CSE2315 Slides week 1

Matthijs Spaan Stefan Hugtenburg

Algorithmics group
Delft University of Technology

14 February 2020

This lecture

- Strings and operations on them
- Proofs with induction over strings
- Languages and operations on them
- Proving equality of languages
- Deterministic finite automata and formal definition
- Acceptation of words and recognition of languages
- Generalized transition function δ^*
- Acceptation in terms of δ^*
- Regular languages
- Regular operations
- Extension input alphabet DFA
- Closure properties of regular languages

Alphabet and Symbols

- An alphabet is a finite, non-empty set of symbols
  Examples: $\Sigma_0 = \{0, 1\}$, $\Sigma_1 = \{a, b, c, \ldots, z\}$
- String, or word is a finite sequence of symbols from an alphabet
- Length $|w|$ of a word w. For instance: $|abc| = 3$ and $|bacacs| = 6$
- The empty string of length 0: ε
- Σ^0, Σ^1, Σ^2, . . . , Σ^n
- $\Sigma^0 = \{\varepsilon\}$,
  $\Sigma^1 = \Sigma$,
  Σ^2: all words w over Σ with $|w| = 2$,
  $\ldots$
  Σ^n: all words w over Σ with $|w| = n$.
- Set of words over Σ: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \cdots \cup \Sigma^n \cup \cdots$
- Note: for all Σ: $\varepsilon \in \Sigma^*$!

Operations on Words

- Concatenation: vw
- n-fold concatenation of w: w^n
- $w^0 = \varepsilon$ and $\varepsilon w = w = w\varepsilon$
- Word reversed: w^R
  Example: $abracadabra^R = arbadacarba$
- Substring, prefix and suffix
  Examples:
  - $abra$ is a prefix of $abracadabra$
  - $dabra$ is a suffix of $abracadabra$
  - $acad$ is a substring of $abracadabra$, but no prefix or suffix
  - $arcdba$ is not a substring of $abracadabra$

Inductive (Recursive) Definition of Word length

Generally, for $w \in \Sigma^*$ and $a \in \Sigma$ we have:

$$\begin{aligned} |\varepsilon| &= 0 \\ |wa| &= |w| + 1 \end{aligned}$$

Theorem: for all words $v, w \in \Sigma^*$:

$$|vw| = |v| + |w|.$$

Proof

By induction over $|w|$:

Basis step: If $|w| = 0$, then $w = \varepsilon$ and

$$|vw| = |v\varepsilon| = |v| = |v| + 0 = |v| + |\varepsilon| = |v| + |w|.$$

Inductive hypothesis: If $|x| = n$, then $|vx| = |v| + |x|$.

Inductive step: If $|w| = n + 1$, then $w = ua$ for some $u \in \Sigma^*$ with $|u| = n$ and for some $a \in \Sigma$. We show that $|vw| = |v| + |w|$:

$$|vw| = |vua| = |vu| + 1 =_{IH} |v| + |u| + 1 = |v| + |ua| = |v| + |w|.$$

∎

Languages

- Σ^*, Σ^+.
- A language is a subset $L \subseteq \Sigma^*$.
- An element $w \in L$ is called a word or a string of L.

Examples of languages: Let $\Sigma = \{a, b\}$:

- $\{abaab, aabbbaa, bbab, bbbb, aab\}$.
- $\{a^n b^m \mid n > m\}$.
- $\{w \in \Sigma^* \mid w = w^R\}$.
- Σ^*, $\emptyset$ and $\{\varepsilon\}$.

Operations on Languages

(see also video Operations on Regular Languages)

Set operations: $\overline{L}$, $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 - L_2$

$$
\begin{aligned}
L^R &= \{w^R \mid w \in L\} \\
L_1 \circ L_2 &= \{vw \mid v \in L_1 \text{ and } w \in L_2\} \\
L^2 &= L \circ L \;=\; LL \quad \text{(See remark 2)} \\
L^3 &= LLL \\
L^* &= L^0 \cup L^1 \cup \cdots \cup L^n \cup \cdots \\
L^+ &= L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots
\end{aligned}
$$

Edge cases: $L^0 = \{\varepsilon\}$ and $\emptyset^* = \{\varepsilon\}$.

Remark 1: $\emptyset \neq \{\varepsilon\}$ and $\varepsilon \in L^*$, for every language L.

Remark 2: We usually write $L_1 L_2$ instead of $L_1 \circ L_2$.

Examples

Let $L = \{ab, aab\}$.

$$
\begin{aligned}
L^R &= \{ba, baa\} \\
LL &= \{abab, abaab, aabab, aabaab\} \\
L^0 &= \{\varepsilon\} \\
L^2 &= \{abab, abaab, aabab, aabaab\} \\
L^* &= \{\varepsilon, ab, aab, abab, abaab, aabab, aabaab, ababab, \ldots\} \\
L^+ &= \{ab, aab, abab, abaab, aabab, aabaab, ababab, \ldots\}
\end{aligned}
$$

Some Properties of Concatenation

1. $\{\varepsilon\}L = L = L\{\varepsilon\}$
2. $\emptyset L = \emptyset = L\emptyset$
3. $L_1(L_2L_3) = (L_1L_2)L_3$
4. $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$
5. $(L_2 \cup L_3)L_1 = L_2L_1 \cup L_3L_1$

Exercise

Let $L_1, L_2, L_3 \subseteq \Sigma^*$.

1. Determine whether it holds in general that:

$$L_1(L_2 \cap L_3) = L_1L_2 \cap L_1L_3$$

2. Show that:

$$\text{If} \quad L_1 \subseteq L_2 \quad \text{then} \quad L_1L_3 \subseteq L_2L_3.$$

3. Show that:

$$L_1(L_2L_3) = (L_1L_2)L_3$$

4. Give an inductive definition of L^n $(n \geq 0)$.

Solution (1)

No!

Let:

- $L_1 = \{a, ab\}$
- $L_2 = \{b\}$
- $L_3 = \{\varepsilon\}$

Now:

- $L_1(L_2 \cap L_3) = L_1\emptyset = \emptyset$.
- $L_1L_2 \cap L_1L_3 = \{ab, abb\} \cap \{a, ab\} = \{ab\}$.

The sets $L_1(L_2 \cap L_3)$ and $L_1L_2 \cap L_1L_3$ are therefore not equal for the given choice of L_1, L_2 and L_3.

Solution (2)

Suppose $L_1 \subseteq L_2$. Take an arbitrary $w \in L_1L_3$. We must show that $w \in L_2L_3$.
Since $w \in L_1L_3$, w can be written as $w = uv$, with $u \in L_1$ and $v \in L_3$.
As $u \in L_1$ and $L_1 \subseteq L_2$, we get $u \in L_2$. This gives us $w = uv \in L_2L_3$.
Because w was arbitrarily chosen, it holds that $L_1L_3 \subseteq L_2L_3$. This proves the statement. ■

Solution (3)

First we show that $L_1(L_2L_3) \subseteq (L_1L_2)L_3$ and subsequently that $(L_1L_2)L_3 \subseteq L_1(L_2L_3)$.
We assume concatenation of words is associative, i.e., if $u, v, w \in \Sigma^*$, then $u(vw) = (uv)w$.

- Suppose $w \in L_1(L_2L_3)$.
  Then w can be written as $w = w_1w'$, with $w_1 \in L_1$ and $w' \in L_2L_3$.
  Now w' can be written as $w' = w_2w_3$, with $w_2 \in L_2$ and $w_3 \in L_3$.
  If we define $w'' = w_1w_2$, it clearly holds that $w'' \in L_1L_2$.
  We conclude that $w = w_1w_2w_3 = w''w_3 \in (L_1L_2)L_3$. ■
- Left as an exercise for the reader.

Solution (4)

Inductively define L^n for all $n \geq 0$ as follows:

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^{n+1} &= L^n L \end{aligned}$$

Relations

- An n-ary relation R is a subset of n-tuples:

$$R \subseteq A_1 \times \cdots \times A_n$$

- A binary relation R is a subset of pairs:

$$R \subseteq A \times B$$

Examples of binary relations on $\mathbb{N}^2$: $=$, $\leq$, $\geq$

Functions

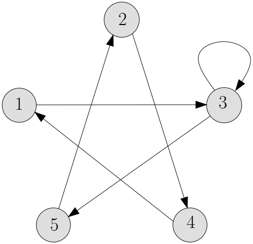
- Functions are special relations
- $f : A \to B$
- Binary functions $f : A_1 \times A_2 \to B$
- Binary functions can be displayed in a table
- Domain, co-domain vs. range
- Total vs. partial functions
- Example addition on $\mathbb{N}$: $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

Directed Graphs

A (directed) graph is a pair (V, E) where:

- V a set of nodes or vertices
- $E \subseteq V \times V$ a set of edges

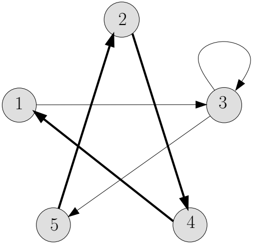
Example



$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1,3), (2,4), (3,3), (3,5), (4,1), (5,2)\} \end{aligned}$$

Directed Graphs (Cont.)

- Paths, cycles, simple cycle en loops
- Simple path
- Labeled graphs
- Subgraphs



Examples of a Formal Definition

A formal definition of an automaton makes it possible to:

1. construct neat proofs about finite automata,
2. think precisely about the material,
3. represent large automata.

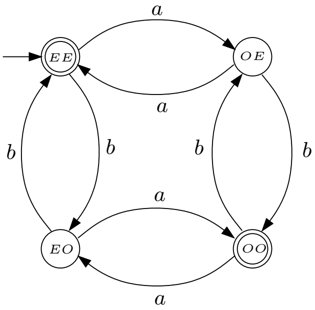
Deterministic Finite Automata (DFAs)

(see also videos Finite State Machines: Introduction and Finite State Machines: Examples)

Definition 1.5 A deterministic finite automaton (*DFA*) is an ordered quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states;
- Σ is a finite set, the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is a *total* function, the transition function;
- $q_0 \in Q$ is the start state;
- $F \subseteq Q$ is the set of accept states.

Example



$$Q = \{EE, OE, EO, OO\}$$
$$\Sigma = \{a, b\}$$
$$q_0 = EE$$
$$F = \{EE, OO\}$$

δ	a	b
EE	OE	EO
OE	EE	OO
EO	OO	EE
OO	EO	OE

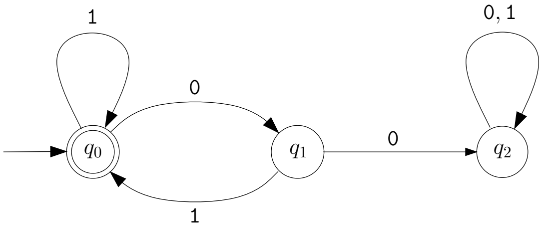
Transition graphs

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be represented as a graph with directed, labeled edges, the so-called transition graph G_M:

- G_M has exactly $|Q|$ nodes (vertices) with the states in Q as labels;
- two nodes with labels q_i and q_j are connected by the edge (q_i, q_j) with label a iff $\delta(q_i, a) = q_j$.

Exercise

Describe the following DFA D as a tuple $(Q, \Sigma, \delta, q_0, F)$:



Solution

$D = (Q, \Sigma, \delta, q_0, F)$ where:

$$
\begin{aligned}
Q &= \{q_0, q_1, q_2\} \\
\Sigma &= \{0, 1\} \\
F &= \{q_0\}
\end{aligned}
$$

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2

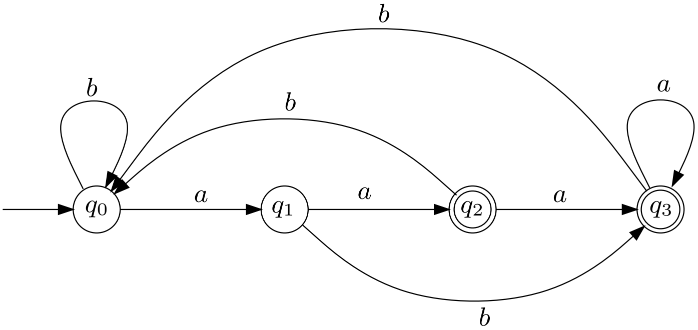
Exercise

Give a transition graph for the following DFA $M = (Q, \Sigma, \delta, q_0, F)$ where:

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3\} \\ \Sigma &= \{a, b\} \\ F &= \{q_2, q_3\} \end{aligned}$$

δ	a	b
q_0	q_1	q_0
q_1	q_2	q_3
q_2	q_3	q_0
q_3	q_3	q_0

Solution



Acceptation of Words and Languages by DFAs

Definition Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $a_1 a_2 \cdots a_n \in \Sigma^*$. M **accepts** (**recognizes**) $a_1 a_2 \cdots a_n$ iff there exist $r_0, \ldots, r_n \in Q$ such that:

1. $r_0 = q_0$,
2. $\delta(r_i, a_{i+1}) = r_{i+1} \qquad (i \geq 0)$,
3. $r_n \in F$.

Definition The language recognized by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is given by $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$.

Generalized transition function (1)

Idea: Define the function δ^*, called the generalized transition function, which answers the following question:

> *Suppose the automaton* $M = (Q, \Sigma, \delta, q_0, F)$ *is in state* $q \in Q$ *and the word* $w \in \Sigma^*$ *still has to be read. In which state will* M *be after* w *has been processed/read in its entirety?*

How does δ^* work? For each letter in w, the function δ^* performs an iteration of δ on the state in which M arrived during the previous iteration. The last state that M reaches this way is the outcome of this process. Notation: $\delta^*(q, w)$.

Since δ^* works using iterations, it can be defined inductively using recursion (see next slide).

Generalized transition function (2)

Basis

Zero steps are needed to process the empty string ε, so M stays in state q. That is,

$\delta^*(q, \varepsilon) = q$.

Inductive clause

Suppose we know that M goes from state $q \in Q$ to state $q' \in Q$ while processing $w \in \Sigma^*$, that is, $\delta^*(q, w) = q'$. Then, while processing wa (with $a \in \Sigma$), M first goes from q to q', after which it must process the letter a from q', resulting in $\delta(q', a)$. That is,

$\delta^*(q, wa) = \delta(q', a) = \delta(\delta^*(q, w), a)$.

This motivates the inductive definition on the next slide.

Inductive Definition of Acceptation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

Definition $\delta^* : Q \times \Sigma^* \to Q$ is defined inductively for every $q \in Q$ as follows:

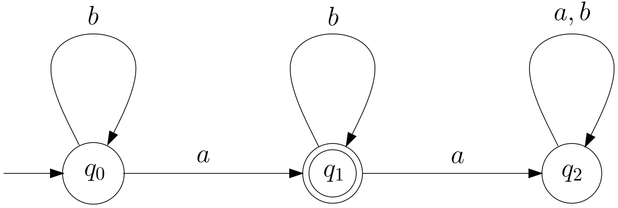
$$\begin{array}{rcl} \delta^*(q, \varepsilon) & = & q, \\ \delta^*(q, wa) & = & \delta(\delta^*(q, w), a), \end{array}$$

- M accepts $w \in \Sigma^*$ iff $\delta^*(q_0, w) \in F$
- $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$
- $\delta^*(q, a) = \delta^*(q, \varepsilon a) = \delta(\delta^*(q, \varepsilon), a) = \delta(q, a)$

Note: A DFA accepts ε iff the start state is an accepting state.

Exercise

Suppose we have the following DFA $M = (Q, \Sigma, \delta, q_0, F)$:



Use δ^* to show that $bba \in L(M)$ but that $aba \notin L(M)$.

Solution (1)

$$\begin{array}{rcl} \delta^*(q_0, bba) & = & \delta(\delta^*(q_0, bb), a) \\ & = & \delta(\delta(\delta^*(q_0, b), b), a) \\ & = & \delta(\delta(\delta(\delta^*(q_0, \varepsilon), b), b), a) \\ & = & \delta(\delta(\delta(q_0, b), b), a) \\ & = & \delta(\delta(q_0, b), a) \\ & = & \delta(q_0, a) \\ & = & q_1 \\ & \in & F. \end{array}$$

This means $bba \in L(M)$.

Solution (2)

$$
\begin{aligned}
\delta^*(q_0, aba) &= \delta(\delta^*(q_0, ab), a) \\
&= \delta(\delta(\delta^*(q_0, a), b), a) \\
&= \delta(\delta(\delta(\delta^*(q_0, \varepsilon), a), b), a) \\
&= \delta(\delta(\delta(q_0, a), b), a) \\
&= \delta(\delta(q_1, b), a) \\
&= \delta(q_1, a) \\
&= q_2 \\
&\notin F.
\end{aligned}
$$

This means that $aba \notin L(M)$.

Regular languages

Definition A language L is regular iff there exists a DFA M with $L = L(M)$.

Exercise

1. Is the language consisting of words with exactly one a regular?
2. Is every regular language finite?
3. Is every language consisting of exactly one word regular?
4. Is every finite language regular?

Answers

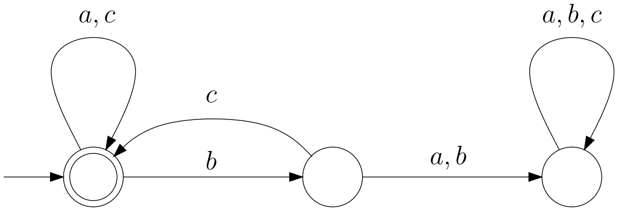
1. Of course, the DFA for this language is on slide 32.
2. No. The language consisting of words with exactly one a is infinite.
3. Yes! Why?
4. Yes! Why?

Exercise

Let $\Sigma = \{a, b, c\}$. Design a DFA that accepts precisely all words w in which every b is followed by at least one c.

Note Words in which no b occurs are also accepted.

Solution



Exercise

Reminder $\delta^* : Q \times \Sigma^* \to Q$ is defined inductively for each $q \in Q$ as follows:

$$\begin{array}{rcl} \delta^*(q, \varepsilon) & = & q, \\ \delta^*(q, wa) & = & \delta(\delta^*(q, w), a), \end{array}$$

Exercise Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Using induction over the length $|w|$, prove that for each $w \in \Sigma^*$ and each $q \in Q$:

$$\delta^*(q, w) \in Q$$

Also specify what must be proven and what your induction hypothesis is.

Hint Based on the definition of a DFA, the following holds for each $a \in \Sigma$ and $q \in Q$: $\delta(q, a) \in Q$.

Solution

Basis step: If $|w| = 0$, then $w = \varepsilon$. Now, if $q \in Q$, we get:

$$\delta^*(q, \varepsilon) =_{\text{definition}} q \in Q.$$

Inductive hypothesis (IH): For all $w \in \Sigma^*$ with $|w| = n$ and $q \in Q$: $\delta^*(q, w) \in Q$.

Inductive step: Given the IH, we must show that the statement holds for words $v \in \Sigma^*$, with $|v| = n + 1$. If $|v| = n + 1$, there exists a $w \in \Sigma^*$ with $|w| = n$ and an $a \in \Sigma$ such that $v = wa$. We get:

$$\delta^*(q, v) = \delta^*(q, wa) =_{\text{definition}} \delta(\delta^*(q, w), a).$$

From the IH, we derive that $\delta^*(q, w) = q'$ for some $q' \in Q$. Therefore:

$$\delta^*(q, v) = \delta(q', a) \in Q,$$

because of the definition of a DFA.

This proves the statement.

Closure Properties of Regular Languages

Question Suppose L, L_1 and L_2 are regular languages. Which of the following languages are also regular?

1. $\overline{L}$
2. L^R
3. $L_1 \cap L_2$
4. $L_1 \cup L_2$
5. $L_1 L_2$
6. L^*

Regular Languages and Closure under Complement

Theorem The class of regular languages is closed under complement. That is, if L is regular, then $\overline{L}$ is also regular.

Proof sketch

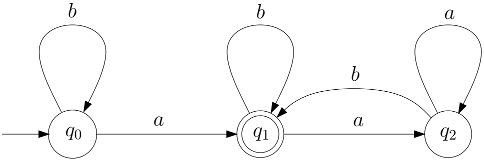
1. If L is regular, there exists a DFA M recognizing L, i.e., $L(M) = L$.
2. Show that M can be transformed into an automaton $\overline{M}$ recognizing $\overline{L}$.
3. Let $M = (Q, \Sigma, \delta, q_0, F)$. Define:

$$\overline{M} = (Q, \Sigma, \delta, q_0, Q - F)$$

4. Show that $\overline{L(M)} = L(\overline{M})$

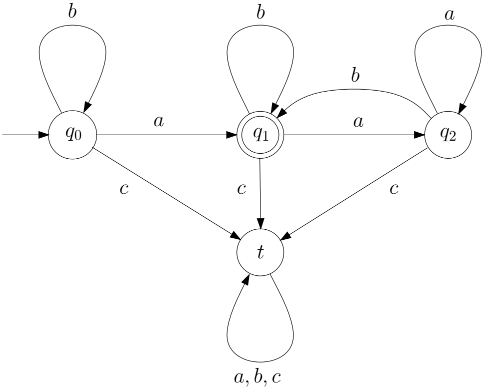
Exercise as intermezzo

Suppose we have the following DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $\Sigma = \{a, b\}$:



Construct a DFA M' with input alphabet $\Sigma' = \{a, b, c\}$ such that $L(M') = L(M)$.

Solution



Extension of the input alphabet

The automaton M on slide 44 has had its input alphabet Σ extended with the letter c: $\Sigma' = \Sigma \cup \{c\}$. We could then construct an automaton M' with alphabet Σ' such that $L(M') = L(M)$.

This can be generalized: an input alphabet can be extended arbitrarily using the construction on slide 44.

Theorem   Let Σ and Σ' be alphabets such that $\Sigma \subseteq \Sigma'$. Then for every DFA $M = (Q, \Sigma, \delta, q_0, F)$, a DFA $M' = (Q', \Sigma', \delta', q_0', F')$ exists with $L(M') = L(M)$.

Proof

Let Σ and Σ' be alphabets such that $\Sigma \subseteq \Sigma'$. Also, let a DFA M be given as $M = (Q, \Sigma, \delta, q_0, F)$.

Define $M' = (Q', \Sigma', \delta', q'_0, F')$ such that:

$$\begin{aligned} Q' &= Q \cup \{q_t\} \\ q'_0 &= q_0 \\ F' &= F \end{aligned}$$

Also define $\delta'(q, a)$ for each $q \in Q'$ and $a \in \Sigma'$ as:

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } q \in Q \text{ and } a \in \Sigma, \\ q_t & \text{otherwise.} \end{cases}$$

Then we have $L(M') = L(M)$. ■

Regular Languages are Closed under Union

(see also video Operations on Regular Languages)

Theorem The class of regular languages is closed under union. In other words, if L_1 and L_2 are regular, then $L_1 \cup L_2$ is also regular.

Proof idea:

1. If L_1 and L_2 are regular, there exist DFAs M_1 and M_2 with $L(M_1) = L_1$ and $L(M_2) = L_2$.
2. We now construct a DFA M that lets M_1 and M_2 compute in parallel on a given input word w.
3. As soon as one of the DFAs M_1 or M_2 accepts the input word w, M accepts w.

Proof

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be the DFAs such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

We construct a DFA M such that:

$$L(M) = L(M_1) \cup L(M_2).$$

Note: Because of the theorem on input alphabet extension, we can assume that M_1 and M_2 have the same alphabet Σ.

Construction for Union

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Define $M = (Q, \Sigma, \delta, q_0, F)$ such that:

$$\begin{array}{rcl} Q & = & Q_1 \times Q_2 \\ q_0 & = & (q_1, q_2) \\ F & = & \{(q, q') \mid q \in F_1 \text{ or } q' \in F_2\} \end{array}$$

For all $a \in \Sigma$ and all $(q, q') \in Q$, define δ as follows:

$$\delta\left((q, q'), a\right) = \left(\delta_1(q, a), \delta_2(q', a)\right)$$

(One step of M via $\delta\left((q, q'), a\right)$ corresponds to two parallel steps of M_1 and M_2 via $\delta_1(q, a)$ and $\delta_2(q', a)$, respectively.)

Proof (Cont.)

To be proven: $L(M) = L(M_1) \cup L(M_2)$.

Sketch: We use δ^* for this. By induction over $|w|$, one can show (try this yourself!):

$$\boxed{\delta^*\left((q, q'), w\right) = \left(\delta_1^*(q, w), \delta_2^*(q', w)\right)}$$

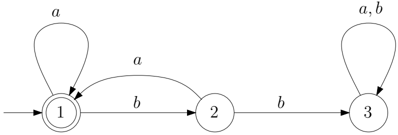
From this it follows that

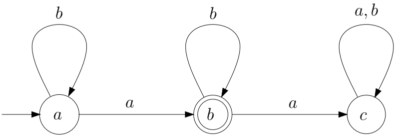
$$\delta^*\left((q, q'), w\right) \in F \quad \text{iff} \quad \delta_1^*(q, w) \in F_1 \text{ or } \delta_2^*(q', w) \in F_2.$$

This means that $L(M) = L(M_1) \cup L(M_2)$. ■

Exercise

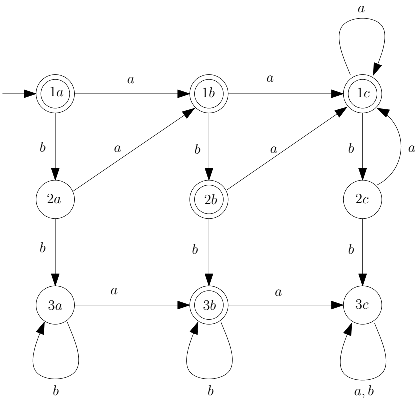
Let M_1 and M_2 be given by, respectively:





Construct a DFA M such that $L(M) = L(M_1) \cup L(M_2)$.

Solution



Exercise

Show that the class of regular languages is closed under intersection.

1. What exactly does this mean? What must be proven?
2. Is it possible, given two DFAs M_1 and M_2, to construct a third DFA M such that $L(M) = L(M_1) \cap L(M_2)$?
3. Is it possible to give a “smarter” (less laborious) proof?

Answers

1. That for all regular languages L_1 and L_2, the intersection $L_1 \cap L_2$ is also regular.
2. Yes, in the same way as the proof for union. Only difference is:

$$F = \{(q, q') \mid q \in F_1 \text{ and } q' \in F_2\}$$

3. Yes. After all, we have $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, and we know regular languages are closed under union and complement.

Exercise

Construct a DFA that recognizes the language containing all strings with aba as substring.

Solution