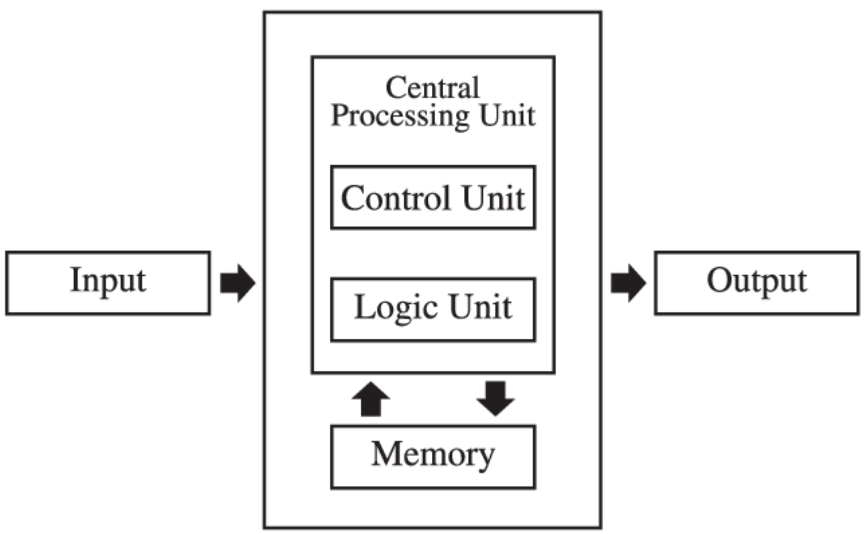
Automata, Computability & Complexity

Author: Daria Shutina

Automata, Computability & Complexity
- 23-02-06
  - von Neumann architecture
  - Finite automata
    - Definitions
    - Example 1: a STD
    - Example 2: finding FA /todo
    - Regular language
    - Regular operations
  - Th (unoin operation of RLs)
  - Th. (concatenation operation of RLs)

23-02-06

von Neumann architecture



Our nowadays computers are based on the von Neumann computer architecture.

Finite automata

For theory achievements, we will use a simpler compute model which gets the input as a string of symbols, has no memory, and generates an output that is either `accept` or `reject`. This machine is called *finite automaton.*

Definitions

Definition 1.1 (Finite automaton) A **finite automaton** (FA) M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma \to Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states / final states**.

A transition function can be described via *state transition diagram* (STD) or *state transition table*.

If a state is a *start state*, it has an arrow pointing from nowhere.

What is an *accept state*? Imagine we have an input. We go from one state to another, as the input says. Then we finish in a definite state. This state can be a final/accept state. In other words, it is where an input ends. Final states are shown as double-circled states in STD.

FA accepts a string if it starts in a start state, uses inly iterations of δ and finishes in one of final states.

Definition 1.2 (Strings accepted by M) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and $w = w_1 w_2 \cdots w_n$ be a string over alphabet Σ.
M **accepts** w if there exists a sequence of states $r_0, r_1, \ldots, r_n$, such that all following three conditions hold:

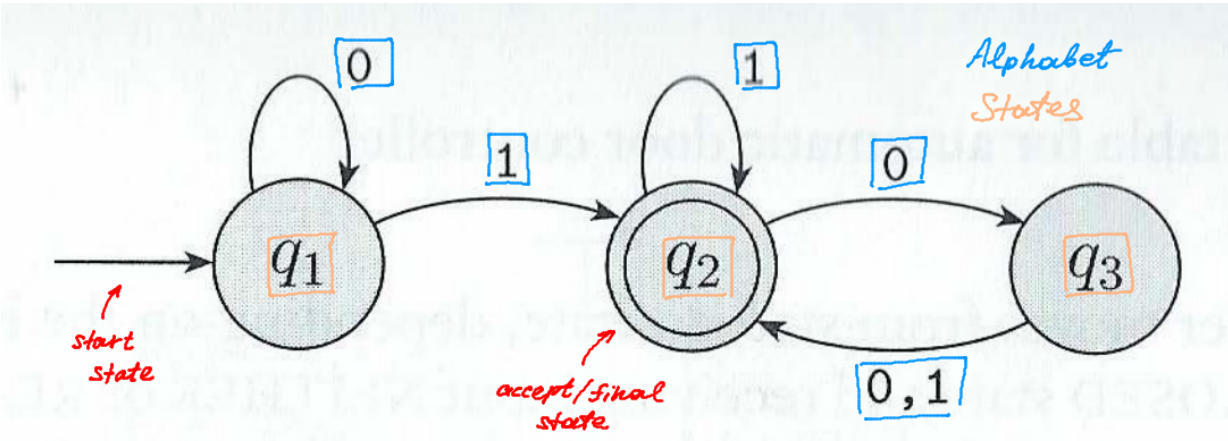
1. $r_0 = q_0$ *(M starts in start state.)*
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots n-1$ *(State change follows transition function.)*
3. $r_n \in F$ *(M ends up in accept state)*

If M does not accept w, it **rejects** it.

A *computation of FA on a string* is a sequence of states such that it starts in a start state and uses only iterations of δ.

$L(M)$ -- *the language of machine M* -- is the set of all strings that are accepted by M. Every FA stil recognizes and empty language $\varnothing$.

Example 1: a STD



$Q = \{q_1, q_2, q_3\}$; $\Sigma = \{0, 1\}$; $F = \{q_2\}$.

q_1 is a start state.

δ can be described with a table:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

FA accepts a string 1101, for example, since it starts in q_1 and finishes in q_2.

Example 2: finding FA /todo

We consider a language $L = \{w \mid w\ contains\ 001\ as\ substring\}$. The alphabet is $\{0, 1\}$.

The idea is that we will have 4 states:

- q -- no subsequence
- q_0 -- we have 0
- q_{00} -- we have 00
- q_{001} -- we have 001 and we neep to stop.

/todo нужна картинка

Regular language

A language is called a *regular language* if $\exists$ FA that recognizes it.

To prove that a language is regular, we need to build a FA that will recognize it. If we are able to build a STD, then the language is regular.

Example of a non-regular lenguage:

$L = \{0^n 1^n \mid n \geqslant 1\}$.

n is not fixed, so we have a problem with choosing a transition function.

Regular operations

Definition 1.6 (Regular operations) Let A and B be languages. We define the regular operations **union**, **concatenation** and **star** as follows:

- **Union**: $A \cup B := \{x | x \in A \text{ or } x \in B\}$
- **Concatenation**: $A \circ B := \{xy | x \in A \text{ and } y \in B\}$.
- **Star**: $A^* := \{x_1 x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$.

Th (unoin operation of RLs)

The class of regular languages is closed under the union operation:

A_1, A_2 are RLs $\Rightarrow A_1 \cup A_2$ is a RL.

Proof:

There are regular languages A_1 and A_2. W

There exists FAs M_1 and M_2 such that

$$\begin{cases} L(M_1) = A_1,\ M_1 = (S, \Sigma, \delta_1, s_0, F_1) \\ L(M_2) = A_2,\ M_2 = (T, \Sigma, \delta_2, t_0, F_2) \end{cases}$$

If M_1 and M_2 have different alphabets, then Σ will be the union of their alphabets.

To show that $A_1 \cup A_2$ is a RL, we construct a FA M such that $M = (Q, \Sigma, \delta, q_0, F)$

$Q = S \times T$

Σ is the same

$\delta((s,t),a) = (\delta_s(s,a), \delta_t(t,a))$

$q_0 = (s_0, t_0)$

$F = F_1 \times F_2$

Th. (concatenation operation of RLs)

The class of regular languages is closed under the concatenation operation.

A_1, A_2 are RLs $\Rightarrow$ $A_1 \circ A_2$ is a RL.

Proof: его нет