1 Language

DEFINITION 1.1(**Language**): **Let Σ denote a finite alphabet (a finite set of letters) A Language over Σ is a subset of Σ^* i.e., a set of strings with letters in Σ.**

DEFINITION 1.2(**Concatenation Operator "$\circ$"**): **Suppose L and L' are Languages, then**

$$L \circ L' = LL' = \{x \cdot y | x \in L \textbf{ and } y \in L'\}$$

Example 1.1. Suppose $L = \{a, bb\}$ and $L' = \{\lambda, c\}$
Then $LL' = \{a, bb, ac, bbc\}$ and $L'L = \{a, bb, ca, cbb\}$

DEFINITION 1.3(λ): **For any string $x \in \Sigma^*$, we have $x \cdot \lambda = \lambda \cdot x = x$**

Remark. $L^0 = \{\lambda\} \neq \lambda$

DEFINITION 1.4(L^i, L^*, L^+):

$$\begin{aligned} L^1 &:= L \\ L^{i+1} &:= L^i \cdot L = L \cdot L^i \\ L^* &:= \bigcup_{i=0}^{\infty} L^i \\ L^+ &:= \bigcup_{i=1}^{\infty} L^i \end{aligned}$$

Remark. $L^* = L^+$ iff $\lambda \in L$

Theorem 1.5
(association) $\forall x, y, z \in \Sigma^*.(xy)z = x(yz) = xyz$

DEFINITION 1.6(**Prefix**): **x is a prefix of y if there exists a string x' such that $y = xx'$**

Example 1.2. λ and y are prefixes of y

DEFINITION 1.7(**Suffix**): **x is a suffix of y if there exist a string such that $y = x'x$**

DEFINITION 1.8(**Substring**): **x is a substring of y if there exists strings x' and x'' such that $x'xx'' = y$**

Example 1.3. Let $y = aabaa$ and $x = aa$
then x is prefix and suffix of y but $x \neq y$

DEFINITION 1.9(**Other operators on Language**): **If L and L' are languages over Σ so are**

$$L \cup L', L \cap L', L - L', \overline{L} = \Sigma^* - L$$

2 Regular Expressions

Definition 2.1**(Regular Expression): Let Σ be a finite alphabet (not include $+, *, \cdot, \lambda, \phi$)**
Let R be the inductively defined set of strings(R is the set of regular Expressions over Σ):

Base Case: $\emptyset, \lambda \in R,\ \Sigma \subseteq R$
Constructor Cases: if $r, r' \in R$**, then** $(r + r') \in R, (r \cdot r') \in R, r^* \in R$

Definition 2.2**(Languages derived by R): The language derived by a regular expression is $\mathcal{L}(r)$ where**

$$\mathcal{L} : R \rightarrow \{L | L \subseteq \Sigma^*\}$$

which is defined inductively
Base cases:

$$\begin{aligned}
&\mathcal{L}(\lambda) := \{\lambda\} \\
&\mathcal{L}(\emptyset) := \emptyset \\
&\mathcal{L}(a) := \{a\} \textbf{ for all } a \in \Sigma
\end{aligned}$$

Constructor Cases:

$$\begin{aligned}
\mathcal{L}((r + r')) &:= \mathcal{L}(r) \cup \mathcal{L}(r') \\
\mathcal{L}((r \cdot r')) &:= \mathcal{L}(r) \cdot \mathcal{L}(r') \\
\mathcal{L}(r^*) &:= \mathcal{L}(r)^*
\end{aligned}$$

Example 2.1. $((r_1 \cdot r_2) \cdot r_3) = (r_1 \cdot r_2 \cdot r_3)$ can remove "(" and ")" when no ambiguty

Definition 2.3**(Regular Lang): A Language L is regular if and only if $L = \mathcal{L}(r)$ for some regular expressions r.**

Definition 2.4**(Equivalent of regular expression): Two regular expression r and r' are equivalent $r \equiv r'$ if and only if**

$$\mathcal{L}(r) = \mathcal{L}(r')$$

Example 2.2. Let L_0 = "string over $\{a, b, c\}$ that start with ab", then the regular expression of L_0 is

$$a \cdot b \cdot (a + b + c)^*$$

i.e., $\mathcal{L}(a \cdot b \cdot (a + b + c)^*) = L_0$

Example 2.3. Let L_1 = "strings over $\{0, 1\}^*$ containing an even number of 1's", then

$$L_1 = \mathcal{L}((0^*10^*1)^*0^*)$$

Example 2.4. Let L_2 = "first and last symbols are different $\subseteq \{0,1\}^*$", then

$$((0(0+1)^*1) + (1(0+1)^*0))$$

Theorem 2.5 *Example 2.3 is true.*
i.e., Denote L_1 as "strings over $\{0,1\}^*$ containing an even number of 1's", then

$$L_1 = \mathcal{L}((0^*10^*1)^*0^*) = \mathcal{L}(r_1)$$

Proof.
Let $x \in L_1$ be arbitrary
if $s \in \{0\}^*$ then
$s \in \mathcal{L}((0^*10^*1)^*0^*))$.
Therefore
If $x \in \mathcal{L}(0^*)$
then $x \in \mathcal{L}((0^*10^*1)^*0^*)$
otherwise x has at least two 1
let x_1 be the shortest prefix of x containing two 1
$x = x_1x'$
Let $x_1 = u1v1$ where $u, v \in \{0\}^* \in \mathcal{L}(r_1)$

Prove from induction on the # of 1's in x
Let $P(n) = \forall x \in \{0,1\}^*.(\text{if } x \text{ contains exactly } 2n \text{ ones then } x \in \mathcal{L}(r_1))$
Suppose P(n)
Let $x \in L_1$ be an arbitrary string such that has $2n+2$ ones
By induction hypothesis,
$x' \in \mathcal{L}(r_1)$, since x_1 has 2n ones.
so $x = x_1x' \in \mathcal{L}(0^*10^*1)\mathcal{L}((0^*10^*1)^*0^*) \subseteq \mathcal{L}((0^*10^*1)^*0^*)$
Hence P(n+1)
By induction $\forall n \in \mathbb{N}.P(n)$
$L_1 \subseteq \mathcal{L}(r_1)$

Suppose $x \in \mathcal{L}(r_1)$
so $\exists k \in \mathbb{N}.x = (x_1...x_k)x'$
where $x_i \in \mathcal{L}(0^*10^*1)$ for $1 \le i \le k$ and $x' \in \mathcal{L}(0^*)$
Note that # 1's in x_i is exactly 2
Number of 1's in x' is 0
Hence x has even number of 1's
i.e.,$x \in L_1$
$\mathcal{L}(r_1) \subseteq L_1$

□

3 Deterministic finite state automaton (DFA or DFSA)

Example 3.1. We determined four things to form a DFA called A

1. Finite set of states $Q = \{q_0, q_1, q_2, q_3\}$
2. Input alphabet $\Sigma = \{0, 1\}$
3. Initial state q_0
4. the set of final states $F = \{q_1, q_2\}$
5. the transition function δ

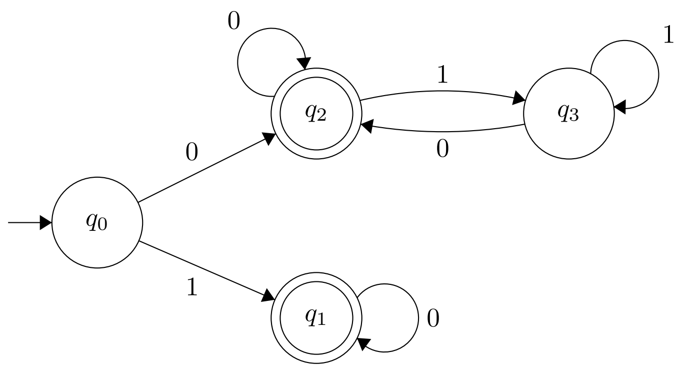

Figure 3.1: A (a example of DFA)

In A, we see that
0110 is accepts, since $q_0 \to q_2$.
0101 is rejected, since $q_0 \to q_3$.

DEFINITION 3.1**(State transition): $\delta : Q \times \Sigma \to Q$ is the transition function.**

$$\delta(q, a) = q' \text{ means that there is a edge labeled } a \text{ from } q \text{ to } q'.$$

DEFINITION 3.2**(deterministic finite state automaton DFA): Formally, a DFA is a 5-tuple**

$$M = (Q, \Sigma, \delta, q_0, F)$$

where $Q, \Sigma, \delta, q_0, F$ are defined in Example 3.1.

DEFINITION 3.3**(Extended transition function): $\delta^* : Q \times \Sigma^* \to Q$ is a extended transition function** BASE CASES**:** $\delta^*(q, \lambda) = q$
CONSTRUCTOR CASES**: for all $a \in \Sigma, x \in \Sigma^*$, we have $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$.**
Alternatively, $\delta^*(q, ax) = \delta^*(\delta(q, a), x)$
If $\delta^*(q, x) = q'$, we say that x takes the automaton M from q to q'.

DEFINITION 3.4**(the language accepted by M): $\mathcal{L}(M) = \{x \in \Sigma^* | M$ accepts $x\}$**
Also, $\mathcal{L}(M) = \{x \in \Sigma^* | \delta^*(q_0, x) \in F\}$

Proof of Example 3.1. Denote $\mathcal{L}(A) = \{x \in \{0, 1\}^* |\ x$ begin with 1 or x begin and end with 0$\}$
We first associate a set of strings (why is not language?) L_i with each state q_i.

$$L_i = \{x \in \Sigma^* | \delta^*(q_0, x) = q_i\}$$

Prove by structural induction or induction on the length of x.
We see that

$$
\begin{aligned}
&L_0 = \{\lambda\} \\
&L_1 = \{x |\ x \text{ start with } 1\} = \mathcal{L}(1(0+1)^*) \\
&L_2 = \{x \in \{0,1\}^* | x \text{ starts and end with } 0\} = \mathcal{L}(0(0+1)^*0 + 0) \\
&L_3 = \{x \in \{0,1\}^* | x \text{ starts with } 0 \text{ and end with } 1\}
\end{aligned}
$$

Then we can prove $L' = L_1 \cup L_2$. □

4 Nondeterministic Finite Automaton NFA

DEFINITION 4.1**(NFA):**

$$M = (Q, \Sigma, \delta, q_0, F)$$

The only difference to DFA is that

$$\delta : Q \times \Sigma \to \mathcal{P}(Q)$$

M acceptes the string X, if there is a path from q_0 to accept state labeld by X.

DEFINITION 4.2**(Extended transition function): Denote the extended transition function $\delta^* : Q \times \Sigma \to \mathcal{P}(Q)$ as**

BASE CASE:	$\delta^*(q, \lambda) = \{q\}$
CONSTRUCTOR CASE:	$\delta^*(q, xa) = \bigcup\{\delta(q', a) \| q' \in \delta^*(q, x)\}$

DEFINITION 4.3**(The language that M accepts):**

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

5 DFA NFA and variant of NFA

Recall. A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

$$\text{DFA } \delta : Q \times \Sigma \to Q$$
$$\text{NFA } \delta : Q \times \Sigma \to \mathcal{P}(Q)$$

The language $L(M)$ accept by M is

$$\text{DFA: } \{x \in \Sigma^* | \delta^*(q_0, x) \in F\}$$
$$\text{NFA: } \{x \in \Sigma^* | \delta^*(q_0, x) \cap F \neq \varnothing\}$$

Theorem 5.1 *For every NFA $M = (Q, \Sigma, \delta, q_0, F)$ there is a DFA $M' = (Q', \Sigma', \delta', q_0', F')$ such that $L(M') = L(M)$*

Proof.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an arbitrary NFA.
Idea: keep track of the states that M can be as it reads the input string.
Let $M' = (Q', \Sigma', \gamma', q_0', F')$ be defined as follows:
$Q' = \mathcal{P}(Q)$
$q_0' = \{q_0\}$
$F' = \{s \in P(Q) = Q' | S \cap F\}$
$\Sigma' = \Sigma$ and $\delta = \delta'$
Denote $\gamma : Q' \times \Sigma \to Q'$ such that
$\gamma(s, a) = \bigcup\{\delta(q, a) | q \in s\}$ for all $s \in Q'$ and $a \in \Sigma$.
This is called subset construction.

Claim $L(M) = L(M')$
For all $w \in \Sigma^*$, let $P(w) :=$ "$\gamma^*(\{q_0\}, w) = \delta^*(q_0, w)$"
Now show that $\forall w \in \Sigma^*.P(w)$ by structural induction
BASE CASE: $w = \lambda$
By definition of extended transition function, we know
$\gamma^*(\{q_0\}, \lambda) = \{q_0\} = \delta^*(q_0, \lambda)$
CONSTRUCTOR CASE: $w = xa$, where $x \in \Sigma^*$ and $a \in \Sigma$
Assume $P(x)$, i.e., $\gamma^*(\{q_0\}, x) = \delta^*(q_0, x)$

$$\begin{aligned}\gamma^*(\{q_0\}, w) &= \gamma(\gamma^*(\{q_0\}, x), a) \text{ by definition} \\ &= \bigcup\{\delta(q, a) | q \in \gamma^*(\{q_0\}, x)\} \text{ by construction} \\ &= \bigcup\{\delta(q, a) | q \in \delta^*(q_0, x)\} \text{ by substitution} \\ &= \delta^*(q_0, w) \text{ by definition}\end{aligned}$$

So $P(w)$ is true.
By induction, $\forall w \in \Sigma^*.P(w)$

Therefore, $w \in \mathcal{L}(M') \iff \gamma^*(\{q_0\}, w) \in F' \iff \gamma^*(\{q_0\}, w) \cap F \neq \varnothing \iff w \in \mathcal{L}(M)$ ☐

6 Variants of NFAs

6.1 NFA with multiple initial states

$$M = (Q, \Sigma, \delta, I, F)$$

where $I \subseteq Q$

$$L(M) = \{x \in \Sigma^* | \exists q \in I.(\delta^*(q, x) \cap F \neq \varnothing)\}$$

i.e., M acceptes x if and only if there is a path from some initial state to a final state labelled by x.

Corollary 6.1 *if L is accepted by an NFA with multiple start start s, then it is accepted by an NFA. we can construct a normal NFA $M'(Q \cup \{q_0\}, \sigma, \delta', q_0, F')$, where $q_0 \notin Q$, such that*

$$\delta'(q_0, a) = \bigcup\{\delta(q, a) | q \in I\} \textit{ for all } a \in \Sigma$$

$$\delta'(q, a) = \delta(q, a) \textit{ for all } q \in Q, a \in \Sigma$$

$$F' = \begin{cases} F & \text{if } I \cap F = \varnothing \\ F \cup \{q_0\} & \text{if } I \cap F \neq \varnothing \end{cases}$$

Proof. omit □

6.2 NFA with λ-transitions

let $M = (Q, \Sigma, \delta, q_0, F)$ where

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$$

Denote $L(M)$ as

$$L(M) = \{x \in \Sigma^* | \delta^*(q_0, x) \cap F \neq \varnothing\}$$

M accepts x if and only if there is a path from q_0 to a final state such that
$x =$ concateration of the labels of the edges on that path.

Corollary 6.2 *if L is accepted by an NFA with λ-transition, then it is accepted by an NFA. Denote $E(q) = \delta^*(q, \lambda) = \{q' \in Q |$ there is a path from q to q' labeled by $\lambda\}$
Then we can construct a NFA with multiple innitial states $M' = (Q, \Sigma, \delta', E(q_0), F)$, where for all $q \in Q, a \in \Sigma$,*

$$\delta'(q, a) = \bigcup\{E(q') | q' \in \delta(q, a)\}$$

Proof. omit □

7 Closure Results

Theorem 7.1 *Suppose $L_1, L_2 \subseteq \Sigma^*$ are accepted by finite automaton, then so are*