MTAT.05.125 Introduction to Theoretical Computer Science

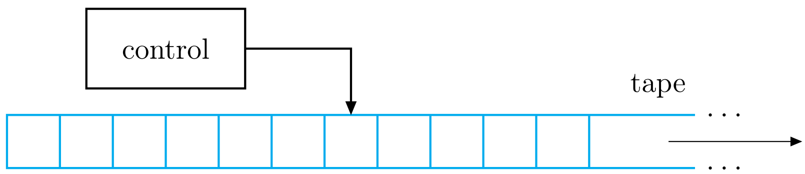
Autumn 2015

Vitaly Skachek
Estonian version by Reimo Palm
English version by Yauhen Yakimenka

Lecture 8. Turing machines.

Finite automata are a very restricted model (mostly because of their finite memory). We are interested in a computational model that is more similar to modern computers.[1]

Idea: add infinite memory to a finite automaton. The memory is implemented as infinite tape.



This computational model is called Turing machine. Turing machine:

- writes and reads information to/from the tape;
- the read&write head can move both to the left and to the right;
- the tape is (one-side) infinite;
- the machine has a state that is not a part of the tape;
- there are special states "accept" and "reject". Once the machine enters one of these states, it accepts or rejects, respectively.

[1]Note however that historically Turing machine was prior to the computers. In some sense it was a theoretical foundation of computers.

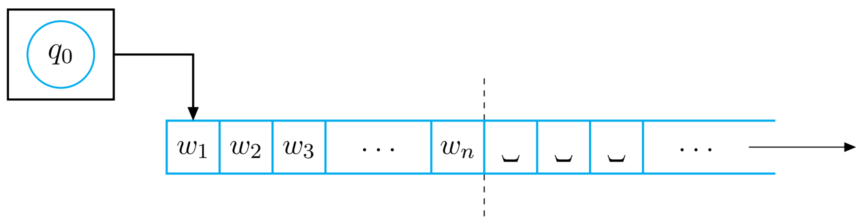
Definition. *Turing machine is a 7-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$*, where*

- Q *is a finite set of states;*
- Σ *is finite input alphabet, blank symbol '*$\sqcup$*'* $\notin \Sigma$*;*
- $\Gamma \supseteq \Sigma$ *is finite tape alphabet, blank symbol '*$\sqcup$*'* $\in \Gamma$*;*
- $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ *is the transition function;*
- $q_0 \in Q$ *is start state;*
- $q_{acc} \in Q$ *is accept state;*
- $q_{rej} \in Q$ *is reject state,* $q_{rej} \neq q_{acc}$*.*

Notes:

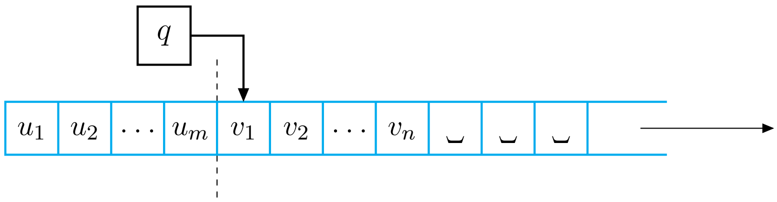
- Blank symbol '$\sqcup$' denotes the end of the input on the tape.
- Turing machine operates as follows. If it is in the state q_i, head is over the symbol $c \in \Gamma$ on the tape (whatever coordinate of this position on the tape is) and $\delta(q, c) = (q_j, d, dir)$, then the machine writes $d \in \Gamma$ into cell under the head, moves according to dir (i.e. one cell to the left or to the right) and changes its state to q_j.
- If the machine tries to move its head to the left of the left-most cell, the head stays at the same place.

Initial configuration of the Turing machine:



- head is in the left-most position;
- machine is in the state q_0;
- blank symbols fill the tape from the end of input and until infinity.

At every moment configuration of the machine could be described as (current state q, tape contents, head location). We denote this as uqv, where q is current state, u is a string written on the tape to the left from current position and v is a string written on the tape from current position to the end of the contents of the tape. For instance, the configuration



is denoted as $u_1u_2 \dots u_m q v_1 v_2 \dots v_n$.

We say that configuration C_1 *yields* configuration C_2 if the Turing machine can legally move from C_1 to C_2 in one step.

Formally, for leftward move uaq_ibv yields uq_jacv if and only if

$$\delta(q_i, b) = (q_j, c, \mathsf{L})\,;$$

and for rightward move uaq_ibv yields $uacq_jv$ if and only if

$$\delta(q_i, b) = (q_j, c, \mathsf{R})\,.$$

Accept configuration: the state is q_{acc}.
Reject configuration: the state is q_{rej}.

Accept and reject configurations are halting configurations, i.e. after them Turing machine stops execution.

Definition. *The set of strings that the Turing machine M accepts is* the language of M, *or the language recognised by M. A language is* Turing-recognisable *if there is a Turing machine that recognises this language.*

Note. It is important to highlight that if TM M recognises a language L then for any string $w \notin L$, M either rejects input w or M never halts on input w. In other words, M never "lies": it either says a correct statement about input ("$w \in L$" or "$w \notin L$) or gives no answer.

Contrary to finite automata, Turing machine can potentially never stop, i.e. not reach halting state in finite time. This is reflected in the following definition.

Definition. *Turing machine* decides *the language if it always arrives to either q_{acc} or q_{rej} and it recognises that language. The language is* Turing-decidable *if there is a Turing machine that decides the language.*

In other words, Turing machine M decides language L if and only if on *any* finite input string w the machine can tell in finite time if w belongs to L or not.

If M decides language L then M also recognises it. The opposite is not always true.

Example 1. Let

$$L = \{w\#w \mid w \in \Sigma^*,\ \Sigma = 0, 1\}.$$

Design Turing machine M that decides L.

On input string s the machine M does the following.

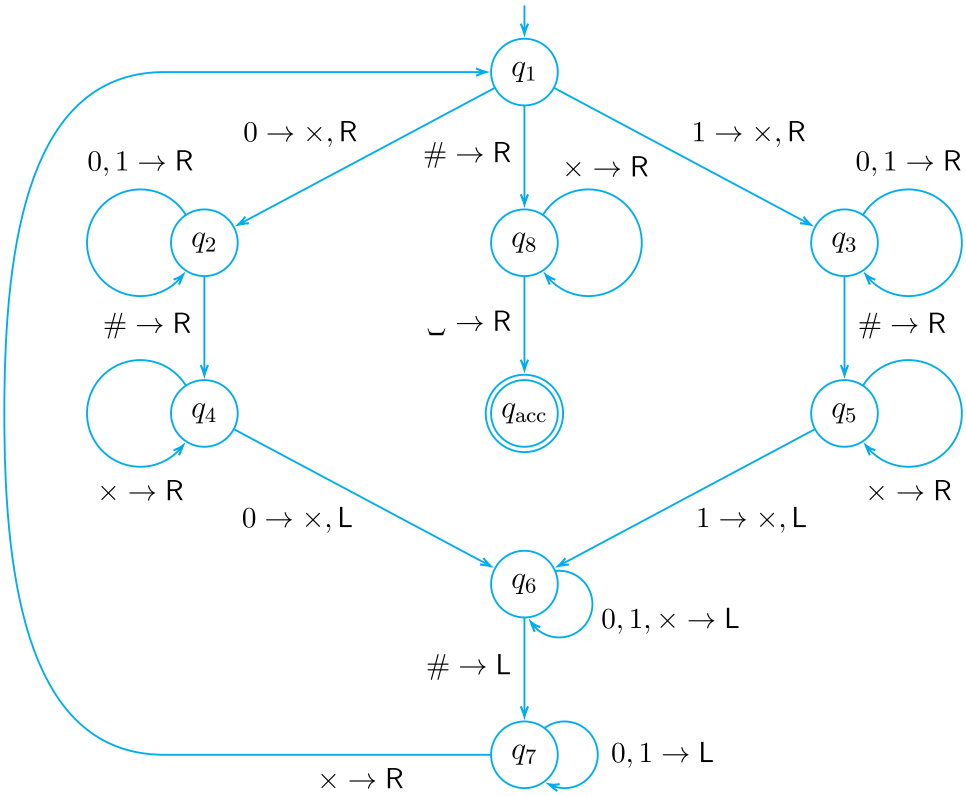
1. Move across the tape to corresponding cells on either side of the $\#$ symbol to check if these cells contain the same symbol. If they do not, or if no $\#$ symbol is found, reject. Mark all the symbols that were checked to keep track of the corresponding symbols.

2. When all the cells to the left from $\#$ were marked, check for any remaining unmarked cells to the right from $\#$. If any unmarked cells remain, reject. Otherwise accept.

Let us see how the machine works on input 0110#0110.

$$\begin{array}{ccccccccccc}
0 & 1 & 1 & 0 & \# & 0 & 1 & 1 & 0 & \llcorner\!\lrcorner & \ldots \\
\times & 1 & 1 & 0 & \# & 0 & 1 & 1 & 0 & \llcorner\!\lrcorner & \ldots \\
\times & 1 & 1 & 0 & \# & \times & 1 & 1 & 0 & \llcorner\!\lrcorner & \ldots \\
\times & \times & 1 & 0 & \# & \times & 1 & 1 & 0 & \llcorner\!\lrcorner & \ldots \\
\times & \times & 1 & 0 & \# & \times & \times & 1 & 0 & \llcorner\!\lrcorner & \ldots \\
 & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \\
\times & \times & \times & \times & \# & \times & \times & \times & 0 & \llcorner\!\lrcorner & \ldots \\
\times & \times & \times & \times & \# & \times & \times & \times & \times & \llcorner\!\lrcorner & \ldots \\
\times & \times & \times & \times & \# & \times & \times & \times & \times & \llcorner\!\lrcorner & \ldots \\
\times & \times & \times & \times & \# & \times & \times & \times & \times & \llcorner\!\lrcorner & \ldots \\
 & & & & & & & & & \text{accepts} &
\end{array}$$

We can describe Turing machines in complete details by giving δ-functions for all possible configurations. See the state diagram:



Here $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \times, \llcorner\!\lrcorner\}$. The machine moves to q_{rej} if there is no input consistent with the diagram.

Practise session

1. Show that the language $L = \{1^{n^2} \mid n \geq 0\}$ is not regular.

Solution. Assume, to the contrary, that L is regular and p is its "pumping length". Take $w = 1^{p^2} \in L$; $|w| = p^2 \geq p$. We could represent w as $w = xyz$, where for all $i \geq 0$ it holds that $xy^iz \in L$.

Consider the string xy^2z. As $|y| \leq |xy| \leq p$, then $|xy^2z| = |xyz| + |y| \leq p^2 + p$.

On the other hand, since $|y| > 0$, we have $|xy^2z| = |xyz| + |y| > |xyz| = p^2$. We have:

$$p^2 < |xy^2z| \leq p^2 + p = p(p+1) < (p+1)^2,$$

which means that the length of xy^2z is strictly between two squares of consecutive integers and therefore it cannot be a square of any integer. Thus, $xy^2z \notin L$. Contradiction! Therefore L is not regular.

2. Take $L = \{0^n 1^m \mid n > m\}$. Prove that L is not regular.

Solution. We repeat the same proof scheme:

- Assume that L is regular and p is its "pumping length".
- Take $w = 0^{p+1}1^p \in L$. Clearly, $|w| \geq p$.
- We can apply the pumping lemma. I.e. there exist x, y, z, such that $w = xyz$ and the conditions of the pumping lemma are satisfied.
- From the pumping lemma's condition 1, for all $i \geq 0$ we have that $xy^iz \in L$. In particular, take $i = 0$. Therefore xz should be in L too.[2]
- Since $|xy| \leq p$, y consists of zeros only. Therefore removing y from xyz decreases the number of zeros by at least one and xz will have not more than p zeros. This contradicts the definition of L and, hence, $xz \notin L$.
- We end up with contradiction. This means that original assumption was wrong and L is not a regular language.

3. Show that $L = \{1^{2^n} \mid n \geq 0\}$ is not regular. (It a set of all strings of ones of length 2^n for $n \geq 0$.)

Solution. We will use a pumping lemma (see previous lecture for details).

Assume that L is regular and p is its pumping length give by the pumping lemma. Choose $w = 1^{2^p}$. Clearly, $w \in L$ and $|w| \geq p$. Therefore $w = xyz$, where $|xy| \leq p$, $|y| > 0$ and for all $i \geq 0$ we have $xy^iz \in L$.

Since $p < 2^p$ for any $p \geq 0$, so $|y| < 2^p$. Thus $|xyyz| = |xyz| + |y| < 2^p + 2^p = 2^{p+1}$. That is why

$$2^p < |xyyz| < 2^{p+1}$$

and the length of the word $xyyz$ is not a power of 2. It means that $xyyz \notin L$. Contradiction. Therefore L is not regular.

4. Let $C_5 = \{x \mid x \text{ is a binary number that is multiple of 5 }\}$. Show that C_5 is regular.

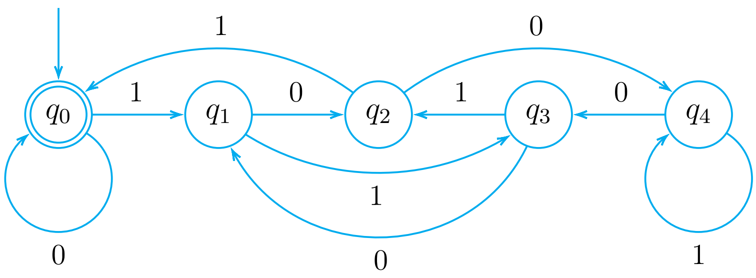
Solution. We construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognises C_5. Let $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$, start state be q_0 and transition

[2] But in the next steps we will show the opposite.

function be

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

State diagram of this automaton is as follows



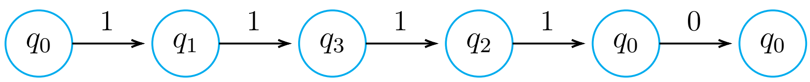
The state of the automaton stores the reminder of currently read input divided by 5: states $q_0, \ldots, q_4$ correspond to reminders $0, \ldots, 4$, respectively (so q_0, i.e. reminder 0, is accept state).

If the number that we read so far is x with remainder $x \mod 5 = r$ and we read one more digit:

read	number	remainder	remainders (respectively)
0	$x \mapsto 2x$	$r \mapsto 2r \mod 5$	$(0,1,2,3,4) \mapsto (0,2,4,1,3)$
1	$x \mapsto 2x+1$	$r \mapsto (2r+1) \mod 5$	$(0,1,2,3,4) \mapsto (1,3,0,2,4)$

According to respective change of remainders we build the transition function. For example, if we have a number with binary representation x with remainder[3] 3 and we read 1, then the new number will have binary representation $x1$ and remainder $(3 \cdot 2 + 1) \mod 5 = 7 \mod 5 = 2$; hence we put arrow $q_3 \to q_2$ with label “1”.

Let us see for instance how the automaton works on input 11110 (i.e. binary representation of 30):



input		1	11	111	1111	11110
value			$= 3$	$= 7$	$= 15$	$= 30$

[3]Note that exact x is not important; it is only a remainder that matters.

5. Are the following statements true or false?

(a) If $L_1 \cup L_2$ is regular and L_1 is finite, then L_2 is regular.

(b) If $L_1 \cup L_2$ is regular and L_1 is regular, then L_2 is regular.

(c) If $L_1 L_2$ is regular and L_1 is regular, then L_2 is regular.

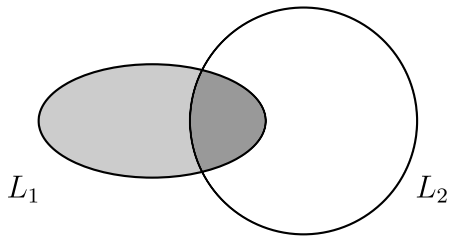
(d) If L^* is regular then L is regular.

Solution.

(a) True. Note that

$$L_2 = (L_1 \cup L_2) \cap (L_1 \setminus L_2)^c,$$

where ^{c} stand for complementary language. $L_1 \cup L_2$ is regular (given), $L_1 \setminus L_2$ is regular (every finite language is regular) and $(L_1 \setminus L_2)^c$ is regular as complementary to regular[4]. Therefore L_2 is an intersection of two regular languages and thus regular itself.



(b) False. Consider $L_1 = \Sigma^*$ and L_2 being any nonregular language. Then $L_1 \cup L_2 = \Sigma^*$ is regular but L_2 is not.

(c) False. Let $L_1 = \{\varepsilon, 0\}$. This is a finite language and thus regular. Let $L_2 = (00)^* \cup \{0^{n^2} \mid n \geq 0,\ n \text{ is prime}\}$. It could be shown this language is not regular. However $L_1 L_2 = 0^*$ is regular.

(d) False. Let $L = \{0^n 1^n \mid n \geq 0\} \cup \{0, 1\}$. This language is not regular (could be proven analogously to example 1 in lecture 8). But $L^* = \Sigma^*$ is regular.

[4]To see that complementary to a regular language L is also regular, we note that from DFA for L we build DFA for L^c by just making all accept states be non-accept, and vice versa.