Proof: We first show that every recursive language can be enumerated in lexicographical order. Let $L$ be a recursive language over an alphabet $\Sigma$. Then it is accepted by some machine $M$ that halts for all input strings. A machine $E$ that enumerates $L$ in lexicographical order can be constructed from $M$ and the machine $E_{\Sigma^*}$ that enumerates $\Sigma^*$ in lexicographical order. The machine $E$ is a hybrid, interleaving the computations of $M$ and $E_{\Sigma^*}$. The computation of $E$ consists of the following loop:

1. The machine $E_{\Sigma^*}$ is run, producing a string $u \in \Sigma^*$.
2. $M$ is run with input $u$.
3. If $M$ accepts $u$, $u$ is written on the output tape of $E$.
4. The generate-and-test loop continues with step 1.

Since $M$ halts for all inputs, $E$ cannot enter a nonterminating computation in step 2. Thus, each string $u \in \Sigma^*$ will be generated and tested for membership in $L$.

Now we show that any language $L$ that can be enumerated in lexicographical order is recursive. This proof is divided into two cases based on the cardinality of $L$.

Case 1: $L$ is finite. Then $L$ is recursive since every finite language is recursive.

Case 2: $L$ is infinite. The argument is similar to that given in Theorem 8.8.2 except that the ordering is used to terminate the computation. As before, a $(k+1)$-tape machine $M$ accepting $L$ can be constructed from a $k$-tape machine $E$ that enumerates $L$ in lexicographical order. The additional tape of $M$ is the input tape; the remaining $k$ tapes allow $M$ to simulate the computations of $E$. The ordering of the strings produced by $E$ provides the information needed to halt $M$ when the input is not in the language. The computation of $M$ begins with a string $u$ on its input tape. Next $M$ simulates the computation of $E$. When the simulation produces a string $w$, $M$ compares $u$ with $w$. If $u = w$, then $M$ halts and accepts. If $w$ is greater than $u$ in the ordering, $M$ halts rejecting the input. Finally, if $w$ is less than $u$ in the ordering, then the simulation of $E$ is restarted to produce another element of $L$ and the comparison cycle is repeated.

Exercises

1. Let $M$ be the Turing machine defined by

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$B$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$, $B$, $R$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_2$, $B$, $L$</td>
<td>$q_1$, $a$, $R$</td>
<td>$q_1$, $c$, $R$</td>
<td>$q_1$, $c$, $R$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_2$, $c$, $L$</td>
<td>$q_2$, $b$, $L$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Trace the computation for the input string $aabca$.
b) Trace the computation for the input string $bcbc$. 
c) Give the state diagram of M.

d) Describe the result of a computation in M.

2. Let M be the Turing machine defined by

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>B</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$, B, R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$, B, R</td>
<td>$q_1$, a, R</td>
<td>$q_1$, b, R</td>
<td>$q_1$, c, L</td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td>$q_2$, b, L</td>
<td>$q_2$, a, L</td>
</tr>
</tbody>
</table>

a) Trace the computation for the input string $abcab$.

b) Trace the first six transitions of the computation for the input string $abab$.

c) Give the state diagram of M.

d) Describe the result of a computation in M.

3. Construct a Turing machine with input alphabet \{a, b\} to perform each of the following operations. Note that the tape head is scanning position zero in state $q_f$ whenever a computation terminates.

- a) Move the input one space to the right. Input configuration $q_0BuB$, result $q_fBBuB$.
- b) Concatenate a copy of the reversed input string to the input. Input configuration $q_0BuB$, result $q_fBu_iuB$.
- c) Insert a blank between each of the input symbols. For example, input configuration $q_0BabaB$, result $q_fBaBbBaB$.
- d) Erase the b's from the input. For example, input configuration $q_0BbabaababB$, result $q_fBaaaB$.

4. Construct a Turing machine with input alphabet \{a, b, c\} that accepts strings in which the first c is preceded by the substring $aaa$. A string must contain a c to be accepted by the machine.

5. Construct a Turing machine with input alphabet \{a, b\} to accept each of the following languages by final state.

- a) $a^ib^j | i \geq 0, j \geq i$
- b) $a^ib^ilc^j | i, j > 0$
- c) Strings with the same number of a's and b's
- d) $uu^R | u \in \{a, b\}^*$
- e) $uu | u \in \{a, b\}^*$

6. Modify your solution to Exercise 5(a) to obtain a Turing machine that accepts the language $a^ib^j | i \geq 0, j \geq i$ by halting.

7. An alternative method of acceptance by final state can be defined as follows: A string $u$ is accepted by a Turing machine $M$ if the computation of $M$ with input $u$ enters
Chapter 9  Turing Computable Functions

Exercises

1. Construct Turing machines with input alphabet \( \{a, b\} \) that compute the specified functions. The symbols \( u \) and \( v \) represent arbitrary strings over \( \{a, b\}^* \).
   a) \( f(u) = aaa \)
   b) \( f(u) = \begin{cases} a & \text{if } \text{length}(u) \text{ is even} \\ b & \text{otherwise} \end{cases} \)
   c) \( f(u) = u^R \)
   d) \( f(u, v) = \begin{cases} u & \text{if } \text{length}(u) > \text{length}(v) \\ v & \text{otherwise} \end{cases} \)

2. Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_f) \) be a Turing machine that computes the partial characteristic function of the language \( L \). Use \( M \) to build a standard Turing machine that accepts \( L \).

3. Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, F) \) be a standard Turing machine that accepts a language \( L \). Construct a machine \( M' \) that computes the partial characteristic function of \( L \). Recall that the tape of \( M' \) must have the form \( q_f B 0B \) or \( q_f B 1B \) upon the completion of a computation of \( \hat{x}_L \).

4. Let \( L \) be a language over \( \Sigma \) and let
   \[ \chi_L(u) = \begin{cases} 1 & \text{if } u \in L \\ 0 & \text{otherwise} \end{cases} \]

be the characteristic function of \( L \).
   a) If \( \chi_L \) is Turing computable, prove that \( L \) is recursive.
   b) If \( L \) is recursive, prove that there is a Turing machine that computes \( \chi_L \).

5. Construct Turing machines that compute the following number-theoretic functions and relations. Do not use macros in the design of these machines.
   a) \( f(n) = 2n + 3 \)
   b) \( \text{half}(n) = \lfloor n/2 \rfloor \) where \( \lfloor x \rfloor \) is the greatest integer less than or equal to \( x \)
   c) \( f(n_1, n_2, n_3) = n_1 + n_2 + n_3 \)
   d) \( \text{even}(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ 0 & \text{otherwise} \end{cases} \)
   e) \( \text{eq}(n, m) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{otherwise} \end{cases} \)
   f) \( \text{lt}(n, m) = \begin{cases} 1 & \text{if } n < m \\ 0 & \text{otherwise} \end{cases} \)

6. Construct the configuration:
   a) ZR;
   b) FL;
   c) E;
   d) T; u
   e) BRI
   f) INI

7. Use the machines:
   a) \( f(n) \)
   b) \( f(n) \)
   c) \( f(n) \)
   d) \( f(n) \)
   e) \( f(n) \)

8. Design the machine
   a) \( g(n) \)
   b) \( p(n) \)
   c) \( d(n) \)

9. Trace
   a) \( n \)
   b) \( n \)
   c) \( n \)

10. Describe:
    a) \( \mu \)
    b) \( \rho \)
    c) \( m \)
    d) \( m \)