FORMAL LANGUAGES AND AUTOMATA

Languages

Intuitively, a language is defined as a set of strings. A string is defined as a finite ordered set of symbols that originate from a finite set Σ, which we call the alphabet. To provide a rigorous definition we recall that

$$\mathbb{N}^* = \{1, 2, 3, \ldots\}$$

and that given a set A, via the Cartesian product, we define

$$A^2 = A \times A = \{(a, b) \mid a, b \in A\}$$
$$A^3 = A \times A \times A = \{(a, b, c) \mid a, b, c \in A\}$$
$$A^4 = A \times A \times A \times A = \{(a, b, c, d) \mid a, b, c, d \in A\}$$

etc.

The n^{th} case is defined as

$$A^n = \{(x_1, x_2, \ldots, x_n) \mid \forall k \in [n] : x_k \in A\}$$

For purposes of the definitions below, we also define

$$A^0 = \{\emptyset\}$$

with $\emptyset$ the empty set.

→ Definition of strings and languages

From the above concepts, we define the notion of language

rigorously as follows:

Def: Given a set Σ, we define

a) The star-closure Σ^* (also: Kleene closure) as

$$\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

b) The positive closure Σ^+ as:

$$\Sigma^+ = \bigcup_{n \in \mathbb{N} - \{0\}} \Sigma^n = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

↳ The corresponding belonging conditions for Σ^* and Σ^+ are given by

$u \in \Sigma^* \iff \exists n \in \mathbb{N} : u \in \Sigma^n$

$u \in \Sigma^+ \iff \exists n \in \mathbb{N} - \{0\} : u \in \Sigma^n$

Def: Let L, Σ be two sets. We say that

L language with alphabet $\Sigma \iff L \subseteq \Sigma^*$

$\iff L \in \mathcal{P}(\Sigma^*)$

notation:

a) Strings are essentially n-tuples but we prefer to denote them as an aggregation of symbols

e.g. for the alphabet $\Sigma = \{a, b, c\}$ we denote

$u = abbc = (a, b, b, c)$

and note that since

$abbc \in \Sigma^4 \Rightarrow \exists n \in \mathbb{N} : abbc \in \Sigma^n$

$\Rightarrow abbc \in \Sigma^*$

b) We use power notation to represent repeating symbols.

e.g. $ab^2c^3b = abbcccb =$

$= (a, b, b, c, c, c, b)$

c) Given a string $u \in \Sigma^*$, the n^{th} symbol of u is represented as u_n.

e.g. For $u = ab^2c = abbc$, we have $u_2 = u_3 = b$ and $u_1 = a$ and $u_4 = c$

Remark: Note that $\Sigma^0 = \{\emptyset\}$, therefore for any alphabet Σ we have $\emptyset \in \Sigma^*$. In the context of formal languages, we define $\lambda = \emptyset$ with λ representing the empty string (or null string).

String properties and operations

Def: Let Σ be an alphabet and let $u \in \Sigma^*$ be a string. We define:

a) The length $|u|$ of u via:

$|u| = m \Leftrightarrow u \in \Sigma^m$

b) For any letter $a \in \Sigma$ of the alphabet, $n_a(u)$ is the number of occurences of the letter a in the string u, and we define it formally as:

$n_a(u) = |\{k \in [|u|] \mid u_k = a\}|$

e.g. Consider $u = a^2bad^2b$. Then

$|u| = 2+1+1+2+1 = 7$ $\quad n_b(u) = 1+1 = 2$

$n_a(u) = 2+1 = 3$ $\quad n_d(u) = 2$

Def : (string concatenation)

a) Let Σ be an alphabet and let $u, v \in \Sigma^+$ be two non-null strings. We define the concatenation $uv \in \Sigma^+$ as follows:

$$\forall a \in [\![1, |u|+|v|]\!] : (uv)_a = \begin{cases} u_a & , \text{if } 1 \leq a \leq |u| \\ v_{a-|u|} & , \text{if } |u| < a \leq |u|+|v| \end{cases}$$

b) To extend concatenation to strings in Σ^*, we also define:

$$\begin{cases} \forall u \in \Sigma^+ : \lambda u = u\lambda = u \\ \lambda\lambda = \lambda \end{cases}$$

Remark : An immediate consequence of this definition is that

$$\forall u, v \in \Sigma^* : |uv| = |u| + |v|$$

e.g: Consider $\Sigma = \{a, b, c, d\}$ and $u, v \in \Sigma^*$ with $u = ab^2db$ and $v = bc^2a$. Then:

$|u| = |ab^2db| = 1+2+1+1 = 5$

$|v| = |bc^2a| = 1+2+1 = 4$

$uv = (ab^2db)(bc^2a) = ab^2db^2c^2a$

$|uv| = |u| + |v| = 5+4 = 9.$

Def : (String reversal)

Let Σ be an alphabet and let $u \in \Sigma^+$ be a non-null string.

a) We define the reverse string u^R via

$$\forall a \in [|u|] : (u^R)_a = u_{|u|+1-a}$$

b) We also define: $\lambda^R = \lambda$.

e.g. For $u = ab^2dac$, the reverse string is

$$u^R = cadb^2a$$

→ **Language operations**

Def : (Language concatenation)

Let Σ be an alphabet and consider two languages $L_1, L_2 \in \mathcal{P}(\Sigma^*)$. We define a new language $L_1L_2 \in \mathcal{P}(\Sigma^*)$ (the concatenation of L_1 and L_2) as:

$$L_1L_2 = \{uv \mid u \in L_1 \wedge v \in L_2\}$$

Def : (Language concatenation power)

Let Σ be an alphabet and let $L \in \mathcal{P}(\Sigma^*)$ be a language. We define L^n for all $n \in \mathbb{N}$ recursively as follows:

$$\begin{cases} L^0 = \{\lambda\} \\ L^1 = L \\ \forall n \in \mathbb{N}^* : L^{n+1} = LL^n \end{cases}$$

Def: (Star-closure and positive closure of languages)
Let Σ be an alphabet and let $L \in \mathcal{P}(\Sigma^*)$ be a language. We define the star-closure L^* and the positive closure L^+ of L as follows:

$$L^* = \bigcup_{n \in \mathbb{N}} L^n \quad \text{and} \quad L^+ = \bigcup_{n \in \mathbb{N} - \{0\}} L^n$$

EXAMPLES

a) Consider the languages
$L_1 = \{\lambda, ab, ac\}$ and $L_2 = \{b, da\}$
Evaluate L_1L_2, L_1^2, L_2^2.

Solution

$$L_1L_2 = \{\lambda, ab, ac\}\{b, da\} =$$
$$= \{\lambda b, \lambda da, abb, abda, acb, acda\} =$$
$$= \{b, da, ab^2, abda, acb, acda\}.$$
$$L_1^2 = L_1L_1 = \{\lambda, ab, ac\}\{\lambda, ab, ac\} =$$
$$= \{\lambda^2, \lambda ab, \lambda ac, ab\lambda, abab, abac, ac\lambda, acab, acac\}$$
$$= \{\lambda, ab, ac, abab, abac, acab, acac\}$$
$$L_2^2 = L_2L_2 = \{b, da\}\{b, da\} =$$
$$= \{bb, bda, dab, dada\} \quad \square$$

b) Let $L_1 = \{a^2b\}$ and $L_2 = \{ba\}$. Evaluate using set builder notation the languages $L_3 = L_1L_2^* \cup L_1^*L_2$ and $L_1^*L_2^*$.

Solution

Since
$$L_1^* = \{a^2b\}^* = \bigcup_{n\in\mathbb{N}} \{a^2b\}^n = \bigcup_{n\in\mathbb{N}} \{(a^2b)^n\} =$$
$$= \{(a^2b)^n \mid n \in \mathbb{N}\}$$
and

$$L_2^* = \{ba\}^* = \bigcup_{n\in\mathbb{N}} \{ba\}^n = \bigcup_{n\in\mathbb{N}} \{(ba)^n\} =$$

$$= \{(ba)^n \mid n\in\mathbb{N}\}$$

it follows that

$$L_3 = L_1 L_2^* \cup L_1^* L_2 = \{a^2b\}\{(ba)^n \mid n\in\mathbb{N}\} \cup \{(a^2b)^n \mid n\in\mathbb{N}\}\{ba\}$$
$$= \{a^2b(ba)^n \mid n\in\mathbb{N}\} \cup \{(a^2b)^n ba \mid n\in\mathbb{N}\}$$
$$= \{a^2b(ba)^n, (a^2b)^n ba \mid n\in\mathbb{N}\}.$$

and

$$L_1^* L_2^* = \{(a^2b)^n \mid n\in\mathbb{N}\}\{(ba)^n \mid n\in\mathbb{N}\} =$$
$$= \{(a^2b)^n (ba)^m \mid n\in\mathbb{N} \wedge m\in\mathbb{N}\} \quad \square$$

c) Let $L = \{u\in\Sigma^* \mid n_a(u) < n_b(u)\}$ be a language on the alphabet $\Sigma = \{a,b\}$. Show that $L^2 \subseteq L$.

Solution

We note that

$$L^2 = LL = \{u\in\Sigma^* \mid n_a(u) < n_b(u)\}\{v\in\Sigma^* \mid n_a(v) < n_b(v)\}$$
$$= \{uv \mid u\in\Sigma^* \wedge v\in\Sigma^* \wedge n_a(u) < n_b(u) \wedge n_a(v) < n_b(v)\}$$
$$= \{uv \mid u,v\in\Sigma^* \wedge n_a(u) < n_b(u) \wedge n_a(v) < n_b(v)\}.$$

Let $w\in L^2$ be given. Then:

$w\in L^2 \Rightarrow \exists u,v\in\Sigma^* : (n_a(u) < n_b(u) \wedge n_a(v) < n_b(v))$

Choose $u,v\in\Sigma^*$ such that $n_a(u) < n_b(u)$ and $n_a(v) < n_b(v)$.

We have:

$$n_a(w) = n_a(uv) = n_a(u) + n_a(v)$$
$$< n_b(u) + n_a(v) \quad [\text{via } n_a(u) < n_b(u)]$$
$$< n_b(u) + n_b(v) \quad [\text{via } n_a(v) < n_b(v)]$$

$= n_B(uv) = n_B(w) \Rightarrow$

$\Rightarrow n_a(w) < n_B(w) \Rightarrow w \in L$

From the above argument we have

$(\forall w \in L^2 : w \in L) \Rightarrow L^2 \subseteq L.$

EXERCISES

(1) Let $L_1 = \{a, b, b^2\}$ and $L_2 = \{ab, a^2\}$ be languages.
a) Evaluate L_1L_2 and L_2L_1
b) Evaluate $L_1^2, L_1^3, L_2^2, L_2^3$.

(2) Let $L = \{abc, b\}$ be a language. Evaluate
L^2, L^3, L^4.

(3) Let $L_1 = \{b^2\}$ and $L_2 = \{a^3\}$. Use definition of set by mapping notation to define the following languages and write the corresponding belonging condition
a) L_1^* b) L_2^* c) $L_1^*L_2^*$ d) $(L_1L_2)^*$

(4) Let $L_1 = \{ab^2\}$ and $L_2 = \{b^4\}$ and $L_3 = \{a^2\}$.
Use definition of set by mapping notation to define the following languages and write the corresponding belonging condition.
a) $L_1L_2^* \cup L_1^*L_2$
b) $L_1^*L_2^*$
c) $(L_2L_3)^*$
d) $L_1^*(L_2^* \cup L_3^*)$
e) $(L_1^* \cup L_2^*)L_3^*$
f) $L_1(L_2L_3)^* \cup (L_1L_2)^*L_3$

(5) Let $\Sigma = \{a, b\}$ be an alphabet and define
$L = \{u \in \Sigma^* \mid n_a(u) = n_b(u)\}$
Show that $L^* = L$

(Hint: A preliminary step is to show by induction that $\forall n \in \mathbb{N} - \{0,1\}: L^n \subseteq L$.)

⑥ Let $\Sigma = \{a, b\}$ be an alphabet and define

$L_1 = \{u \in \Sigma^* \mid n_a(u) = n_b(u) + 1\}$

$L_2 = \{u \in \Sigma^* \mid n_a(u) = n_b(u) + 2\}$

a) Show that $(\{a\}L_1) \cup (L_1\{a\}) \subseteq L_2$

b) Find a counterexample that disproves the claim $(\{a\}L_1) \cup (L_1\{a\}) = L_2$.

c) Show that $L_1^* = L_1$.

⑦ Let $\Sigma = \{a, b\}$ be an alphabet.

a) Show that Σ^* is countably infinite.

b) Is the set of all languages using alphabet $\Sigma = \{0,1\}$ countable or uncountable?

Grammars

Grammars provide a method for defining languages that can be more powerful than set builder notation. We begin with an alphabet set Σ. In grammar terminology Σ is called a set of terminal symbols. Then, we make the following definitions:

Def: A grammar G is defined as a 4-tuple $G = (V, \Sigma, \$, P)$ where

a) V is a set of variables

b) Σ is the alphabet

c) $\$ \in V$ is the start variable

d) $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ is a set of productions

such that

$$\begin{cases} V, \Sigma, P \text{ are finite sets} \\ V \cap \Sigma = \emptyset \\ V \neq \emptyset \wedge \Sigma \neq \emptyset \end{cases}$$

Remark: Production rules describe how the grammar is allowed to transform one string into another. Given the production rule $(x \to y) \in P$ with $x \in (V \cup \Sigma)^+$ and $y \in (V \cup \Sigma)^*$, the grammar is allowed to transform any string of the form $u = axb$ with $a, b \in (V \cup \Sigma)^*$ to $v = ayb$, in which case we write $u \underset{G}{\Rightarrow} v$.

We now give the formal definitions corresponding to the previous remark:

Def: Let $G=(V,\Sigma,S,P)$ be a grammar and let $u,v\in(V\cup\Sigma)^*$. We say that

a) $u \underset{G}{\Rightarrow} v \iff \exists a,b,y\in(V\cup\Sigma)^* : \exists x\in(V\cup\Sigma)^+ :$
$$: \begin{cases} u = axb \wedge v = ayb \\ (x\to y)\in P \end{cases}$$

b) $$\begin{cases} u \overset{1}{\underset{G}{\Rightarrow}} v \iff u \underset{G}{\Rightarrow} v \\ \forall n\in\mathbb{N}^* : \left(u \overset{n+1}{\underset{G}{\Rightarrow}} v \iff \exists w\in(V\cup\Sigma)^* : \left(u \underset{G}{\Rightarrow} w \wedge w \overset{n}{\underset{G}{\Rightarrow}} v\right)\right.\end{cases}$$

c) $u \overset{*}{\underset{G}{\Rightarrow}} v \iff \exists n\in\mathbb{N}^* : u \overset{n}{\underset{G}{\Rightarrow}} v$

Remark: $u \underset{G}{\Rightarrow} v$ means that u derives v with the application of exactly one production rule.
$u \overset{n}{\underset{G}{\Rightarrow}} v$ means that u derives v with the application of exactly n production rules. Finally, $u \overset{*}{\underset{G}{\Rightarrow}} v$ means that u derives v with the application of an arbitrary number of production rules.

Def: Let $G=(V,\Sigma,S,P)$ be a grammar. We define the language $\mathcal{L}(G)$ generated by the grammar G via
$$\mathcal{L}(G) = \{u\in\Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} u\}$$

The corresponding belonging condition is:

$$u \in \mathcal{L}(G) \Leftrightarrow u \in \Sigma^* \wedge S \overset{*}{\underset{G}{\Rightarrow}} u$$

Remark: It is easy to see that the relation "$\overset{*}{\underset{G}{\Rightarrow}}$" is transitive, in the sense that:

$$\boxed{\forall u,v,w \in (V \cup \Sigma)^* : ((u \overset{*}{\underset{G}{\Rightarrow}} v \wedge v \overset{*}{\underset{G}{\Rightarrow}} w) \Rightarrow (u \overset{*}{\underset{G}{\Rightarrow}} v))}$$

An immediate consequence is the following lemma:

$$\boxed{\forall u,v \in \Sigma^* : \left(\begin{cases} u \in \mathcal{L}(G) \\ u \overset{*}{\underset{G}{\Rightarrow}} v \end{cases} \Rightarrow v \in \mathcal{L}(G) \right)}$$

↕→ notation

If a grammar contains production rules of the form $X \to y_1$ and $X \to y_2$, we can rewrite them in condensed form as $X \to y_1 | y_2$. In general, given production rules $X \to y_1, X \to y_2, \ldots, X \to y_n$, we can rewrite them as $X \to y_1 | y_2 | \ldots | y_n$.

EXAMPLE

Consider the grammar $G = (V, \Sigma, S, P)$ with $V = \{S, A\}$ and $\Sigma = \{a, b\}$ and production rules

$$S \longrightarrow Ab$$
$$A \longrightarrow aAb$$
$$A \longrightarrow \lambda$$

Show that

a) $aabbb \in \mathcal{L}(G)$

b) $\mathcal{L}(G) = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$.

Solution

a) Since:

$$\begin{aligned} S &\overset{G}{\Rightarrow} Ab && [\text{via } S \to Ab] \\ &\overset{G}{\Rightarrow} aAbb && [\text{via } A \to aAb] \\ &\overset{G}{\Rightarrow} aaAbbb && [\text{via } A \to aAb] \\ &\overset{G}{\Rightarrow} aa\lambda bbb && [\text{via } A \to \lambda] \\ &= aabbb \end{aligned}$$

it follows that $aabbb \in \mathcal{L}(G)$.

b) It is sufficient to show that

$$\begin{cases} \forall u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\} : u \in \mathcal{L}(G) \\ \forall u \in \mathcal{L}(G) : u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\} \end{cases}$$

($\Rightarrow$) : Let $u \in \mathcal{L}(G)$ be given. It follows that $S \overset{k}{\underset{G}{\Rightarrow}} u$.

Note that the first step has to be $S \longrightarrow Ab$.

The next p steps have no choice but to be $A \to aAb$

Then the only way to terminate by eliminating A is to apply $A \to \lambda$. This results in the derivation

$$S \underset{G}{\Rightarrow} Ab \underset{G}{\overset{p}{\Rightarrow}} a^p A b^p b \underset{G}{\Rightarrow} a^p \lambda b^p b = a^p b^{p+1}.$$

It follows that if the derivation of u uses p steps $A \to aAb$ that

$$u = a^p b^{p+1} \Rightarrow \exists n \in \mathbb{N} : u = a^n b^{n+1}$$
$$\Rightarrow \underline{u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\}}$$

$\underline{(\Leftarrow)}$: Let $u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\}$ be given. Then

$u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\} \Rightarrow \exists n \in \mathbb{N} : u = a^n b^{n+1}$

Choose some $n_0 \in \mathbb{N}$ such that $u = a^{n_0} b^{n_0+1}$.

We claim that $\forall n \in \mathbb{N} : S \overset{*}{\underset{G}{\Rightarrow}} a^n A b^{n+1}$. We prove the claim by induction:

For $n = 0$: $u = a^0 b^{0+1} = \lambda b = b$ and since

$$S \underset{G}{\Rightarrow} Ab \quad [\text{via } S \to Ab]$$
$$\underset{G}{\Rightarrow} \lambda b \quad [\text{via } A \to \lambda]$$
$$= b = a^0 b^1$$

we have : $S \overset{*}{\underset{G}{\Rightarrow}} a^0 b^1$.

For $n = k$, assume that $S \overset{*}{\underset{G}{\Rightarrow}} a^k A b^{k+1}$

For $n = k+1$, we will show that $S \overset{*}{\underset{G}{\Rightarrow}} a^{k+1} A b^{k+1}$.

Since

$$S \overset{*}{\underset{G}{\Rightarrow}} a^k A b^{k+1} \quad [\text{induction hypothesis}]$$
$$\underset{G}{\Rightarrow} a^k a A b b^{k+1} \quad [\text{via } A \to aAb]$$
$$= a^{k+1} A b^{k+2}$$

we have shown by induction that

$$\forall n \in \mathbb{N} : S \Rightarrow a^n A b^{n+1} \qquad (1)$$

It follows that

$$\left(\$ \overset{*}{\underset{G}{\Rightarrow}} a^{n_0} A b^{n_0+1} \quad [\text{via Eq. (1)}]\right.$$
$$\overset{*}{\underset{G}{\Rightarrow}} a^{n_0} \lambda b^{n_0+1} \quad [\text{via } A \to \lambda]$$
$$\left. = a^{n_0} b^{n_0+1}\right) \Rightarrow \left(\$ \overset{*}{\underset{G}{\Rightarrow}} a^{n_0} b^{n_0+1}\right) \Rightarrow$$
$$\Rightarrow u = a^{n_0} b^{n_0+1} \in \mathcal{L}(G).$$

From the above argument:

$$\begin{cases} \forall u \in \mathcal{L}(G) : u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\} \Rightarrow \\ \forall u \in \{a^n b^{n+1} \mid n \in \mathbb{N}\} : u \in \mathcal{L}(G) \end{cases}$$
$$\Rightarrow \begin{cases} \mathcal{L}(G) \subseteq \{a^n b^{n+1} \mid n \in \mathbb{N}\} \\ \{a^n b^{n+1} \mid n \in \mathbb{N}\} \subseteq \mathcal{L}(G) \end{cases}$$
$$\Rightarrow \mathcal{L}(G) = \{a^n b^{n+1} \mid n \in \mathbb{N}\}$$

EXAMPLE

Consider the grammar $G=(V,\Sigma,\$,P)$ with $V=\{\$\}$ and $\Sigma=\{a,b\}$ and production rules

$$\$ \to \$\$ \mid \lambda \mid a\$b \mid b\$a$$

Show that $\mathcal{L}(G)=\{u\in\Sigma^* \mid n_a(u)=n_b(u)\}$

Solution

($\Rightarrow$): Let $v\in\mathcal{L}(G)$ be given. We note that

a) the production rules $\$\to a\$b$ and $\$\to b\$a$ generate an equal number of "a" and "b"

b) the production rules $\$\to\$\$$ and $\$\to\lambda$ do not modify the number of "a" and "b".

It follows that

$$n_a(v)=n_b(v) \Rightarrow v\in\{u\in\Sigma^* \mid n_a(u)=n_b(u)\}.$$

($\Leftarrow$): It is sufficient to show that

$$\forall \mu\in\mathbb{N}: \forall v\in\Sigma^*: (n_a(v)=n_b(v)=\mu \Rightarrow v\in\mathcal{L}(G))$$

We use proof by induction of $\mu\in\mathbb{N}$.

For $\mu=0$: Let $v\in\Sigma^*$ be given such that $n_a(v)=n_b(v)=0$.

Then $|v|=n_a(v)+n_b(v)=0+0=0 \Rightarrow v=\lambda$

and therefore, via the production rule $\$\to\lambda$:

$$\$ \underset{G}{\Rightarrow} \lambda = v \Rightarrow v\in\mathcal{L}(G).$$

For $\mu=\mu_0>0$, assume that

$$\forall \mu\in\mathbb{N}: \forall v\in\Sigma^*: (n_a(v)=n_b(v)\leq\mu_0 \Rightarrow v\in\mathcal{L}(G))$$

For $\mu=\mu_0+1$, we will show that

$$\forall v\in\Sigma^*: (n_a(v)=n_b(v)=\mu_0+1 \Rightarrow v\in\mathcal{L}(G))$$

Let $v \in \Sigma^*$ be given and assume that $n_a(v) = n_b(v) = \mu_0 + 1$.
We distinguish between the following cases:

Case 1: Assume that $v = awb$ with $w \in \Sigma^*$. Then

$$\left.\begin{aligned} n_a(w) &= n_a(awb) - 1 = (\mu_0+1) - 1 = \mu_0 \\ n_b(w) &= n_b(awb) - 1 = (\mu_0+1) - 1 = \mu_0 \end{aligned}\right\} \Rightarrow$$

$\Rightarrow n_a(w) = n_b(w) = \mu_0 \Rightarrow$

$\Rightarrow w \in \mathcal{L}(G)$ (via induction hypothesis).

It follows that

$$\$ \underset{G}{\Rightarrow} a\$b \quad [\text{via } \$ \to a\$b]$$
$$\overset{*}{\underset{G}{\Rightarrow}} awb \quad [\text{via } w \in \mathcal{L}(G)]$$

and therefore $v = awb \in \mathcal{L}(G)$.

Case 2: Assume that $v = bwa$ with $w \in \Sigma^*$. Then

$$\left.\begin{aligned} n_a(w) &= n_a(bwa) - 1 = (\mu_0+1) - 1 = \mu_0 \\ n_b(w) &= n_b(bwa) - 1 = (\mu_0+1) - 1 = \mu_0 \end{aligned}\right\} \Rightarrow$$

$\Rightarrow n_a(w) = n_b(w) = \mu_0 \Rightarrow$

$\rightarrow w \in \mathcal{L}(G)$ (via induction hypothesis)

It follows that

$$\$ \underset{G}{\Rightarrow} b\$a \quad [\text{via } \$ \to b\$a]$$
$$\overset{*}{\underset{G}{\Rightarrow}} bwa \quad [\text{via } w \in \mathcal{L}(G)]$$

and therefore $v = bwa \in \mathcal{L}(G)$.

Case 3: Assume that with no loss of generality

$v = awa$ with $w \in \Sigma^*$

We claim that $\exists p, q \in \Sigma^* : (w = pq \wedge ap \in \mathcal{L}(G) \wedge qa \in \mathcal{L}(G))$

Define $\forall n \in [2\mu_0 + 2] : \Delta(n) = n_a(v_1 v_2 \cdots v_n) - n_b(v_1 v_2 \cdots v_n)$

We have:

$\Delta(1) = n_a(v_1) - n_b(v_1) = n_a(a) - n_b(b) = 1 - 0 = 1 > 0$ (1)

and

$$\begin{aligned}\Delta(2\mu_0+1) &= n_a(v_1 v_2 \cdots v_{2\mu_0+1}) - n_b(v_1 v_2 \cdots v_{2\mu_0+1}) = \\ &= [n_a(v) - n_a(a)] - [n_b(v) - n_b(a)] = \\ &= [n_a(v) - n_b(v)] - [n_a(a) - n_b(a)] = \\ &= 0 - [1 - 0] = -1 < 0 \qquad (2)\end{aligned}$$

From Eq.(1) and Eq.(2):

$\exists m \in [2\mu_0+2] : \Delta(m) = 0$

Choose an $m \in [2\mu_0+2]$ such that $\Delta(m) = 0$ and define $p, q \in \Sigma^*$ such that

$ap = v_1 v_2 \cdots v_m$

$qa = v_{m+1} v_{m+2} \cdots v_{2\mu_0+2}$.

and note that $v = apqa$. Then:

$$\begin{aligned}n_a(ap) - n_b(ap) &= n_a(v_1 v_2 \cdots v_m) - n_b(v_1 v_2 \cdots v_m) \\ &= \Delta(m) = 0 \Rightarrow\end{aligned}$$

$\Rightarrow n_a(ap) = n_b(ap) \Rightarrow ap \in \mathcal{L}(G)$. [via induction hypothesis]

and

$$\begin{aligned}n_a(qa) - n_b(qa) &= [n_a(apqa) - n_a(ap)] - [n_b(apqa) - n_b(ap)] \\ &= [n_a(v) - n_b(v)] - [n_a(ap) - n_b(ap)] \\ &= 0 - 0 = 0 \Rightarrow\end{aligned}$$

$\Rightarrow n_a(qa) = n_b(qa) \Rightarrow qa \in \mathcal{L}(G)$ [via induction hypothesis]

This argument proves the claim. It follows that

$$\begin{aligned}(\$ &\underset{G}{\Rightarrow} \$\$ && \text{[via } \$ \to \$\$\text{]} \\ &\underset{G}{\overset{*}{\Rightarrow}} ap\$ && \text{[via } ap \in \mathcal{L}(G)\text{]} \\ &\underset{G}{\overset{*}{\Rightarrow}} apqa && \text{[via } qa \in \mathcal{L}(G)\text{]} \\ &= v) \Rightarrow (\$ \underset{G}{\overset{*}{\Rightarrow}} v) \Rightarrow v \in \mathcal{L}(G)\end{aligned}$$

We have thus shown that

$\forall \mu \in \mathbb{N} : \forall v \in \Sigma^* : (n_a(v) = n_b(v) = \mu \Rightarrow v \in \mathcal{L}(G))$

Let $v \in \{u \in \Sigma^* \mid n_a(u) = n_b(u)\}$ be given. Then

$$v \in \{u \in \Sigma^* \mid n_a(u) = n_b(u)\} \Rightarrow \begin{cases} v \in \Sigma^* \\ n_a(v) = n_b(v) \in \mathbb{N} \end{cases} \Rightarrow$$

$$\Rightarrow v \in \mathcal{L}(G).$$

From the above argument we have:

$$\begin{cases} \forall v \in \mathcal{L}(G) : v \in \{u \in \Sigma^* \mid n_a(u) = n_b(u)\} \\ \forall v \in \{u \in \Sigma^* \mid n_a(u) = n_b(u)\} : v \in \mathcal{L}(G) \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} \mathcal{L}(G) \subseteq \{u \in \Sigma^* \mid n_a(u) = n_b(u)\} \\ \{u \in \Sigma^* \mid n_a(u) = n_b(u)\} \subseteq \mathcal{L}(G) \end{cases} \Rightarrow$$

$$\Rightarrow \mathcal{L}(G) = \{u \in \Sigma^* \mid n_a(u) = n_b(u)\}$$

EXERCISES

(8) Consider a grammar $G = (V, \Sigma, \$, P)$ with $V = \{\$, A\}$ and $\Sigma = \{a, b\}$ and production rules

$$\$ \longrightarrow aA \mid \lambda$$
$$A \longrightarrow b\$$$

Show that $\mathcal{L}(G) = \{(ab)^n \mid n \in \mathbb{N}\}$

(9) Consider a grammar $G = (V, \Sigma, \$, P)$ with $V = \{\$, A, B\}$ and $\Sigma = \{a\}$ and production rules

$$\$ \longrightarrow Aa$$
$$A \longrightarrow Ba$$
$$B \longrightarrow Aa \mid \lambda$$

Show that $\mathcal{L}(G) = \{a^n \mid n \in \mathbb{N} \wedge n \geq 2\}$

(10) Consider the grammar $G = (V, \Sigma, \$, P)$ with $V = \{\$\}$ and $\Sigma = \{a, b\}$ and production rules

$$\$ \longrightarrow a\$a \mid b\$b \mid \lambda$$

Show that $\mathcal{L}(G) = \{w^R w \mid w \in \Sigma^*\}$.

(11) Consider the grammar $G = (V, \Sigma, \$, P)$ with $V = \{\$, A\}$ and $\Sigma = \{a, b\}$ and production rules

$$\$ \longrightarrow a\$b \mid A$$
$$A \longrightarrow Ab \mid \lambda$$

Show that $\mathcal{L}(G) = \{a^n b^m \mid n, m \in \mathbb{N} \wedge m > n \geq 1\}$.

(12) Consider the grammar $G = (V, \Sigma, S, P)$ with $V = \{S\}$ and $\Sigma = \{a, b\}$ and production rules

$$S \longrightarrow aSbb \mid \lambda$$

Show that $\mathcal{L}(G) = \{ a^n b^{2n} \mid n \in \mathbb{N} \}$.

(13) Consider the grammar $G = (V, \Sigma, S, P)$ with $V = \{S, A\}$ and $\Sigma = \{a, b\}$ and production rules

$$S \longrightarrow aaA$$
$$A \longrightarrow aBb$$
$$B \longrightarrow aBb \mid \lambda$$

Show that $\mathcal{L}(G) = \{ a^{n+2} b^n \mid n \in \mathbb{N} \wedge n \geq 1 \}$.

(14) Consider two grammars $G_1 = (V_1, \Sigma, S_1, P_1)$ and $G_2 = (V_2, \Sigma, S_2, P_2)$ that share the same alphabet and assume that $V_1 \cap V_2 = \emptyset$.

a) Let $G = (V, \Sigma, S, P)$ with $V = \{S\} \cup V_1 \cup V_2$ and $P = \{ S \longrightarrow S_1 S_2 \} \cup P_1 \cup P_2$. Show that $\mathcal{L}(G) = \mathcal{L}(G_1)\mathcal{L}(G_2)$.

b) Let $G = (V, \Sigma, S, P)$ with $V = \{S\} \cup V_1 \cup V_2$ and $P = \{ S \longrightarrow S_1, S \longrightarrow S_2 \} \cup P_1 \cup P_2$. Show that $\mathcal{L}(G) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$.

(15) Consider the grammar $G = (V, \Sigma, S, P)$ with $V = \{S, A\}$ and $\Sigma = \{a, b\}$ and production rules

$$S \longrightarrow aSaSb \mid bSaSa \mid aSbSa \mid SS \mid \lambda$$

Show that $\mathcal{L}(G) = \{ w \in \Sigma^* \mid n_a(w) = 2n_b(w) \}$.

Deterministic Finite Accepters

The deterministic finite accepter (dfa) is the mathematical model of a basic computational device that can accept or reject strings defined over an alphabet Σ

Def: A deterministic finite accepter (dfa) M is defined as the 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where
a) Q is a finite set of internal states
b) Σ is the input alphabet
c) $\delta : Q \times \Sigma \longrightarrow Q$ is a transition function
d) $q_0 \in Q$ is the initial state
e) $F \subseteq Q$ is a set of final states.

notation: The set of all dfa machines that can be defined over a given alphabet Σ is denote as $dfa(\Sigma)$.

The computational action of a dfa M can be defined via the extended transition function as follows:

Def: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a dfa. The extended transition function $\delta^* : Q \times \Sigma^* \longrightarrow Q$ is defined recursively as follows:

$\forall q \in Q : \delta^*(q, \lambda) = q$

$\forall q \in Q : \forall u \in \Sigma^* : \forall a \in \Sigma : \delta^*(q, ua) = \delta(\delta^*(q, u), a)$

interpretation : To evaluate $\delta^*(q,u)$ with $q \in Q$ and $u \in \Sigma^*$, we begin with internal state q and process the string u character by character. For each processed character $a \in \Sigma$, the next state is given by $\delta(q,a)$. As we consume the string u, the dfa transitions from state to state. The resulting sequence of states is given by:

$$\begin{cases} q_1 = q \\ \forall n \in [|u|-1] : q_{n+1} = \delta(q_n, u_n) \end{cases}$$

The final state $q_{|u|}$ is returned by $\delta^*(q,u)$.

▸ Language accepted by a dfa

Def : Let $M = (Q, \Sigma, \delta, q_0, F)$ be a dfa. The language $\mathcal{L}(M)$ accepted by the dfa M is defined as

$$\mathcal{L}(M) = \{u \in \Sigma^* \mid \delta^*(q_0,u) \in F\}$$

interpretation: The belonging condition for $\mathcal{L}(M)$ is given by:

$$\forall u \in \Sigma^* : (u \in \mathcal{L}(M) \iff \delta^*(q_0,u) \in F)$$

This means that a string u is accepted by the dfa M if and only if, initializing M at the initial state q_0 and processing the string u results in a state $\delta^*(q_0,u)$ which is among the permitted final internal states in F. If the resulting state $\delta^*(q_0,u)$ is not among the states in F, then M rejects the string u.

Def: Let $L \in \mathcal{P}(\Sigma^*)$ be a language on Σ. We say that:

L regular $\Leftrightarrow \exists M \in \text{dfa}(\Sigma): L = \mathcal{L}(M)$

▶ Graph representation of a dfa

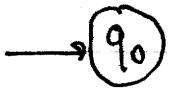
Consider a dfa $M = (Q, \Sigma, \delta, q_0, F)$. We can represent M with a directed graph $G = \text{Graph}(M) = (V(G), E(G), \psi_G)$ such that:

a) The set of vertices is $V(G) = Q$. The final states in $F \subseteq Q$ are represented by specially labeled vertices, as shown in the example below:

for $q \notin F$

for $q \in F$

The initial vertex q_0 is indicated with an open-ended incoming arrow:



for initial state q_0

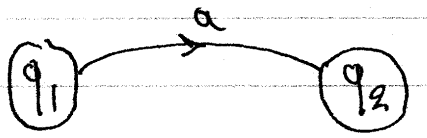
b) The set of edges is

$E(G) = Q \times \Sigma = \{(q,a) \mid q \in Q \wedge a \in \Sigma\}$

and the corresponding incidence function $\psi_G: E(G) \to V(G) \times V(G)$ is given by

$\forall q \in Q: \forall a \in \Sigma: \psi_G((q,a)) = (q, \delta(q,a))$

In other words, every transition rule $\delta(q_1, a) = q_2$ defines an edge $e = (q_1, a)$ from vertex q_1 to q_2:



The outgoing edges from q_1 are distinguished by $a \in \Sigma$, so by convention each outgoing edge is distinguished by the character $a \in \Sigma$.

EXAMPLE - ILLUSTRATION

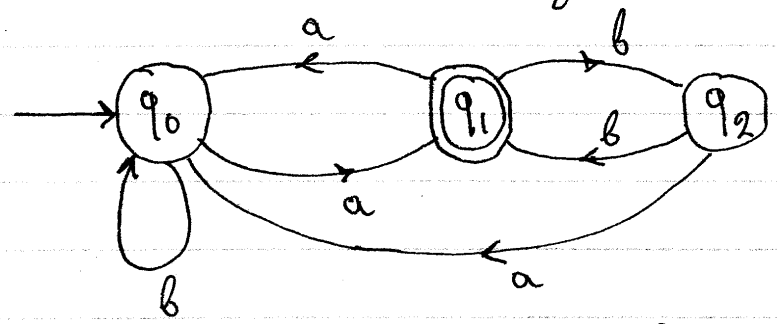
Consider the deterministic finite accepter $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1, q_2\}$ and $\Sigma = \{a, b\}$ and $F = \{q_1\}$ with transition function δ given by:

$\delta(q_0, a) = q_1$ $\quad$ $\delta(q_1, b) = q_2$

$\delta(q_0, b) = q_0$ $\quad$ $\delta(q_2, a) = q_0$

$\delta(q_1, a) = q_0$ $\quad$ $\delta(q_2, b) = q_1$

• This dfa is represented by the following graph:



• To show that $u = abbaa \in L(M)$, we argue as follows:

$$\delta(q_0,a)=q_1 \Rightarrow \delta^*(q_0,ab)=\delta(q_1,b)=q_2 \Rightarrow$$
$$\Rightarrow \delta^*(q_0,abb)=\delta(q_2,b)=q_1 \Rightarrow$$
$$\Rightarrow \delta^*(q_0,abba)=\delta(q_1,a)=q_0 \Rightarrow$$
$$\Rightarrow \delta^*(q_0,abbaa)=\delta(q_0,a)=q_1 \in F$$
$$\Rightarrow \delta^*(q_0,abbaa) \in F \Rightarrow$$
$$\Rightarrow abbaa \in \mathcal{L}(M)$$

Remarks

a) Note that in order for a graph G, as shown in the above example, to represent a dfa, it is necessary for each vertex to have one outgoing edge for every element of Σ.

b) The string abbaa defines a walk on the above graph given by the following alternating sequence of vertices/edges:

$$w=(q_0,(q_0,a),q_1,(q_1,b),q_2,(q_2,b),q_1,(q_1,a),$$
$$q_0,(q_0,a),q_1)$$

The walk traces out the string $\sigma(w)=abbaa$ and we can say that a string $u \in \Sigma^*$ is accepted by M if and only if there is a walk w from q_0 to an element of F such that $\sigma(w)=u$.

We now restate the above more formally as follows:

Def: Let $G=Graph(M)$ be the directed graph representing the dfa $M \in dfa(\Sigma)$. Let $w \in W(G)$ be a walk on G. The string $\sigma(w) \in \Sigma^*$ induced by the walk w is defined as:
$\forall n \in [|w|]: [\sigma(w)]_n = a \iff \exists q \in Q: e_n(w)=(q,a)$

Remark:
Recall the following notation:
$|w|$ = the length of the walk w (number of edges)
$e_n(w)$ = the n^{th} edge of walk w.
Also recall that for vertices $u, v \in V(G)$, we define
$W(G|u,v)$ = the set of all walks on G from u to v.

Thm: Let G be the directed graph $G = Graph(M)$ representing the dfa $M = (Q, \Sigma, \delta, q_0, F)$. Then:
$\forall u \in \Sigma^* : (u \in \mathcal{L}(M) \iff \exists q \in F : \exists w \in W(G|q_0, q) : \sigma(w) = u$

Equivalently, we can state that
$$\mathcal{L}(M) = \{u \in \Sigma^* \mid \exists w \in \bigcup_{q \in F} W(G|q_0, q) : \sigma(w) = u\}$$

In words: A string $u \in \Sigma^*$ is accepted by the dfa M if and only if there is some walk w from the initial state q_0 to some final state $q \in F$ that induces the string u.

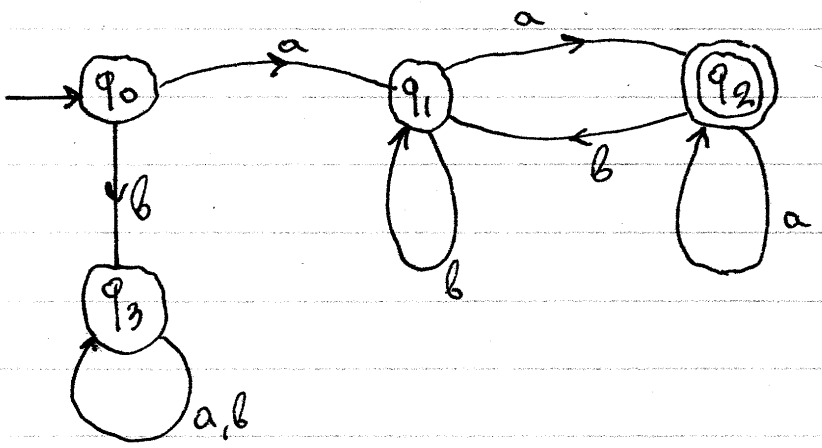
Methodology: The graph terminology is useful in constructing general arguments about the language $\mathcal{L}(M)$ accepted by some dfa $M \in dfa(\Sigma)$, as shown in the following example.

EXAMPLE

Show that the language $L = \{awa \mid w \in \Sigma^*\}$ with $\Sigma = \{a, b\}$ is regular.

Solution

Consider the dfa given by the following graph M:



($\Rightarrow$): Let $u \in \mathcal{L}(M)$ be given. Then:

$$u \in \mathcal{L}(M) \Rightarrow \delta^*(q_0, u) \in F \Rightarrow \delta^*(q_0, u) \in \{q_2\} \Rightarrow$$
$$\Rightarrow \delta^*(q_0, u) = q_2$$

To show that $u \in L$, assume that $u \notin L$. Then:

$$u \notin L \Rightarrow u \notin \{awa \mid w \in \Sigma^*\} \Rightarrow$$
$$\Rightarrow \overline{\exists w \in \Sigma^* : u = awa} \Rightarrow$$
$$\Rightarrow \forall w \in \Sigma^* : u \neq awa$$

Given this restriction, we distinguish between the following cases.

Case 1: Assume that $u = bw$ with $w \in \Sigma^*$. Then:

$$\delta^*(q_0, u) = \delta^*(q_0, bw) = \delta^*(\delta^*(q_0, b), w) =$$
$$= \delta^*(q_3, w) = q_3 \neq q_2$$

which is a contradiction.

Case 2: Assume that $u = awb$ with $w \in \Sigma^*$. Then:

$$\delta^*(q_0, u) = \delta^*(q_0, awb) = \delta^*(q_1, wb) =$$
$$= \delta^*(\delta^*(q_1, w), b) \qquad (1)$$

We distinguish between the following subcases:

- Case 2A: Assume that $\delta^*(q_1, w) = q_1$. Then, from Eq. (1)
$\delta^*(q_0, u) = \delta^*(\delta^*(q_1, w), b) = \delta^*(q_1, b) = q_1 \neq q_2$
which is a contradiction.
- Case 2B: Assume that $\delta^*(q_1, w) = q_2$. Then, from Eq. (1)
$\delta^*(q_0, u) = \delta^*(\delta^*(q_1, w), b) = \delta^*(q_2, b) = q_1 \neq q_2$
which is a contradiction.

Since all possibilities lead to a contradiction, it follows that $u \in L$. We have thus shown that: $\forall u \in \mathcal{L}(M) : u \in L$.

($\Leftarrow$): Let $u \in L$ be given. Then,

$$u \in L \Rightarrow u \in \{awa \mid w \in \Sigma^*\} \Rightarrow$$
$$\Rightarrow \exists w \in \Sigma^* : u = awa$$

Choose an $w \in \Sigma^*$ such that $u = awa$. Note that $\delta(q_0, a) = q_1$. From q_1, there are no walks from q_1 to q_3 and no walks from q_1 to q_0. It follows that

$$\delta^*(q_0, aw) \in \{q_1, q_2\}.$$

We distinguish between the following cases:

Case 1: Assume that $\delta^*(q_0, aw) = q_1$. Then:

$$\delta^*(q_0, awa) = \delta(\delta^*(q_0, aw), a) = \delta(q_1, a) = q_2 \in F \Rightarrow$$
$$\Rightarrow \delta^*(q_0, awa) \in F \Rightarrow u = awa \in \mathcal{L}(M).$$

Case 2: Assume that $\delta^*(q_0, aw) = q_2$. Then,

$\delta^*(q_0, awa) = \delta(\delta^*(q_0, aw), a) = \delta(q_2, a) = q_2 \in F \Rightarrow$

$\Rightarrow \delta^*(q_0, awa) \in F \Rightarrow u = awa \in \mathcal{L}(M)$.

We have thus shown that $\forall u \in L : u \in \mathcal{L}(M)$.

From the above argument:

$$\begin{cases} \forall u \in \mathcal{L}(M) : u \in L \\ \forall u \in L : u \in \mathcal{L}(M) \end{cases} \Rightarrow \begin{cases} \mathcal{L}(M) \subseteq L \\ L \subseteq \mathcal{L}(M) \end{cases} \rightarrow \mathcal{L}(M) = L$$

$\Rightarrow \exists M \in dfa(\Sigma) : L = \mathcal{L}(M)$

$\Rightarrow L$ regular.

▶ Recursion on extended transition function

Thm: Let $M=(Q,\Sigma,\delta,q_0,F)$ be a dfa with extended transition function $\delta^*: Q\times\Sigma^* \to Q$. Then
$\forall q\in Q: \forall u,v\in\Sigma^*: \delta^*(q,uv)=\delta^*(\delta^*(q,u),v)$

Proof

Let $q\in Q$ and let $u,v\in\Sigma^*$ be given. We use induction on the length of v.

For $|v|=0$, we have $v=\lambda$ and therefore

$$\delta^*(q,u\lambda)=\delta^*(q,u)=\delta^*(\delta^*(q,u),\lambda)=\delta^*(\delta^*(q,u),v)$$

Assume that $\forall w\in\Sigma^*: (|w|\le k \Rightarrow \delta^*(q,uw)=\delta^*(\delta^*(q,u),w))$

For $|v|=k+1$, we will show that

$$\delta^*(q,uv)=\delta^*(\delta^*(q,u),v)$$

Choose a $w\in\Sigma^*$ and $a\in\Sigma$ such that $v=wa$. Note that $|w|=|wa|-|a|=(k+1)-1=k$, so the induction hypothesis applies to w. Then

$$\begin{aligned}
\delta^*(q,uv) &= \delta^*(q,uwa) && [\text{via } v=wa]\\
&= \delta(\delta^*(q,uw),a) && [\text{def of } \delta^*]\\
&= \delta(\delta^*(\delta^*(q,u),w),a) && [\text{induction hypothesis}]\\
&= \delta^*(\delta^*(q,u),wa) && [\text{def of } \delta^*]\\
&= \delta^*(\delta^*(q,u),v) && [\text{via } v=wa]
\end{aligned}$$

From the above argument, we have shown that
$\forall q\in Q: \forall u,v\in\Sigma^*: \delta^*(q,uv)=\delta^*(\delta^*(q,u),v)$.

EXERCISES

(16) Let $\Sigma = \{a, b\}$. Construct a dfa M that accepts the following languages L and prove that $L = L(M)$.

a) $L = \{u \in \Sigma^* \mid n_a(u) = 1\}$

b) $L = \{u \in \Sigma^* \mid n_a(u) \geq 1\}$

c) $L = \{u \in \Sigma^* \mid n_a(u) = 2\}$

d) $L = \{uvu \mid u, v \in \Sigma^* \wedge |u| = 2\}$

e) $L = \{a^n \mid n \in \mathbb{N} - \{3\}\}$

f) $L = \{abwb^2 \mid w \in \Sigma^*\}$

g) $L = \{a, aba, b\}$

h) $L = \{ab, abab\}$

(17) Use the identity $\delta^*(q, uv) = \delta^*(\delta^*(q, u), v)$ to show that

a) $\forall L \in \mathcal{P}(\Sigma^*) : (L \text{ regular} \Rightarrow L^2 \text{ regular})$

b) $\forall L_1, L_2 \in \mathcal{P}(\Sigma^*) : \left(\begin{cases} L_1 \text{ regular} \\ L_2 \text{ regular} \end{cases} \Rightarrow L_1 L_2 \text{ regular} \right)$

Pigeonhole principle and non-regular languages

Intuitively, the pigeonhole principle states that if we put n objects (e.g. pigeons) in m boxes (e.g. pigeonholes) and if $n > m$, then at least one box has at least two objects in it. The principle still applies when m, n are infinite cardinalities and can be stated rigorously as follows:

Lemma (Pigeonhole principle)
Let A, B be two sets. Then

$$\begin{cases} f \in \mathrm{Map}(A,B) \Rightarrow \exists x,y \in A : (x \neq y \wedge f(x) = f(y)) \\ A > B \end{cases}$$

Proof

Assume that $f \in \mathrm{Map}(A,B)$ and $A > B$. To show a contradiction, assume that $\overline{\exists x,y \in A : (x \neq y \wedge f(x) = f(y))}$. Then,

$$\begin{aligned} \overline{\exists x,y \in A : (x \neq y \wedge f(x) = f(y))} &\Rightarrow \forall x,y \in A : (f(x) \neq f(y) \vee x = y) \\ &\Rightarrow \forall x,y \in A : (f(x) = f(y) \Rightarrow x = y) \quad [\text{via } p \Rightarrow q \equiv \bar{p} \vee q] \\ &\Rightarrow f \text{ one-to-one} \Rightarrow A \leq B \end{aligned}$$

and therefore:

$$\begin{aligned} A > B \wedge A \leq B &\Rightarrow (A > B \vee A \sim B) \wedge A \leq B \quad [\text{Extension}] \\ &\Rightarrow A \geq B \wedge A \leq B \quad [\text{Definition}] \\ &\Rightarrow A \sim B \quad [\text{Schröder–Bernstein}] \end{aligned}$$

On the other hand, by definition: $A > B \Rightarrow A \not\sim B$, so we have a contradiction. It follows that

$$\exists x,y \in A : (x \neq y \wedge f(x) = f(y)).$$

EXAMPLE

Show that the language $L=\{a^n b^n \mid n\in\mathbb{N}\}$ is not regular.

Solution

To show that L is not regular, assume that L is regular in order to derive a contradiction. Choose $M\in dfa(\{a,b\})$ such that $\mathcal{L}(M)=L$ with $M=(Q,\{a,b\},\delta,q_0,F)$.

Define $f:\mathbb{N}^*\longrightarrow Q$ given by:

$\forall n\in\mathbb{N}^* : f(n)=\delta^*(q_0,a^n)$

From the pigeonhole principle,

$\begin{cases}\mathbb{N}^* \text{ countably infinite} \Rightarrow \mathbb{N}^* > Q \Rightarrow \\ Q \text{ finite}\end{cases}$

$\Rightarrow \exists n_1,n_2\in\mathbb{N}^* : (n_1\neq n_2 \wedge f(n_1)=f(n_2))$

Choose $n_1=\kappa$ and $n_2=\mu$ such that $\kappa\neq\mu$ and $f(\kappa)=f(\mu)=q\in Q$. We also note that

$a^\kappa b^\kappa \in L \Rightarrow \delta^*(q_0,a^\kappa b^\kappa)\in F$

Define $q_f=\delta^*(a^\kappa b^\kappa,q_0)$. It follows that

$\delta^*(q_0,a^\mu b^\kappa)=\delta^*(\delta^*(q_0,a^\mu),b^\kappa)=$

$=\delta^*(f(\mu),b^\kappa)=$

$=\delta^*(f(\kappa),b^\kappa)=$

$=\delta^*(\delta^*(q_0,a^\kappa),b^\kappa)$

$=\delta^*(q_0,a^\kappa b^\kappa)$

$=q_f\in F \Rightarrow$

$\Rightarrow \delta^*(q_0,a^\mu b^\kappa)\in F \Rightarrow a^\mu b^\kappa\in L \Rightarrow$

$\Rightarrow a^\mu b^\kappa \in \{a^n b^n \mid n\in\mathbb{N}\} \Rightarrow \underline{\mu=\kappa}.$ ← Contradiction

It follows that L is not regular.

Lemma: (Pumping Lemma)

Let Σ be an alphabet and let $L \in \mathcal{P}(\Sigma^*)$ be a language. Then:

$$L \text{ regular} \wedge L \text{ infinite} \Rightarrow$$

$$\Rightarrow \exists m \in \mathbb{N}^* : \forall w \in L : \left(|w| \geq m \Rightarrow \exists x,y,z \in \Sigma^* : \begin{cases} xyz = w \\ |xy| \leq m \\ |y| \geq 1 \\ \forall k \in \mathbb{N} : xy^k z \in L \end{cases} \right)$$

Proof

Assume that L regular $\wedge$ L infinite. Choose $M \in dfa(\Sigma)$ such that $L = \mathcal{L}(M)$. Let $n = |Q - \{q_0\}|$ be the number of non-initial internal states of M and write

$$Q = \{q_0, q_1, q_2, \dots, q_n\}$$

Choose $m = n+1 \in \mathbb{N}^*$. Let $w \in L$ be given and assume that $|w| \geq m$. Define $l = |w|$.

▸ Construction of x,y,z

Define $p_0, p_1, p_2, \dots, p_l$ according to

$$\begin{cases} p_0 = q_0 \\ \forall k \in [l] : p_k = \delta(p_{k-1}, w_k) \end{cases}$$

and note that $p_0, p_1, \dots, p_l$ are the internal states that the machine M goes through as it processes the string w. Since $w \in L$, we have $p_l = \delta(q_0, w) = q_f \in F$.

From the pigeonhole principle it follows that

$$\exists a,b \in [n+1] : (a > b \wedge p_a = p_b)$$

Choose $a,b \in [n+1]$ such that $a > b$ and $p_a = p_b$.

Define $x, y, z \in \Sigma^*$ such that

$$\begin{cases} x = w_1 w_2 \cdots w_a \\ y = w_{a+1} w_{a+2} \cdots w_b \\ z = w_{b+1} w_{b+2} \cdots w_l \end{cases}$$

and note that

$$\begin{cases} \delta^*(q_0, x) = p_a \\ \delta^*(q_0, xy) = p_b = p_a \\ \delta^*(p_b, z) = q_f \in F \end{cases}$$

▶ Proof of claims

We have

$|xy| = |x| + |y| = (a-1+1) + (b-(a+1)+1) =$

$= a + (b-a-1+1) = a + (b-a) = b \leq n+1$ (via $b \in [n+1]$)

and

$|y| = b-(a+1)+1 = b-a-1+1 = b-a > 0$ (via $b > 0$)

$\Rightarrow |y| > 0 \Rightarrow |y| \geq 1$.

We claim that $\forall k \in \mathbb{N} : \delta^*(q_0, xy^k) = p_b$.

For $k=0$:

$\delta^*(q_0, xy^k) = \delta^*(q_0, xy^0) = \delta^*(q_0, x) = p_a = p_b$.

For $k = k_0$, assume that $\delta^*(q_0, xy^{k_0}) = p_b$.

For $k = k_0+1$, we have

$\delta^*(q_0, xy^{k_0+1}) = \delta^*(\delta^*(q_0, xy^{k_0}), y) =$ [Definition]

$= \delta(p_b, y) =$ [induction hyp]

$= \delta(p_a, y) =$ [via $p_a = p_b$]

$= p_b$

We have thus shown the claim.

Let $\underline{k \in \mathbb{N}}$ be given. Then:

$$\delta^*(q_0, xy^k z) = \delta^*(\delta^*(q_0, xy^k), z)$$
$$= \delta^*(p_6, z)$$
$$= q_f \in F \Rightarrow$$

$$\rightarrow \underline{xy^k z \in L}$$

and therefore: $\forall k \in \mathbb{N} : xy^k z \in L$

From the above argument we obtain the lemma.

EXAMPLES

a) Let $\Sigma = \{a, b\}$. Show that

$L = \{ww^R \mid w \in \Sigma^*\}$

is not regular, via the pumping lemma

Solution

To show that L is not regular, assume that L is regular.

We also note that

$\forall n \in \mathbb{N}^*: a^{2n} = a^n a^n = a^n (a^n)^R \Rightarrow$

$\Rightarrow \forall n \in \mathbb{N}^*: a^{2n} \in L$

$\Rightarrow$ L infinite.

It follows that the pumping lemma applies. Choose $m \in \mathbb{N}^*$ such that

$$\forall w \in L: \left(|w| \geq m \Rightarrow \exists x, y, z \in L: \begin{cases} |xy| \leq m \wedge |y| \geq 1 \\ \forall k \in \mathbb{N}: xy^k z \in L \end{cases} \right)$$

Let $w = a^m b^{2m} a^m = a^m b^m b^m a^m = (a^m b^m)(a^m b^m)^R \in L$

and note that $|w| = m + 2m + m = 4m \geq m$. Choose

$x, y, z \in L$ such that $|xy| \leq m$ and $|y| \geq 1$ and

$\forall k \in \mathbb{N}: xy^k z \in L$.

Define $l_1 = |x|$ and $l_2 = |y|$. Since

$l_1 + l_2 = |x| + |y| = |xy| \leq m$

it follows that $x = a^{l_1}$ and $y = a^{l_2}$. Then:

$xyz = w \Leftrightarrow a^{l_1} a^{l_2} z = a^m b^{2m} a^m \Leftrightarrow$

$\Leftrightarrow z = a^{m - l_1 - l_2} b^{2m} a^m$

For $k = 2$: $xy^2 z \in L$. Note that:

$$xy^2z = a^{l_1}(a^{l_2})^2\left[a^{m-l_1-l_2}b^{2m}a^m\right] =$$
$$= a^{l_1+2l_2+m-l_1-l_2}b^{2m}a^m$$
$$= a^{m+l_2}b^m b^m a^m \in L \Rightarrow$$
$$\Rightarrow m+l_2 = m \Rightarrow l_2 = 0$$

which is a contradiction since $l_2 = |y| \geq 1$

It follows that L is not regular. □

b) Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid n_a(w) < n_b(w)\}$.
Show that L is not regular.

Solution

To show that L is not regular, assume that L is regular. We also note that

$\forall n \in \mathbb{N}^* : (n_a(b^n) = 0 < n = n_b(b^n)) \Rightarrow$

$\Rightarrow \forall n \in \mathbb{N}^* : (b^n \in L) \Rightarrow L$ infinite.

It follows that the pumping lemma applies.

Choose $m \in \mathbb{N}^*$ such that

$$\forall w \in L : \left(|w| \geq m \Rightarrow \exists x, y, z \in \Sigma^* : \begin{cases} w = xyz \wedge |y| \geq 1 \wedge |xy| \leq m \\ \forall n \in \mathbb{N} : xy^n z \in L \end{cases} \right)$$

Choose $w = a^n b^{n+1} \in L$ with $n > m$, and note that

$|w| = |a^n b^{n+1}| = n + (n+1) = 2n+1 > 2n > n > m \Rightarrow |w| \geq m$.

Choose $x, y, z \in \Sigma^*$ such that $w = xyz$ and $|y| \geq 1$ and $|xy| \leq m$ and $\forall k \in \mathbb{N} : xy^k z \in L$.

Define $l_1 = |x|$ and $l_2 = |y|$. Since $l_1 + l_2 = |x| + |y| = |xy| \leq m < n$, it follo

it follows that $x = a^{l_1}$ and $y = a^{l_2}$ and

$xyz = a^n b^{n+1} \Leftrightarrow a^{l_1} a^{l_2} z = a^n b^{n+1} \Leftrightarrow$

$\Leftrightarrow z = a^{n - l_1 - l_2} b^{n+1}$

We then have:

$xy^2 z = a^{l_1} (a^{l_2})^2 [a^{n - l_1 - l_2} b^{n+1}] =$

$= a^{l_1 + 2l_2 + n - l_1 - l_2} b^{n+1} = a^{n + l_2} b^{n+1} \in L \Rightarrow$

$\Rightarrow n_a(a^{n+l_2}b^{n+1}) < n_b(a^{n+l_2}b^{n+1}) \Rightarrow$

$\Rightarrow n+l_2 < n+1 \Rightarrow l_2 < 1 \Rightarrow |y| < 1$

which is a contradiction since $|y| \geq 1$.

We conclude that L is not regular.

EXERCISES

(18) Use the pigeonhole principle to show that the following languages are not regular.

a) $L = \{a^n b^{2n} \mid n \in \mathbb{N}\}$

b) $L = \{a^{2n} b^{3n+1} \mid n \in \mathbb{N}\}$

c) $L = \{(ab)^n \cdot a^k \mid n, k \in \mathbb{N} \wedge n > k\}$

(19) Use the pumping lemma to show that the following languages are not regular.

a) $L = \{a^n b^n \mid n \in \mathbb{N}\}$

b) $L = \{x^a y^b \mid a, b \in \mathbb{N} \wedge a < b\}$

c) $L = \{x^a y^b z^c \mid a, b, c \in \mathbb{N} \wedge a + b \leq c\}$

d) $L = \{u \in \{a, b\}^* \mid n_a(u) = n_b(u)\}$

e) $L = \{uu \mid u \in \{a, b\}^+\}$

f) $L = \{x^a y^b x^c \mid a, b, c \in \mathbb{N} \wedge a = c\}$

Non-deterministic finite accepter

The fundamental difference between a non-deterministic finite accepter (hereafter, nfa) and a dfa is that in an nfa, the transition function maps from the current state to a set of possible states, depending on the string input. A string is accepted if a walk from the initial state to some final state exists that traces out the string characters. By contrast, in a dfa, given the current state and string input, there is a unique next state, and a unique walk on the graph representation of the dfa as a function of the string input. In spite of the increased flexibility of the nfa, it is equally powerful to the dfa; a language is accepted by an nfa if and only if it is accepted by some dfa. The formal definitions for these concepts are as follows:

Def: A non-deterministic finite accepter (nfa) M is defined as the 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

a) Q is a set of internal states

b) Σ is the alphabet set

c) $\delta : Q \times (\Sigma \cup \{\lambda\}) \to P(Q)$ is a transition function

d) $q_0 \in Q$ is an initial state

e) $F \subseteq Q$ is a set of final states

notation: The set of all nfas that can be defined over a given alphabet is denoted as $nfa(\Sigma)$.

▸ Graph representation of nfa

A non-deterministic finite accepter can be represented by a directed graph according to the following definition.

Def: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an nfa. We define the directed graph $G = Graph(M)$ with

a) Set of vertices $V(G) = Q$

b) Set of edges $E(G) = \{(q_1, q_2, a) \in Q \times Q \times (\Sigma \cup \{\lambda\}) : q_2 \in \delta(q_1, a)\}$

c) Incidence function $\psi_G : E(G) \to V(G) \times V(G)$ given by $\forall e = (q_1, q_2, a) \in E(G) : \psi_G(e) = (q_1, q_2)$

notation

a) The edge (q_1, q_2, a) represents a transition from q_1 to q_2 upon processing the character a. The character "a" labels the edge.

b) Otherwise, to draw G we use the same conventions used for directed graph representations of dfas.

Remark: Note that, unlike directed graphs representing dfas, directed graphs of nfas can have λ-edges.

Def : Let $G = Graph(M)$ be the directed graph representing the nfa $M = (Q, \Sigma, \delta, q_0, F)$. Let $w \in W(G)$ be a walk on G. The string $\sigma(w) \in (\Sigma \cup \{\lambda\})^*$ induced by the walk w is defined as

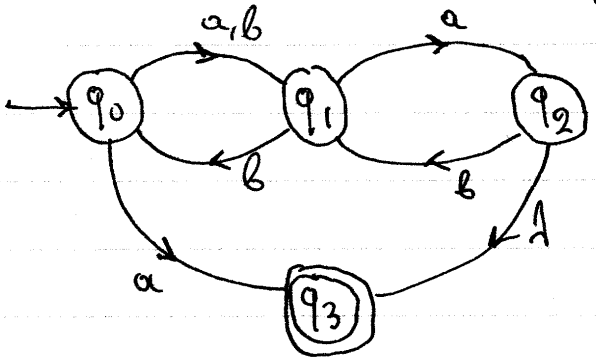
$$\forall n \in [|w|] : ([\sigma(w)]_n = a \iff \exists q_1, q_2 \in Q : e_n(w) = (q_1, q_2, a))$$

Remark : Because $\sigma(w) \in (\Sigma \cup \{\lambda\})^*$ may contain the null symbol λ, it can be simplified to a shorter string $\sigma^*(w) \in \Sigma^*$ by deleting all null symbols.

EXAMPLE

Consider the nfa represented by the following directed graph:

G:



Then $Q = \{q_0, q_1, q_2, q_3\}$ and $\Sigma = \{a, b\}$ and $F = \{q_3\}$ and δ is defined as

$\delta(q_0, \lambda) = \emptyset$ $\quad$ $\delta(q_1, \lambda) = \emptyset$ $\quad$ $\delta(q_2, \lambda) = \{q_3\}$

$\delta(q_0, a) = \{q_1, q_3\}$ $\quad$ $\delta(q_1, a) = \{q_2\}$ $\quad$ $\delta(q_2, a) = \emptyset$

$\delta(q_0, b) = \{q_1\}$ $\quad$ $\delta(q_1, b) = \{q_0\}$ $\quad$ $\delta(q_2, b) = \{q_1\}$

$\delta(q_3, \lambda) = \emptyset$ $\quad$ $\delta(q_3, a) = \emptyset$ $\quad$ $\delta(q_3, b) = \emptyset$

Consider the walk w given by

$w = (q_0, (q_0, q_1, b), q_1, (q_1, q_2, a), q_2, (q_2, q_3, \lambda), q_3)$

Then $\sigma(w) = bad$ and $\sigma^*(w) = ba$.

▸ The extended transition function

A unique feature of nfas is that the transition function δ allows transitions between internal states via the null string λ. This complicates the definition of the extended transition function δ^*. Given a string $u \in \Sigma^*$ and an initial internal state $q \in Q$, we wish to define $\delta^*(q, u)$ as the set of all states that can be reached from q following either null edges or edges that follow and consume the string u character by character. Because the nfa is non-deterministic, it is possible that from some state in Q, there are multiple possible next states corresponding to a given character in Σ. The first step is to define a mapping $\Lambda : \mathcal{P}(Q) \to \mathcal{P}(Q)$ that gives the set of all states that can be reached from some set of states $P \in \mathcal{P}(Q)$ following a finite sequence of null edges. Then the Λ function can be used to define the extended transition function δ^*. Both functions are defined recursively as follows:

Def: Let $M=(Q,\Sigma,\delta,q_0,F)$ be an nfa. We define:

a) The λ-closure function $\Lambda:\mathcal{P}(Q)\to\mathcal{P}(Q)$ is defined recursively as follows:

$$\begin{cases} \forall P\in\mathcal{P}(Q): \Lambda_0(P)=\{q\in Q \mid \exists q_1\in P: q\in\delta(q_1,\lambda)\} \\ \forall n\in\mathbb{N}^*: \forall P\in\mathcal{P}(Q): \Lambda_n(P)=\Lambda_0(\Lambda_{n-1}(P)) \\ \forall P\in\mathcal{P}(Q): \Lambda(P)=\bigcup_{n\in\mathbb{N}}\Lambda_n(P) \end{cases}$$

b) The extended transition function $\delta^*:Q\times\Sigma^*\to\mathcal{P}(Q)$ is also defined recursively as follows:

$$\begin{cases} \forall q\in Q: \delta^*(q,\lambda)=\Lambda(\{q\}) \\ \forall q\in Q: \forall u\in\Sigma^*: \forall a\in\Sigma: \delta^*(q,ua)=\Lambda\left(\bigcup_{p\in\delta^*(q,u)}\delta(p,a)\right) \end{cases}$$

▶ Language accepted by an nfa

Def: Let $M=(Q,\Sigma,\delta,q_0,F)$ be an nfa with extended transition function $\delta^*:Q\times\Sigma^*\to\mathcal{P}(Q)$. The language $\mathcal{L}(M)$ accepted by the nfa M is:

$$\mathcal{L}(M)=\{u\in\Sigma^* \mid \delta^*(q_0,u)\cap F\neq\emptyset\}$$

Remark: An equivalent way to define the language $\mathcal{L}(M)$ accepted by M is via the graph representation $G=\text{Graph}(M)$ of M. Intuitively, a string $u\in\Sigma^*$ is accepted by M, if and only if there is a walk from q_0 to some final state $q\in F$ such that $\sigma^*(w)=u$.

Using quantifiers, we write:

$$u \in \mathcal{L}(M) \Leftrightarrow \exists q \in F : \exists w \in W(\text{Graph}(M) \mid q_0, q) : \sigma^*(w) = u$$

The belonging condition can be also rewritten using set builder notation as follows:

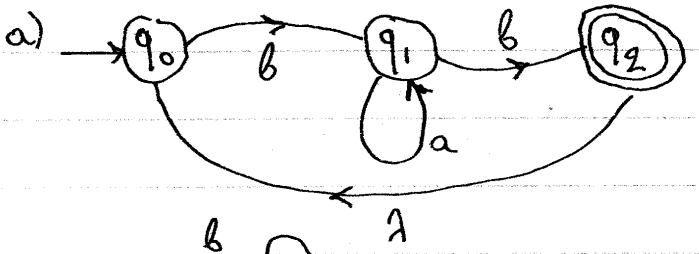
$$\mathcal{L}(M) = \{ u \in \Sigma^* \mid \exists w \in \bigcup_{q \in F} W(\text{Graph}(M) \mid q_0, q) : \sigma^*(w) = u \}$$

$$= \{ \sigma^*(w) \mid w \in \bigcup_{q \in F} W(\text{Graph}(M) \mid q_0, q) \}$$

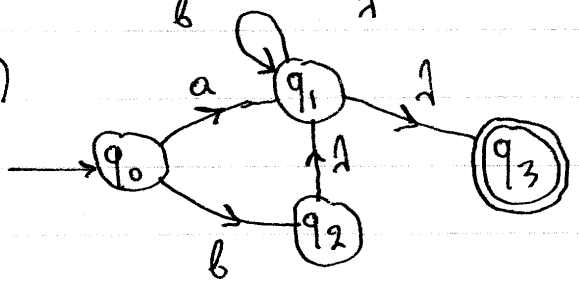
EXERCISES

(20) Write the formal definition for the following non-deterministic finite accepters:

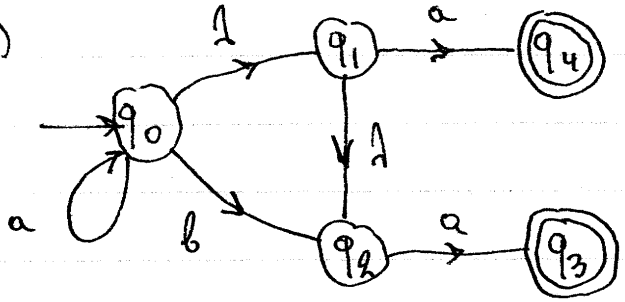
a)



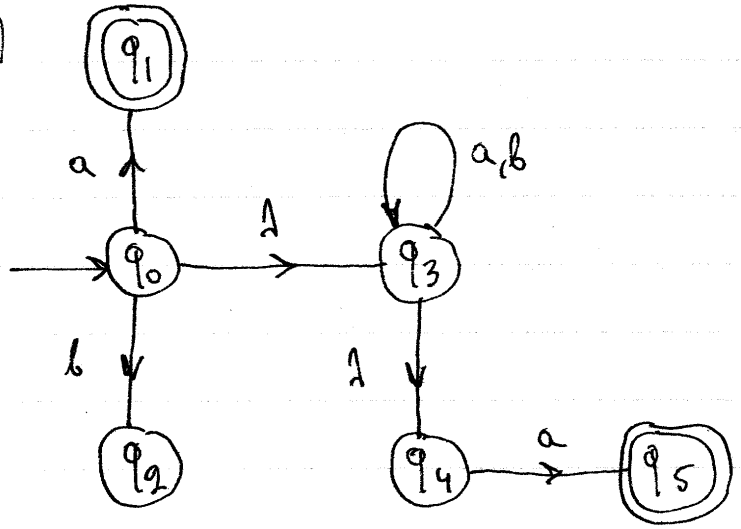
b)



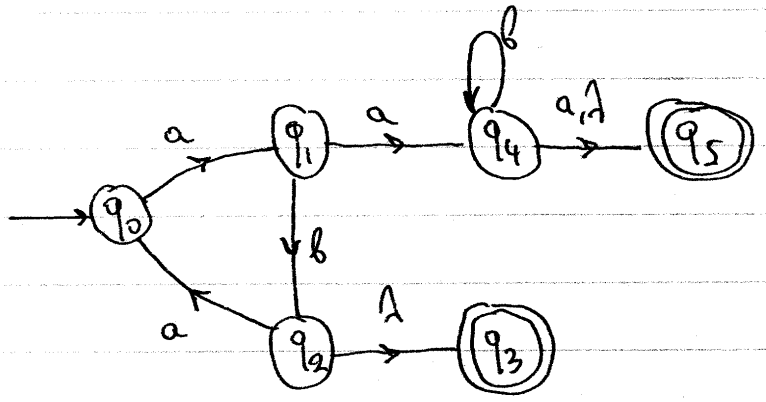
c)



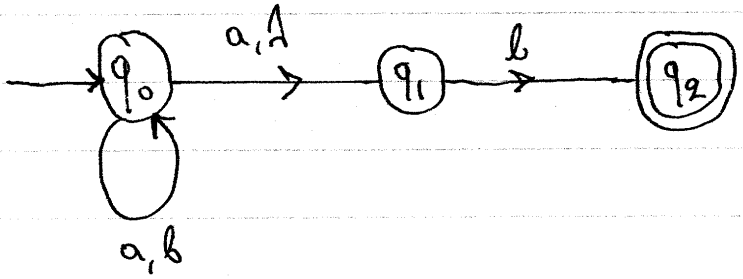
d)



e)



(21) Consider the non-deterministic finite accepter represented by



Evaluate

a) $\delta^*(q_0, aaa)$

b) $\delta^*(q_0, ba^2)$

c) $\delta^*(q_0, a^2bab)$

d) $\delta^*(q_0, a^n b^n)$ for $n \in \mathbb{N}$

e) $\delta^*(q_0, b^n a^n)$ for $n \in \mathbb{N}$

(Hint: For (d), (e) you will need to use method of induction as part of a wider argument)

(22) Given a alphabet Σ, show that for all $M \in nfa(\Sigma)$ there is at least one $M_0 \in nfa(\Sigma)$ that has exactly one final state such that $\mathcal{L}(M) = \mathcal{L}(M_0)$.

(23) Show that all finite languages are regular.
(Hint: Construct an appropriate nfa. Use induction on the cardinality of the language in question)

Equivalence of nfa and dfa

Let Σ be an alphabet. Given a deterministic finite accepter $M_2 \in dfa(\Sigma)$ with $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ we can easily define an equivalent non-deterministic finite accepter $M_1 \in nfa(\Sigma)$ with $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ such that $\mathcal{L}(M_1) = \mathcal{L}(M_2)$. By choosing:

$$\begin{cases} Q_1 = Q_2 \wedge q_{01} = q_{02} \wedge F_1 = F_2 \\ \forall q \in Q_1 : \forall a \in \Sigma : \delta_1(q,a) = \{\delta_2(q,a)\} \\ \forall q \in Q_1 : \delta_1(q,\lambda) = \{q\} \end{cases}$$

We can then claim that

$$\forall M_2 \in dfa(\Sigma) : \exists M_1 \in nfa(\Sigma) : \mathcal{L}(M_1) = \mathcal{L}(M_2)$$

We will now provide an algorithm for converting an nfa to a dfa, which in turn establishes the converse statement:

$$\forall M_1 \in nfa(\Sigma) : \exists M_2 \in dfa(\Sigma) : \mathcal{L}(M_1) = \mathcal{L}(M_2).$$

Algorithm: (nfa to dfa)

Let $M_1 \in nfa(\Sigma)$ be an nfa with $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$
We construct an equivalent dfa $M_2 \in dfa(\Sigma)$ with $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ with $Q_2 \subseteq \mathcal{P}(Q_1)$ as follows:

a) Define $q_{02} = \{q_{01}\}$ (initial state) and define $Q_{2,0} = \{q_{02}\}$ as an initial approximation of Q_2.

b) Starting from $n=0$, assume we have worked our way to $Q_{2,n}$, and have defined δ partially. Find an element

$q \in Q_{2,in}$ and $a \in \Sigma$ for which $\delta_2(q,a)$ is undefined.

Define:

$$\begin{cases} \delta_2(q,a) = \bigcup_{s \in q} \delta_1^*(s,a) \\ Q_{2,in+1} = Q_{2,in} \cup \{\delta_2(q,a)\} \end{cases}$$

c) Repeat the previous step until
$\forall q \in Q_{2,in} : \forall a \in \Sigma : (\delta_2(q,a)$ has been defined)
Then set $Q_2 = Q_{2,in}$ and note that δ_2 is completely defined as well.

d) Define the set of final states F_2 as follows:
$F_2 = \{q \in Q_2 \mid \exists s \in q : s \in F_1\}$

↳ While running the above algorithm, we note that it is possible to find $\delta_2(q,a) = \emptyset$ for some $q \in \mathcal{P}(Q_1)$ and $a \in \Sigma$. Then $\emptyset$ will be a state of the dfa and the above algorithm will then result in a definition of δ_2 such that

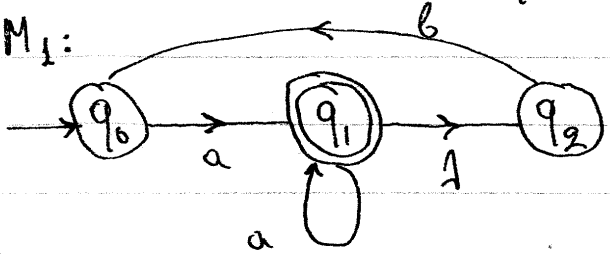
$$\forall a \in \Sigma : \delta_2(\emptyset, a) = \emptyset$$

It follows that if $\emptyset \in Q_2$, then the internal state $\emptyset$ will function as a "trap state" in that it is impossible to transition from $\emptyset$ to other states.

EXAMPLE

Define a dfa that is equivalent to the following nfa:

M_1:



Solution

We begin with $Q_{20} = \{\{q_0\}\}$. Then

$\delta_2(\{q_0\}, a) = \bigcup_{s \in \{q_0\}} \delta_1^*(s, a) = \delta_1^*(\{q_0\}, a) = \{q_1, q_2\}$

Let $Q_{21} = \{\{q_0\}, \{q_1, q_2\}\}$

$\delta_2(\{q_0\}, b) = \bigcup_{s \in \{q_0\}} \delta_1^*(s, b) = \delta_1^*(q_0, b) = \emptyset$

Let $Q_{22} = \{\{q_0\}, \{q_1, q_2\}, \emptyset\}$

$\delta_2(\{q_1, q_2\}, a) = \bigcup_{s \in \{q_1, q_2\}} \delta_1^*(s, a) = \delta_1^*(q_1, a) \cup \delta_1^*(q_2, a)$

$= \{q_1, q_2\} \cup \emptyset = \{q_1, q_2\}.$

$\delta_2(\{q_1, q_2\}, b) = \bigcup_{s \in \{q_1, q_2\}} \delta^*(s, b) =$

$= \delta_1^*(q_1, b) \cup \delta_1^*(q_2, b) =$

$= \{q_0\} \cup \{q_0\} = \{q_0\}$

$\delta_2(\emptyset, a) = \emptyset$

$\delta_2(\emptyset, b) = \emptyset$

At this point the algorithm terminates with

$Q_2 = Q_{q_2} = \{\{q_0\}, \{q_1, q_2\}, \emptyset\}$

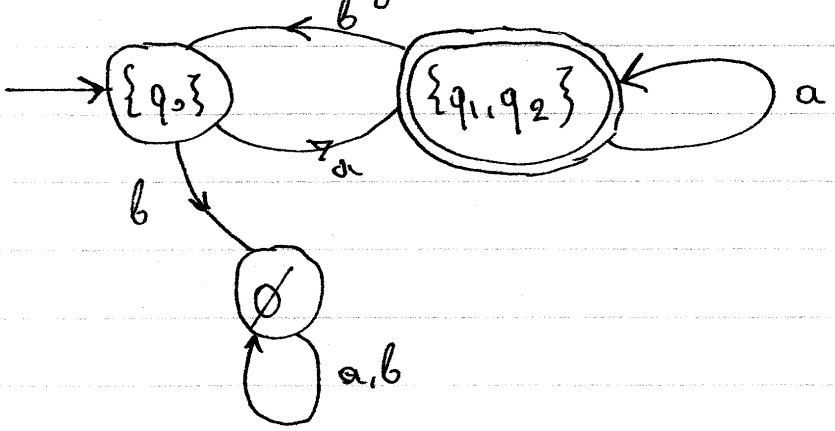
The set of final states is

$F_2 = \{q \in Q_2 \mid \exists \$ \in q : \$ \in F_1\}$

$= \{q \in \{\{q_0\}, \{q_1, q_2\}, \emptyset\} \mid \exists \$ \in q : \$ \in \{q_1\}\} =$

$= \{\{q_1, q_2\}\}$

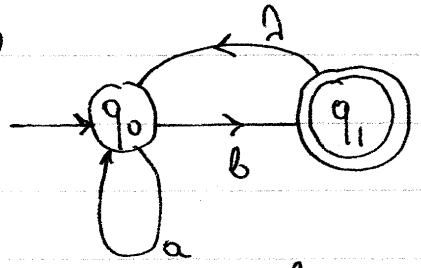
The corresponding dfa M_2 has representation



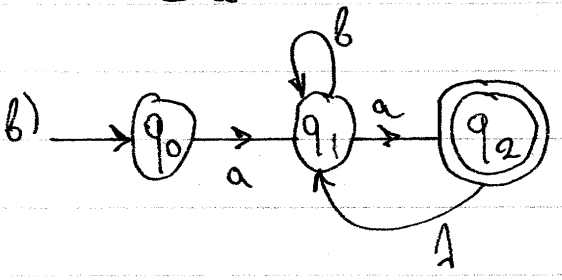
EXERCISES

(24) Define dfas that are equivalent to the following nfas and show the details of the constructions:

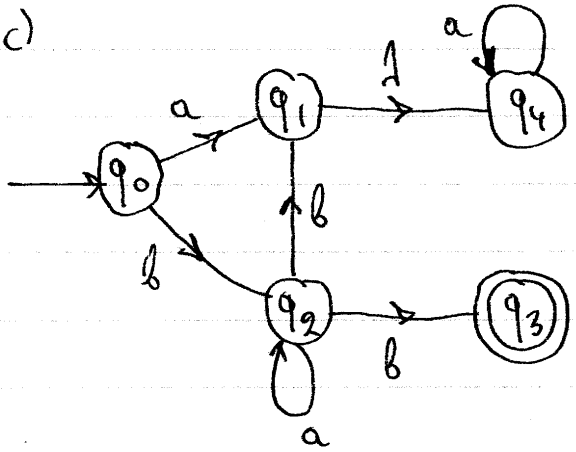
a)



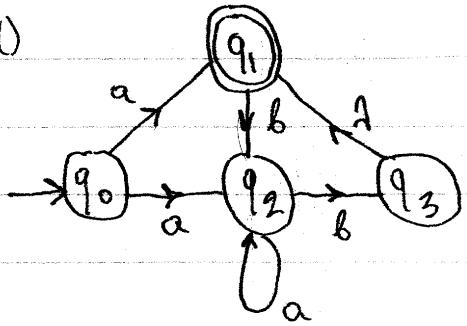
b)



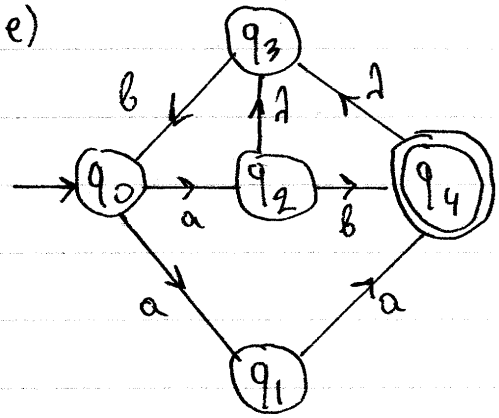
c)



d)



e)



Regular Expressions

Regular expressions can be used to provide a concise representation of regular languages. It is also simple to deduce a corresponding nfa from a regular expression and then convert it to a dfa. We use recursion to define regular expressions and the language induced by a regular expression, as follows:

Def: Let Σ be an alphabet. The set $\text{Reg}(\Sigma)$ of all regular expressions is defined as follows:

a) $\emptyset$, λ, and all $a \in \Sigma$ are regular expressions

b) Given $r_1, r_2 \in \text{Reg}(\Sigma)$, $r_1 \vee r_2$, $r_1 r_2$, r_1^*, (r_1) are also regular expressions

c) We build $\text{Reg}(\Sigma)$ by combining (a) and (b) in a finite number of steps.

A formal definition of the set $\text{Reg}(\Sigma)$ of all regular expressions can be given via a formal grammar:

Def: Let Σ be an alphabet and consider a grammar G with variables $V = \{\$\}$, alphabet $\Sigma \cup \{\emptyset, (,), *\}$ and the following production rules:

$$\begin{cases} \forall a \in \Sigma: \quad \$ \to a \\ \$ \to \emptyset \mid \lambda \mid \$ \vee \$ \mid \$\$ \mid \$^* \mid (\$) \end{cases}$$

Then, we define $\text{Reg}(\Sigma) = \mathcal{L}(G)$.

Example : Given the alphabet $\Sigma = \{a, b\}$, the following strings are possible regular expressions:

$(ab)^*$

$a \vee (b^*)$

$(a^*)(b^*) \vee (ba)^*$

Def : Let Σ be an alphabet and $r \in Reg(\Sigma)$ a regular expression. The language $\mathcal{L}(r)$ defined by the regular expression r is given recursively, according to the following rules:

(a) $\mathcal{L}(\emptyset) = \emptyset$

(b) $\mathcal{L}(\lambda) = \{\lambda\}$

(c) $\forall a \in \Sigma : \mathcal{L}(a) = \{a\}$

(d) $\forall r_1, r_2 \in Reg(\Sigma) : \mathcal{L}(r_1 \vee r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$

(e) $\forall r_1, r_2 \in Reg(\Sigma) : \mathcal{L}(r_1 r_2) = \mathcal{L}(r_1)\mathcal{L}(r_2)$
$= \{uv \mid u \in \mathcal{L}(r_1) \wedge v \in \mathcal{L}(r_2)\}$

(f) $\forall r \in Reg(\Sigma) : \mathcal{L}((r)) = \mathcal{L}(r)$

(g) $\forall r \in Reg(\Sigma) : \mathcal{L}(r^*) = [\mathcal{L}(r)]^* = \bigcup_{n \in \mathbb{N}} [\mathcal{L}(r)]^n$

Def : (Equivalent regular expressions)
Let $u \in \Sigma$ be an alphabet and let $r_1, r_2 \in Reg(\Sigma)$. We say that

$$r_1 \equiv r_2 \Leftrightarrow \mathcal{L}(r_1) = \mathcal{L}(r_2)$$

Remark : To minimize ambiguity in applying the above rules we adopt the following precedence rules:

(a) $*$ takes precedence over all operations

(e.g. $ab* \equiv a(b*)$, $a \vee b* \equiv a \vee (b*)$

(b) Concatenation takes precedence over disjunction

(e.g. $ab \vee c \equiv (ab) \vee c$).

EXAMPLE

a) Define the language defined by the regular expression $r = (ab)^*$

Solution

$$\mathcal{L}(r) = \mathcal{L}((ab)^*) = [\mathcal{L}(ab)]^* = [\mathcal{L}(a)\mathcal{L}(b)]^* =$$
$$= [\{a\}\{b\}]^* = \{ab\}^* = \bigcup_{n\in\mathbb{N}} \{ab\}^n =$$
$$= \{(ab)^n \mid n \in \mathbb{N}\}.$$

b) Define the language defined by the regular expression $r = a^*(a \vee b)$

Solution

$$\mathcal{L}(r) = \mathcal{L}(a^*(a \vee b)) = \mathcal{L}(a^*)\mathcal{L}(a \vee b) = \mathcal{L}(a^*)[\mathcal{L}(a) \cup \mathcal{L}(b)]$$
$$= \{a\}^*[\{a\} \cup \{b\}] = \{a\}^*\{a, b\} =$$
$$= \{a^n \mid n \in \mathbb{N}\}\{a, b\} =$$
$$= \{a^{n+1}, a^n b \mid n \in \mathbb{N}\}.$$

c) Let $L = \{a^{2n} b^{2m+1} \mid n \in \mathbb{N} \wedge m \in \mathbb{N}\}$ be a language. Find a regular expression r that generates the language L.

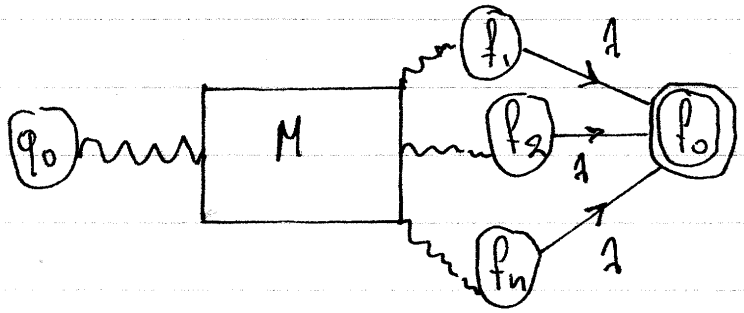
Solution

$$L = \{a^{2n} b^{2m+1} \mid n \in \mathbb{N} \wedge m \in \mathbb{N}\} =$$
$$= \{a^{2n} \mid n \in \mathbb{N}\}\{b^{2m} \mid n \in \mathbb{N}\}\{b\} =$$
$$= \{(aa)^n \mid n \in \mathbb{N}\}\{(bb)^m \mid m \in \mathbb{N}\}\{b\}$$
$$= \{aa\}^*\{bb\}^*\{b\} = [\mathcal{L}(aa)]^*[\mathcal{L}(bb)]^*\mathcal{L}(b)$$
$$= \mathcal{L}((aa)^*)\mathcal{L}((bb)^*)\mathcal{L}(b) = \mathcal{L}((aa)^*(bb)^*b) \Rightarrow$$
$$\Rightarrow r \equiv (aa)^*(bb)^*b.$$

▶ Constructing an nfa from a regular expression

Given a regular expression $r \in Reg(\Sigma)$, the problem is to construct an nfa $M \in nfa(\Sigma)$ such that $\mathcal{L}(r) = \mathcal{L}(M)$.

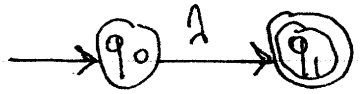
Remark: With no loss of generality we can assume that our nfas have a unique final state. In any other nfa with multiple final states $f_1, f_2, \ldots, f_n \in F$, we can simply create a new final state f_0 and connect it with $f_1, f_2, \ldots, f_n$ using λ-edges:



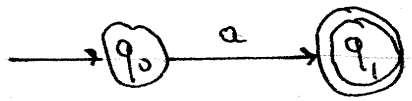
Algorithm: The needed nfa can be constructed recursively as follows:

a) For $r = \emptyset$, the corresponding nfa is:

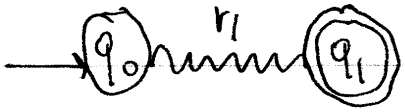
b) For $r = \lambda$, the corresponding nfa is



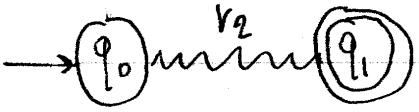
c) For $r = a$ with $a \in \Sigma$, the corresponding nfa is:



d) Recursive cases: Let us assume that we have already constructed the following nfas for the regular expressions $r_1, r_2 \in Reg(\Sigma)$:



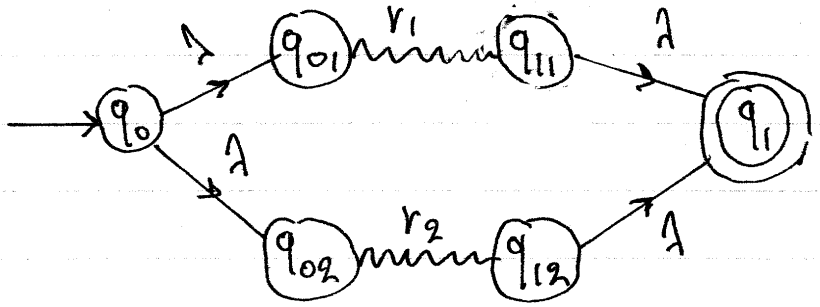
← nfa for $\mathcal{L}(r_1)$



← nfa for $\mathcal{L}(r_2)$

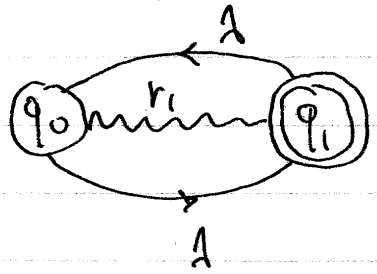
We may then construct rules for $r_1 \vee r_2$, $r_1 r_2$, r_1^* as follows:

▸ 1) For $r_1 \vee r_2$, the corresponding nfa is:

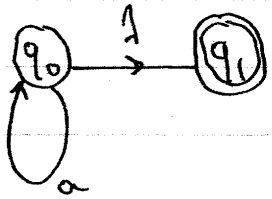


▸ 2) For $r_1 r_2$ the corresponding nfa is:

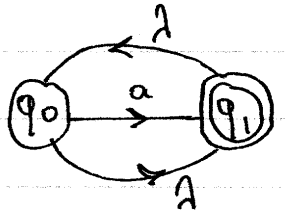
- For r_1^*, the corresponding nfa is:



↳ Special case: For $r = a^*$ with $a \in \Sigma$, the corresponding nfa can be written as:



as an alternative to



EXAMPLES

Construct nfas that accept the following regular expressions:

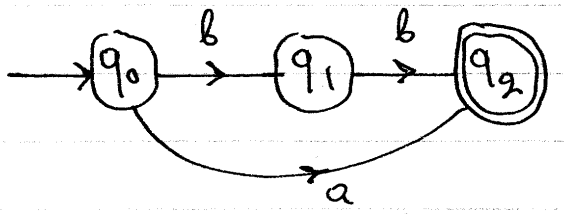
a) $r = a \vee (bb)$

b) $r = (ab^*a) \vee \lambda$

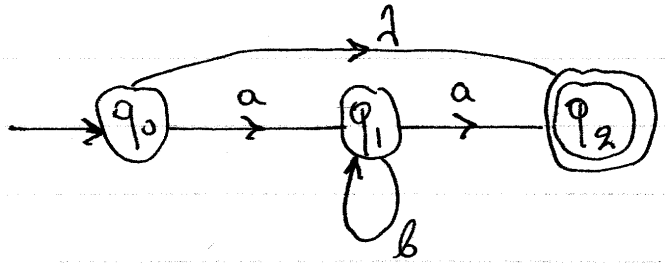
c) $r = (a \vee b)^* (ba^*)$

Solution

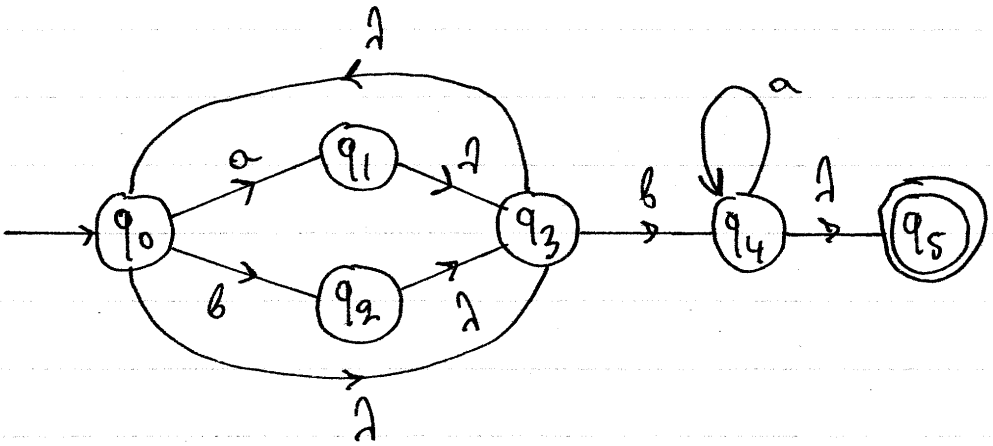
a) For $r = a \vee (bb)$



b) For $r = (ab^*a) \vee \lambda$



c) For $r = (a \vee b)^* (ba^*)$



EXAMPLE

Construct an nfa that accepts the language

$L = \{ab^n a^m b \mid n \in \mathbb{N} \wedge m \in \mathbb{N}\}$

Solution

We note that

$L = \{ab^n a^m b \mid n \in \mathbb{N} \wedge m \in \mathbb{N}\} =$

$= \{a\}\{b^n \mid n \in \mathbb{N}\}\{a^m \mid m \in \mathbb{N}\}\{b\}$

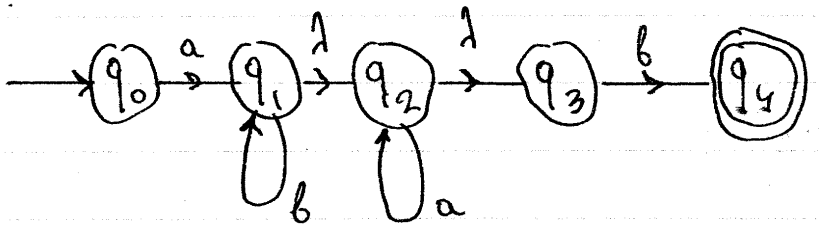
$= \{a\}\{b\}^*\{a\}^*\{b\} =$

$= \mathcal{L}(a)\mathcal{L}(b^*)\mathcal{L}(a^*)\mathcal{L}(b) =$

$= \mathcal{L}(ab^*a^*b)$

and therefore the corresponding nfa is:

M:



EXERCISES

(25) Write the language accepted by the following expressions in set builder notation and construct an non-deterministic finite accepter that accepts that language

a) $r = (a \cup ba)^*$

b) $r = ab(ba)^*$

c) $r = b^*a^*b^2a^*$

d) $r = (ab)^* \cup a^*$

e) $r = (aa^*) \cup (ba^*b^*)$

f) $r = (bab)^* \cup (a \cup b^*)^*$

(26) Construct non-deterministic finite accepters that accept the following regular expressions and convert them to dfas.

a) $r = (ab)^* \cup a$

b) $r = (a \cup b)^* b^*$

c) $r = (a^* \cup (ba))^*$

d) $r = [(ab)^*] \cup a^*$

e) $r = (a^*bab^*)^*$

(27) Use regular expressions to construct non-deterministic finite accepters that accept the following languages, thereby establishing that they are regular.

a) $L = \{ x^a y^b z^c \mid a, b, c \in \mathbb{N} \}$

b) $L = \{ x^{a+2} y^b \mid a, b \in \mathbb{N} \}$

c) $L = \{ xyx^a, x^b y^{a+2} \mid a, b \in \mathbb{N} \}$

d) $L = \{ x^2 y x^{a+1} y^b \mid a, b \in \mathbb{N} \}$

e) $L = \{ x^2 y^a, x^{a+1} y^a \mid a \in \mathbb{N} \}$