Regular Languages

Textbook: 1.1-1.4

Should attempt the exercises at the end of the chapter, and can discuss them on the discussion board.

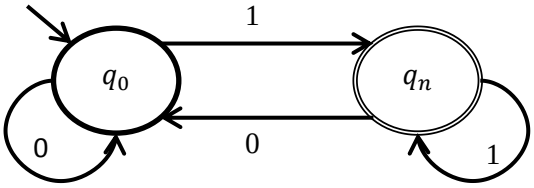
DFA – deterministic finite automata

Mathematical definition:

$M = (Q, \Sigma, \delta, q_0, F)$ is a DFA where:

- Q: a finite set of states
- Σ: a finite alphabet
- $\delta: Q \times \Sigma \to Q$- a transition function
- $q_0 \in Q$: an initial state
- $F \subset Q$: a set of accepting states

Example:



The automata M **accepts a language** L, written:

$$L(M) = \{s \in \{0,1\}^* | \; s \; ends \; with \; 1\}$$

The $*$ is called the **Kleene-star** operation and Σ^* is the set of all strings in the alphabet Σ, including the empty string ϵ:

$$\{0,1\}^* = \{\epsilon, 0,1,00,01,10,11,000 \dots\}$$

Note that we can enumerate Σ^*.

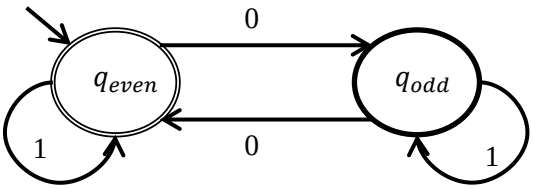
The transition function δ can be written as a table of $Q \times \Sigma \to Q$. In this case:

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

Exercise:

Create a DFA that accepts all strings with an even number of 0's:

$L(M) = \{s \in \{0,1\}^* | \; the \; number \; of \; 0's \; in \; s \; is \; even\}$



The intuition is that we don't know a long is the string going to be, so we need some way of keeping track of the parity of the 0's. at the point we see one 0, we need to know how many did we already see. Since we have a finite number of states, we cannot hold a state for each 0 (first 0, second 0 etc.). the representation above is kind of like that, modulo 2.

Definition:

Let $w \in \Sigma^*$ and M a DFA over Σ, then M **accepts** w if $w = w_1 \dots w_n$, $w_i \in \Sigma$, $1 \leq i \leq n$ and there is a sequence $q_0, q_1, \dots, q_n \in Q$ (where q_0 is the start state) and $\delta(q_i, w_{i+1}) = q_{i+1}$ for all $0 \leq i \leq n-1$, and $q_n \in F$.
$w \in L(M)$ is a **decidable** question, i.e. it can be checked in a finite time.

Definition:

Let M be a DFA, then the **language recognized by M** is $L(M) = \{w \in \Sigma^* | \ M \ accepts \ w\}$.

Definition:

Let Σ be a finite alphabet, and let $L \subset \Sigma^*$, then L is a **regular language** if there is a DFA M over Σ s.t. $L = L(M)$.

Regulatr Operations:

Let $A, B \subset \Sigma^*$. We can consider different operations, such as:

- Union: $A \cup B$
- Concatenation: $A \circ B = \{ab \mid a \in A, b \in B\}$
- Star: $A^* = \{a_1 a_2 \dots a_k | \ k \geq 0, a_i \in A \ for \ all \ 1 \leq i \leq k\}$
  For instance: $\Sigma = \{0,1\}, A = \{00,01\} \Rightarrow A^* = \{\epsilon, 00, 01, 0000, 0001, 0100, 0101, \dots\}$

If A, B are regular languages, are the languages described above also regular languages?

Closure under union:

Let $A, B \subset \Sigma^*$ be regular languages. Is $A \cup B$ a regular language?

By definition we have DFAs M_A, M_B such that $A = L(M_A), B = L(M_B)$. We will construct a DFA $M_{A \cup B}$ as follows:

We need to keep track along the way on both what M_A and M_B would do. Formally:

Let $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A), M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$.

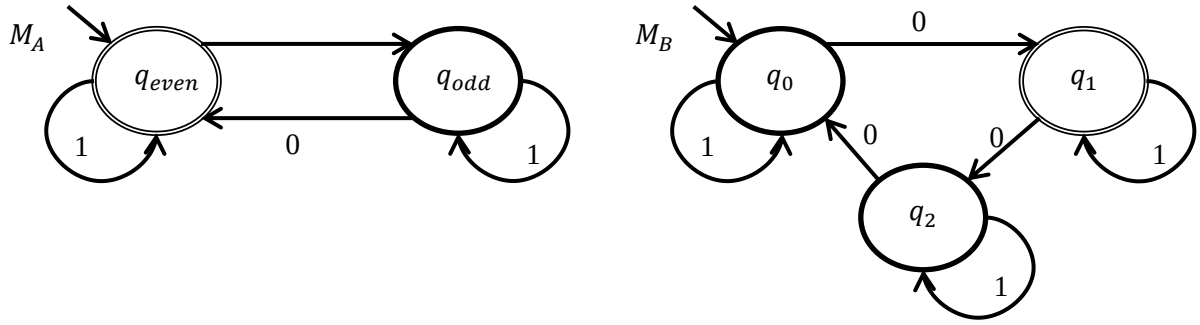
We define $M_{A \cup B} = (Q_A \times Q_B, \Sigma, \delta_{A \cup B}, \{q_A, q_B\}, F_A \times Q_B \cup Q_A \times F_B)$

And the transition function: $\delta_{A \cup B}\big((q'_A, q'_B), a\big) = \big(\delta_A(q'_A, a), \delta_B(q'_B, a)\big)$

- The transition function takes what M_A would do to the first coordinate and what M_B would do to the second coordinate.
- The accepting states are a pair where it is sufficient that only one would be an accepting state in the original automatons.

Example:

$A = \{w \in \{0,1\}^* \mid even\ number\ of\ 0's\}, B = \{w \in \{0,1\}^* \mid number\ of\ 0's\ is\ 1 \bmod 3\}$.



(complete $A \cup B$ by yourselves).

Therefore we have proven:

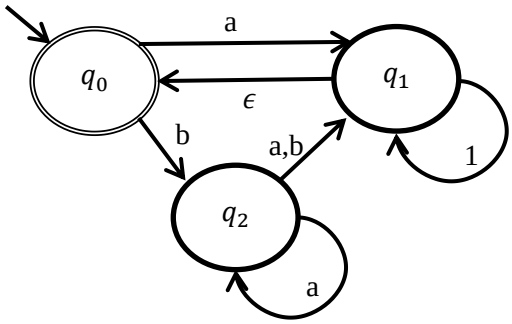
$$A, B \text{ are regular} \Rightarrow A \cup B \text{ is regular}$$

NFA – non-deterministic finite automata:

$M = (Q, \Sigma, \delta, q_o, F)$ is a NFA where:

- Q: a finite set of states
- Σ: a finite alphabet
- $\delta: Q \times \Sigma_\epsilon \to \Gamma(Q)$ where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma(Q) = \{T \subset Q\}$ ($|\Gamma(Q)| = 2^{|Q|}$)
- $q_0 \in Q$: a starting state
- $F \subset Q$: a finite state of accepting states

An example NFA:



A NFA N will accept a string w if you can "guess" the right path. Formally:

Definition

Let N be a NFA and $w \in \Sigma^*$, then $\boldsymbol{N}$ **accepts** $\boldsymbol{w}$ if $w = w_1 \dots w_n$ where $w_i \in \Sigma_\epsilon$ and there exists a sequence $r_0, \dots, r_n \in Q$ s.t. $r_0 = q_0$, $r_n \in F$ and $r_{i+1} \in \delta(r_i, w_{i+1})$ for $0 \le i \le n-1$.

Note that w is in Σ_ϵ and not Σ, and that $\delta(\cdot)$ is a set of states and not a single state.

You can say that a non-deterministic computation runs all possibilities in parallel, and it is sufficient that one will accept.

Definition:

Let M, N be automatas, then M and $N$are **equivalent** if $L(M) = L(N)$.

Theorem:

For any NFA N there is an equivalent DFA M.

Proof:

We construct the "power automata" of N. If $N = (Q, \Sigma, \delta, q_0, F)$ then let $M = (\Gamma(Q), \Sigma, \delta', \{q_0\}, \{T \subset Q | \, T \bigcap F \neq \emptyset)$, that is:

- The states are now sets of the original states
- The start state is a set (singleton) accordingly
- The accepting states are all power sets s.t. at least one of the original states it contains is an originally accepting state.
- $\delta' : \Gamma(Q) \times \Sigma \to \Gamma(Q)$ is defined as: $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$

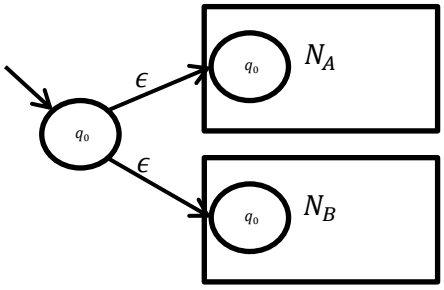
If N contains ϵ transitions, $\delta'(R, a) = E(\bigcup_{r \in R} \delta(r, a)$ and the starting state is $E(\{q_0\})$.

Note that any DFA is also a NFA, simply a very restricted one with only one option at each step and no ϵ transitions.

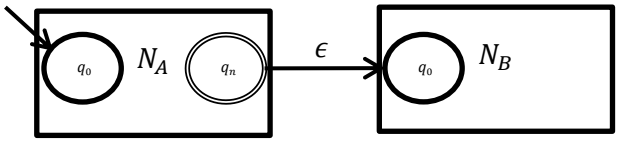
Theorem

A language is regular ⇔ it is recognized by a NFA.

Simpler proof of closure under union:



Closure under concatenation:



In addition, regular langauges are closed under:

- Kleene star
- Intersection
- Complement

Regular Expressions

A way of specifying regular languages. For instance:

$0^*10^* = \{w \in \{0,1\}^* \mid w\ contains\ exactly\ one\ 1\}$

In fact it should be written: $\{0\}^* \circ 1 \circ \{0\}^*$, as $\circ$ is the concatenation symbol and $*$ is applied on sets.

More examples:

- $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
- $R \circ \emptyset = \emptyset$ - the reason it is $\emptyset$ is that we take the first part from R and the second part is taken from $\emptyset$, but $\emptyset$ doesn't contain anything so the concatenation is also empty.
- $E \circ \{\epsilon\} = R$

Theorem

A language is regular ⇔ a regular expression describes it.

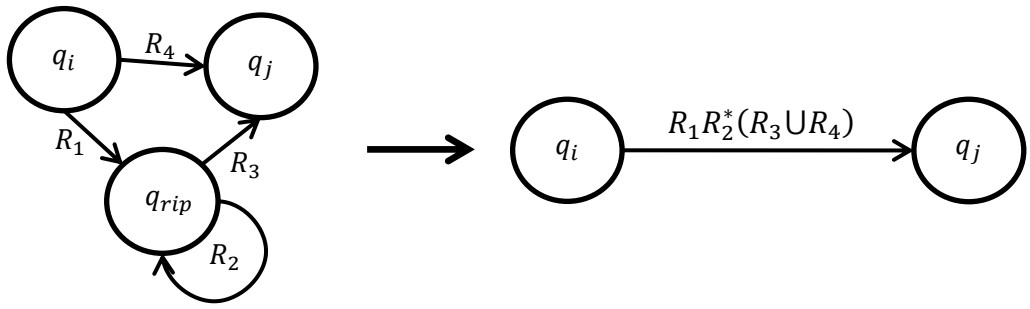
Proof:

We will show that if we have a regular expression, we can build the corresponding DFA / NFA, and vice-versa.

Examples: $\{\epsilon\}$ will be accepted by a machine with asingle accepting state and no transitions.

For the second part, assume we have a regular language, then there is a DFA that describes it and we need to build a regular expression that describes the language.

Example:



The Pumping Lemma:

Let A be a regular language, then there is an integer $p > 0$ such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where:

- $|y| > 0$
- $|xy| \leq p$
- $xy^iz \in A$, $i \geq 0$ (z can be ϵ)

The pumping lemma is used to show that languages are not regular.

For instance: 0^n1^n is not regular (looking from an automata point of view, it cannot remember how many 0's it has seen).

Using the pumping lemma:

$0^p 1^p = xyz$ s.t. $|y| > 0$ and we can repeat y as many times as we like.

- If y is only 0's, then we can put as many 0's as we like, but then we get more 0's than 1's – a contradiction.
- If y is only 1's, the same happens.
- If y contains both 0's and 1's, pumping that we could have gotten something like: 001011, which is not good.

So no y fits, therefore $0^n 1^n$ is not regular.