Theoretical Computer Science (Bridging Course)

Dr. G. D. Tipaldi
F. Boniardi
Winter semester 2014/2015

University of Freiburg
Department of Computer Science

Exercise Sheet 6

Due: 11th December 2014

Exercise 6.1 (Turing Machines)

(a) Design a Turing Machine that decides the language $L := \{0^n 1^n \mid n \geq 1\}$. Explain your choice.

Solution: Alternately, the TM will change a 0 to an X and then a 1 to a Y until all $0'$s and $1'$s have been matched.

In more detail, starting at the left end of the input, the TM enters a loop in which it changes a 0 to a X and moves to the right over whatever $0'$s and Y's it sees, until it comes to a 1. It changes the 1 to a Y and moves left, over Y's and $0'$s, until it finds an X. At that point, it looks for a 0 immediately to the right, and if it finds one, changes it to X and repeats this process, changing a matching 1 to a Y.

The formal specification is $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, q_{reject}, q_{accept})$, where:

- $Q := \{q_0, q_1, q_2, q_3, q_r, q_a\}$.
- $\Sigma = \{0, 1\}$.
- $\Gamma = \{0, 1, X, Y, \sqcup\}$
- The transition function δ is given by

$Q \times \Gamma$	0	1	X	Y	$\sqcup$
q_0	(q_1, X, R)	$(q_r, 1, R)$	(q_r, X, R)	(q_3, Y, R)	$(q_r, \sqcup, R)$
q_1	$(q_1, 0, R)$	(q_2, Y, L)	(q_r, X, R)	(q_1, Y, R)	$(q_r, \sqcup, R)$
q_2	$(q_2, 0, L)$	$(q_r, 1, R)$	(q_0, X, R)	(q_2, Y, L)	$(q_r, \sqcup, R)$
q_3	$(q_r, 0, R)$	$(q_r, 1, R)$	(q_r, X, R)	(q_3, Y, R)	$(q_a, \sqcup, R)$
q_r	-	-	-	-	-
q_a	-	-	-	-	-

- $q_{reject} := q_r$.
- $q_{accept} := q_a$.

(b) Give the sequence of configurations for the input string 0011.

Solution:

$$
\begin{aligned}
&q_0 0011 \rightsquigarrow X q_1 011 \rightsquigarrow X 0 q_1 11 \rightsquigarrow \\
&\rightsquigarrow X q_2 0 Y 1 \rightsquigarrow q_2 X 0 Y 1 \rightsquigarrow X q_0 0 Y 1 \rightsquigarrow \\
&\rightsquigarrow X X q_1 Y 1 \rightsquigarrow X X Y q_1 1 \rightsquigarrow X X q_2 Y Y \rightsquigarrow \\
&\rightsquigarrow X q_2 X Y Y \rightsquigarrow X X q_0 Y Y \rightsquigarrow X X Y q_3 Y \rightsquigarrow \\
&\rightsquigarrow X X Y Y q_3 \sqcup \rightsquigarrow X X Y Y \sqcup q_a \sqcup
\end{aligned}
$$

(c) Give the sequence of configurations for the input string 0010.

Solution:

$$
\begin{aligned}
&q_0 0010 \rightsquigarrow X q_1 010 \rightsquigarrow X 0 q_1 10 \rightsquigarrow \\
&\rightsquigarrow X q_2 0 Y 0 \rightsquigarrow q_2 X 0 Y 0 \rightsquigarrow X q_0 0 Y 0 \rightsquigarrow \\
&\rightsquigarrow X X q_1 Y 0 \rightsquigarrow X X Y q_1 0 \rightsquigarrow X X Y 0 q_1 \sqcup \rightsquigarrow \\
&\rightsquigarrow X X Y 0 \sqcup q_r \sqcup
\end{aligned}
$$

Exercise 6.2 (Turing Machines)
Describe a TM that decides the language

$$L = \{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}\,.$$

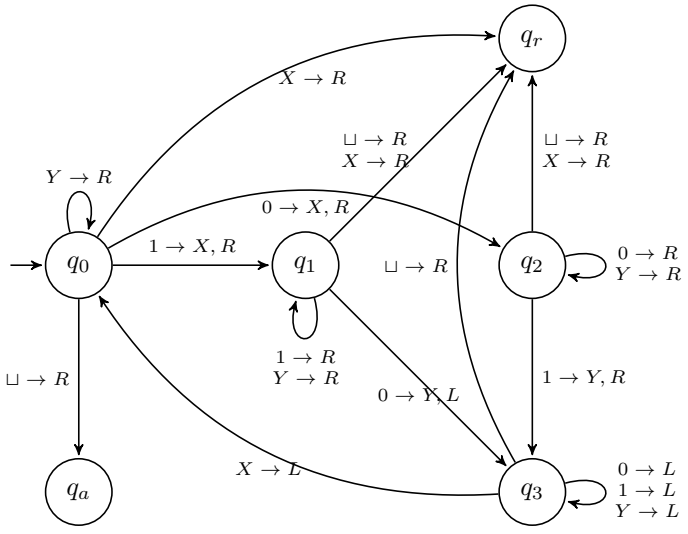
where $|w|_0$ and $|w|_1$ are respectively the number of 0's and 1's in w.

Solution: We call $\mathcal{M}$ such Turing Machine. $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ can be the following:

- $Q := \{q_0, q_1, q_2, q_3, q_a, q_r\}$.
- $\Sigma := \{0, 1\}$.
- $\Gamma := \{0, 1, X, \sqcup\}$.
- $q_{accept} := q_a$.
- $q_{reject} := q_r$.
- The transition function δ is given by

State	0	1	$\sqcup$	X	Y
q_0	(q_2, X, R)	(q_1, X, R)	$(q_a, \sqcup, R)$	(q_r, X, R)	(q_0, Y, R)
q_1	(q_3, Y, L)	$(q_1, 1, R)$	$(q_r, \sqcup, R)$	(q_r, X, R)	(q_1, Y, R)
q_2	$(q_2, 0, R)$	(q_3, Y, L)	$(q_r, \sqcup, R)$	(q_r, X, R)	(q_2, Y, R)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	$(q_r, \sqcup, R)$	(q_0, X, R)	(q_3, Y, L)
q_r	-	-	-	-	-
q_a	-	-	-	-	-

The state diagram of $\mathcal{M}$ is reported below.



Exercise 6.3 (Pushdown Automata, Turing Machines)
How would one simulate a PDA on a Turing machine? Please do not write the Turing machine itself, but rather write the key idea in plain English.

Solution: We show how a PDA can be simulated by a non-deterministic two-tape TM. This would in a second step be transformed into a normal (deterministic, one-tape) TM using the transformations shown in the lecture.
We use the second tape of the TM to represent the stack of the PDA, whereas the first tape is only used to read the input of the PDA. The states of the TM basically represent the states of the PDA. In addition, we need some auxiliary states (see below). The TM works as follows: Whenever the PDA would read a symbol, the TM must read the symbol under the head on the first tape and moves the head one step to the right. Whenever the PDA uses the empty symbol as input symbol, the head on the first tape keeps its position. At the same time, the TM performs the appropriate action on the second tape in such a way the the head is always positioned on the "top-most" symbol:

- If the PDA consumes a symbol from the stack but does not write a new one, the TM deletes the symbol under the head and moves the head one position to the left.
- If the PDA consumes a symbol from the stack and writes a new one, the TM overwrites the symbol below the head with the new one.
- If the PDA does not consume a symbol from the stack but writes a new one, the TM moves one step to the right and writes the new symbol. For this we actually need to perform two steps which can be done using some auxiliary states (we need to remember the symbol to write and the current state of the TM).
- If the PDA does not change the stack, the TM does not change the second tape.

This TM accepts the input of the PDA iff it stops in an accept state (if there are several accepting states in the PDA, we use an additional transition that brings the TM from these states to its single accept state).

Exercise 6.4 (Non Deterministic Turing Machines)

We call a natural number *composite* if it is not prime[1], formally, the set of natural composite numbers is

$$\{hk \mid h, k \in \mathbb{N},\ h, k \geq 2\}$$

Give a nondeterministic Turing machine of the alphabet of vertical bars $\Sigma = \{|\}$ that recognizes the language of composite numbers encoded as unary numbers (i.e. a natural number n is encoded in the form $|^n$). You should not give a formal construction, but describe the idea behind it as precise as possible.

Solution: Using a nondeterministic Turing machine "recognizing" composite numbers is not that hard. We can use the non-determinism to guess. The following is the idea behind the construction of a non-deterministic Turing Machine τ that recognizes composite numbers in unary form, i.e., $|^n$ for some $n \in \mathbb{N}$.

1. Nondeterministically choose two numbers p and q, such that $p \geq 1, q \geq 1$ and transform the input in the form $\sqcup|^n \sqcup |^q \sqcup |^q \sqcup$
2. Multiply p by q to obtain $\sqcup|^n \sqcup |^{pq} \sqcup$
3. Check the numbers of $|$ before the middle $\sqcup$ and then accept if this number is equal, and reject otherwise.

Since the Turing Machine is non-deterministic, we are sure on the step 2 of the algorithm, the given number n will be tested against all its potential factors and hence we will know with probability 1 whether the number is composite or not.

[1] Remember that 1 and -1 are <u>not</u> prime numbers! By definition, prime numbers must not be invertible in $\mathbb{Z}$.