Exercise 9.1.1

What strings are

(a) w_{37}? 37 has binary representation 100101. The construction from section 9.1.1 tells us that a string w corresponds to integer $1w$. So $w = 00101$.

(b) w_{100}? 100 has binary representation 1100100, so the string w so that $1w = 1100100$ is $w = 100100$.

Exercise 9.1.3

Here are two definitions of languages that are similar to the definition of L_d, yet are different from that language. For each, show that the language is not accepted by a Turing machine.

(a) L_{2d}, the set of all w_i so that w_i is not accepted by M_{2i}.

Suppose M_k accepts this language. Note that the encoding we have chosen for valid Turing machines, sequences of transitions coded as $0^i10^j10^k10^l10^m$ and separated by 11, but no 11 at the end, must always represent an even number. Thus $k = 2j$ for some j. So machine M_{2j} accepts L_{2d}. (If k were odd, then M_k does not represent a valid Turing machine).

Then w_j is either in L_{2d} or it isn't. Suppose that it is. Then M_{2j} does not accept string w_j. Similarly, if w_j is not in L_{2d}, then M_{2j} does not accept w_j, but for w_j to not be in L_{2d}, machine M_{2j} must accept w_j. $\rightarrow\leftarrow$ Thus M_{2j} does not accept L_{2d}. So L_{2d} is not accepted by any Turing machine.

(b) $L_{\frac{d}{2}}$, the set of all w_i so that w_{2i} is not accepted by M_i.

Suppose that M_k accepts $L_{\frac{d}{2}}$. Now w_{2k} is either in $L_{\frac{d}{2}}$ or it isn't. Suppose that it is. Then M_k does not accept string w_{2k}. But w_{2k} is in $L_{\frac{d}{2}}$ and thus should be accepted by M_k. Oops. Suppose that w_{2k} is not in $L_{\frac{d}{2}}$. Then M_k accepts w_{2k}. But M_k accepts $L_{\frac{d}{2}}$ and should not accept w_{2k}. $\rightarrow\leftarrow$ Thus M_k does not accept $L_{\frac{d}{2}}$. So $L_{\frac{d}{2}}$ is not accepted by any Turing machine.

Exercise 9.2.1

Show that the halting problem, the set of (M,w) pairs so that M halts when given input w is RE but not recursive.

The complement of this language is the set of pairs (M,w) so that M does not halt on string w. A machine that accepts this complement language would never halt on strings in the complement language, since it is simulating a machine which does not halt. So no machine can be built to accept the complement language. Since this complement language cannot be recursive, the original language cannot be either by theorem 9.3. □

Exercise 9.2.4

Let $L_1, L_2, \ldots, L_k$ be a collection of languages over alphabet Σ so that:

1. For all $i \neq j$, $L_i \cap L_j = \emptyset$.
2. $L_1 \cup L_2 \cup \ldots \cup L_k = \Sigma^*$.
3. Each of the languages L_i is recursively enumerable.

Show that each of the languages is recursive.

Proof. Suppose that one of the languages is not recursive, say, L_1 (clearly the choice of L_1 is unimportant). Then $L_2, \ldots, L_k$ are recursive. Then $\cup_{i=2}^{k} L_i$ is recursive since the union of recursive languages is recursive. But by property 2, $L_1 = \Sigma^* \setminus \cup_{i=2}^{k} L_i$. Then by theorem 9.3, L_1 is recursive. $\rightarrow\leftarrow$ Therefore each of the languages is recursive.

A similar argument applies if two of the languages are not recursive. Then the others are recursive and hence their union is recursive. Then the union of the non-recursive languages is the complement of a recursive language and is therefore recursive.

Similarly if $n < k$ of the languages is not recursive.

So all of the languages are recursive. □

Alternatively, to show that L_j is recursive, note that L_j is RE and $\overline{L_j} = \Sigma^* \setminus \cup_{i \neq j} L_i$ is also RE since the finite union of RE languages is RE. So L_j and $\overline{L_j}$ are both RE. By Theorem 9.4, L_j is recursive. □