by character, marking a character as it is copied. This procedure alters each tape square twice, once to write the character for the first time and again to mark that it has been copied. The position of the original Turing machine’s tape head is marked on the tape. When copying the cells at, or adjacent to, the marked position, the tape contents is updated according to the rules of the original Turing machine.

To carry out the simulation with a write-once machine, operate as before, except that each cell of the previous tape is now represented by two cells. The first of these contains the original machine’s tape symbol and the second is for the mark used in the copying procedure. The input is not presented to the machine in the format with two cells per symbol, so the very first time the tape is copied, the copying marks are put directly over the input symbols.

**3.15 (a)** For any two decidable languages  $L_1$  and  $L_2$ , let  $M_1$  and  $M_2$  be the TMs that decide them. We construct a TM  $M'$  that decides the union of  $L_1$  and  $L_2$ :

“On input  $w$ :

1. Run  $M_1$  on  $w$ . If it accepts, *accept*.
2. Run  $M_2$  on  $w$ . If it accepts, *accept*. Otherwise, *reject*.”

$M'$  accepts  $w$  if either  $M_1$  or  $M_2$  accepts it. If both reject,  $M'$  rejects.

**3.16 (a)** For any two Turing-recognizable languages  $L_1$  and  $L_2$ , let  $M_1$  and  $M_2$  be the TMs that recognize them. We construct a TM  $M'$  that recognizes the union of  $L_1$  and  $L_2$ :

“On input  $w$ :

1. Run  $M_1$  and  $M_2$  alternatively on  $w$  step by step. If either accept, *accept*. If both halt and reject, *reject*.”

If either  $M_1$  and  $M_2$  accept  $w$ ,  $M'$  accepts  $w$  because the accepting TM arrives to its accepting state after a finite number of steps. Note that if both  $M_1$  and  $M_2$  reject and either of them does so by looping, then  $M'$  will loop.

**3.22** The language  $A$  is one of the two languages,  $\{0\}$  or  $\{1\}$ . In either case the language is finite, and hence decidable. If we aren’t able to determine which of these two languages is  $A$ , we won’t be able to describe the decider for  $A$ , but we can give two Turing machines, one of which is  $A$ ’s decider.