Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Mohamad Ahmadi

Theoretical Computer Science - Bridging Course
Winter Term 2019/2020
Exercise Sheet 5

for getting feedback submit electronically by 12:15, Monday, November 25, 2019

Exercise 1: The Shift Operation *(4+4 Points)*

Consider a Turing machine $\mathcal{M}$ that is given an arbitrary input string over alphabet $\Sigma = \{1, 2, \ldots, n\}$ on its input tape. We would like $\mathcal{M}$ to insert an empty cell, i.e., $\sqcup$, at the beginning of the tape without removing any symbol on the tape. As an example, the Turing machine is supposed to change the input tape of the form $\langle 2, 4, 4, 6, 1, 8, 4, \sqcup, \sqcup, \ldots \rangle$ to $\langle \sqcup, 2, 4, 4, 6, 1, 8, 4, \sqcup, \sqcup, \ldots \rangle$. Although this operation is not explicitly defined for a Turing machine, one can consider such an operation as shifting the whole string one cell to the right on the input tape.

(a) Give a formal definition of $\mathcal{M}$ to perform the desired operation such that $\mathcal{M}$ recognizes the language Σ^*.

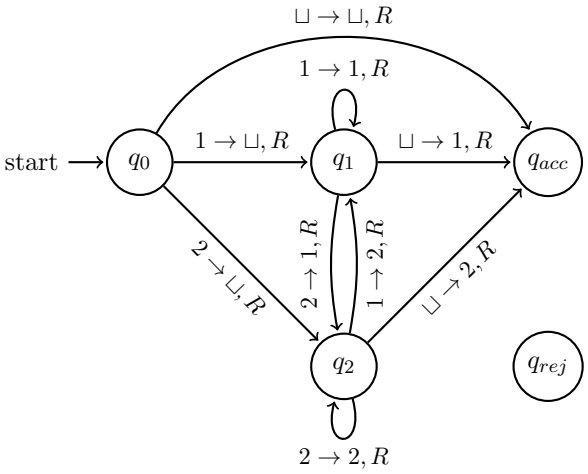
(b) For $n = 2$, i.e., $\Sigma = \{1, 2\}$, draw the state diagram of your constructed Turing machine.

Sample Solution

(a) Consider the set of states to be $Q = \{q_0, q_{acc}, q_{rej}\} \cup \{q_c | c \in \Sigma\}$. Then, the transition functions are as follows.

$$\forall c \in \Sigma;\ \delta(q_0, c) = (q_c, \sqcup, R)$$
$$\forall c, c' \in \Sigma;\ \delta(q_c, c') = (q_{c'}, c, R)$$
$$\forall c \in \Sigma;\ \delta(q_c, \sqcup) = (q_{acc}, c, R)$$
$$\delta(q_0, \sqcup) = (q_{acc}, \sqcup, R)$$

(b)



Exercise 2: Constructing Turing Machines I *(4+1+2+1 Points)*

Let $\Sigma = \{0, 1\}$. For a string $s = s_1 s_2 \dots s_n$ with $s_i \in \Sigma$ let $s^R = s_n s_{n-1} \dots s_1$ be the *reversed* string. *Palindromes* are strings s for which $s = s^R$. Then $L = \{sas^R \mid s \in \Sigma^*, a \in \Sigma \cup \{\varepsilon\}\}$ is the language of all palindromes over Σ.

(a) Give a state diagram of a Turing machine recognizing L.

(b) Give the maximum number (or a close upper bound for the number) of head movements your Turing machine makes until it halts, if started with an input string $s \in \Sigma^*$ of length $|s| = n$ on its tape.

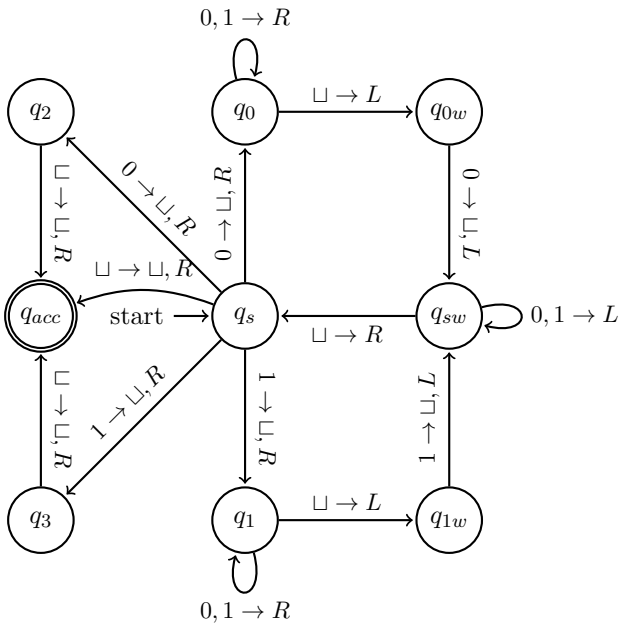
(c) Describe (informally) the behavior of a 2-tape Turing machine which recognizes L and uses significantly fewer head movements on long inputs than your 1-tape Turing machine.

(d) Give the maximum number (or a close upper bound for the number) of head movements your Turing machine makes on any of the two tapes until it halts, if started with an input string $s \in \Sigma^*$ of length $|s| = n$ on the first tape.

Sample Solution

We describe both Turing machines by their behaviour.

(a)



(b) The head of the Turing machine will move $(n+1) + n + (n-1) + \ldots + 1 \leq n^2$ steps and does not need any additional space besides the input word.

(c) Move the tape head of the input tape to the end, and then read backwards to the start of the input tape. As you go, write the tape symbols in order on the second tape so that you end up with the reverse of the input tape on the second tape. Reset both tape heads, and then move each to the end of the tape, at each step comparing the symbols each head is pointing to. If you find a position where the symbols are different, the input is not a palindrome and you halt-reject. If you get to the end (first blank symbol on the end) without finding a mismatch then it is a palindrome and you halt-accept.

(d) The heads move three times through the input on both tapes, leading to $O(n)$ head moves in total.

Exercise 3: Constructing Turing Machines II *(4 Points)*

Let $L = \{\langle w \rangle, \langle w+1 \rangle \mid w \in \mathbb{N}\}$, e.g., the word $\langle 6 \rangle, \langle 7 \rangle = 110, 111$ is contained in L. Design a Turing machine which accepts L. You do not need to provide a formal description of the Turing machine but your description has to be detailed enough to explain every possible step of a computation.
Remark: Here $\langle w \rangle$ *is the binary encoding of the number* w*, e.g., the number* 6 *is going to be the string* 110*.*

Sample Solution

We first check whether the string is in the format $\{0,1\}^*, \{0,1\}^*$. Then we add 1 to the left hand string and then we simply compare the left and right string symbol by symbol and accept if they are equal, otherwise we reject.
The tricky part is transforming the left string such that it is the binary encoding of $w+1$.
For that we move to the very right of it; assume that after this the TM is in some state q_1. The meaning of q_1 is that we need to add one to the number which is encoded by the string which starts at the very left and reaches until the current head position.
If we read a 0 in q_1 we simply change it to a 1. In this case we are done and can move to the comparing step. However, if we read a 1 we cannot add 1 to the respective position. Just as when doing *written addition* we thus have to carry the addition over to the next position. That is, we change the current symbol on the tape from 1 to 0, do not change the state and move the read/write head one position to the left. Whenever we reach a 0 in this process we'll be done. But when adding 1 to a string like $s = 11111$ we will never reach a zero, but instead we reach the left end of the string. Adding 1 to s results in 100000, i.e., if we reach the blank symbol in state q_1 we also need to substitute it with a 1 and we can continue with the comparing step.
In the comparison step we always overwrite the symbols which we have already compared with a new symbol Z to find the corresponding position when comparing the next two symbols.