CPSC 351 — Tutorial Exercise #9
Introduction to Turing Machines

This exercise is intended to help you to understand what happens when (reasonably simple) Turing machines are executed on input stings. It also includes a first "Turing machine design exercise" — which asks you to modify a Turing machine, that has already been studied, to solve a new problem that is related to the problem considered when the original Turing machine was introduced.

Getting Started

It is unlikely that this problem — which students should be able to complete — will be discussed in the tutorial for this exercise.

1. Let $\Sigma = \{0, 1\}$. The supplemental document for Lecture #10, "Turing Machine Design", includes the development of a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

such that that $\Gamma = \{0, 1, \mathtt{X}, \mathtt{Y}, \sqcup\}$, and the states and transitions of this Turing machine are as shown in Figure 1 on page 2.

In order to make the picture easier to read, the accept state is shown as "q_A" instead of "q_{accept}". All missing transitions are transitions to the reject state with the form "$\delta(q, \sigma) = (q_{\text{reject}}, \sigma, \mathtt{R})$" for some state $q \in Q$ and symbol $\sigma \in \Gamma$.

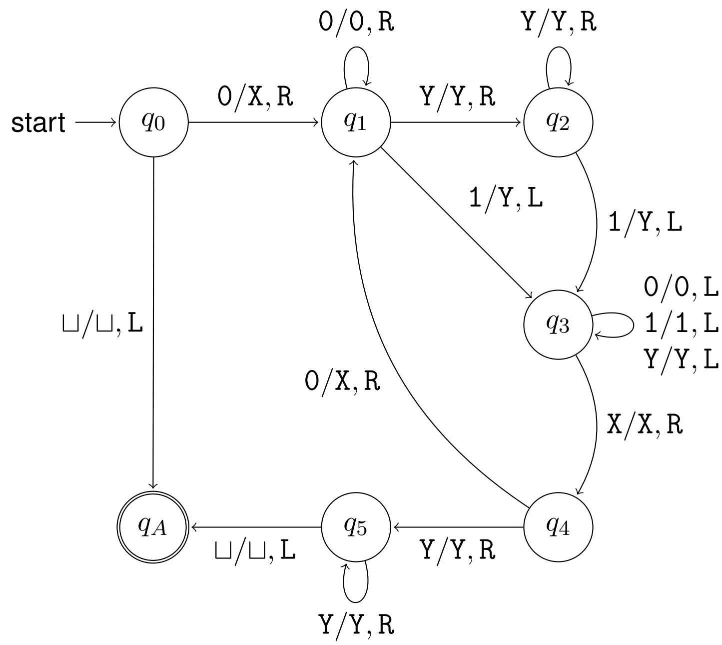

Figure 1: Turing Machine for the Language L

Show the sequence of transitions that are followed when this Turing machine is executed on each of the following input strings — and say whether each of these strings is in the language of this Turing machine.

(a) λ
(b) 0
(c) 1
(d) 01
(e) 10
(f) 001
(g) 011
(h) 0011
(i) 0101

Problems To Be Discussed in the Tutorial

Let $\Sigma_1 = \Sigma_2 = \{0, 1\}$. Recall that the ***unpadded binary representation*** of the number 0 is the string 0 with length one in $\Sigma_1^\star$ and that, if n is a positive integer, then the ***unpadded binary representation*** of n is the string

$$\sigma_k \sigma_{k-1} \ldots \sigma_1 \sigma_0 \in \Sigma_1^\star$$

with length k, for a non-negative integer k, such that $k \geq 0$, $\sigma_k = 1$, and (if we equate the symbol 1 with the number 1 and if we equate the symbol 0 with the number 0)

$$\sum_{h=0}^{k} \sigma_h \cdot 2^h = n.$$

Thus the unpadded binary representations of the numbers 1, 2, 3 and 4 are the strings 1, 10, 11, and 100, respectively.

Let $L_{\mathsf{bin}} \subseteq \Sigma_1^\star$ be the set of strings in $\Sigma_1^\star$ that are unpadded binary representations of non-negative integers .

Consider total function $f_{-1} : \Sigma_1^\star \to \Sigma_2^\star$ such that, for all $\omega \in \Sigma_1^\star$,

- If $\omega \in L_{\mathsf{bin}}$ and ω is the unpadded binary representation of a ***positive*** integer n, then $f_{-1}(\omega)$ is the unpadded binary representation of $n - 1$.
- If either $\omega = 0$ (so that $\omega \in L_{\mathsf{bin}}$ and ω is the unpadded binary representation of 0) or $\omega \notin L_{\mathsf{bin}}$ then $f_{-1}(\omega) = \lambda$.

2. Modify the ***high-level description*** of a Turing machine (that is, algorithm) that computes the function $f_{+1} : \Sigma_1^\star \to \Sigma_2^\star$, considered in the lecture presentation for Lecture #10, to obtain a high-level description of a Turing machine that computes the above function $f_{-1} : \Sigma_1^\star \to \Sigma_2^\star$, instead.

3. Use your high-level description to produce an ***implementation-level*** description of a Turing machine that computes the above function $f_{-1} : \Sigma_1^\star \to \Sigma_2^\star$ — explaining how the implementation-level corresponds to the high-level description, when this is not clear.

4. Use your implementation-level description to produce a ***formal description*** of a Turing machine that computes the above function $f_{-1} : \Sigma_1^\star \to \Sigma_2^\star$ — explaining how the formal-level description corresponds to the implementation-level description, when this is not clear.