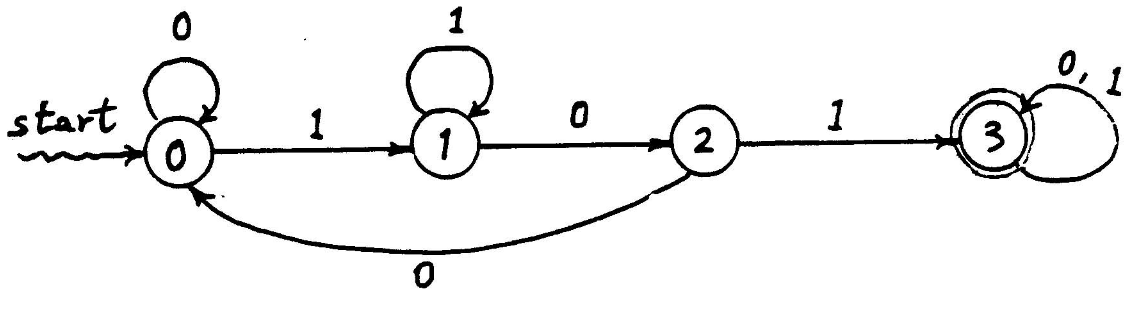
II. FINITE AUTOMATA

Example Design a "sequential lock". The lock has 1-bit sequential input. Initially the lock is closed. If the lock is closed it will open when the last three input signals are "1", "0", "1", and then remains open.

— state (transition) diagram



— state (or transition) table

Present state \ Present symbol	0	1
0	0	1
1	2	1
2	0	3
3	3	3

state set : $\{0,\ 1,\ 2,\ 3\}$
input alphabet : $\{0,\ 1\}$
Transition function : $\delta(0,0) = 0,\ \delta(0,1) = 1,\ \ldots$
Start state : 0
Final state set : $\{3\}$

<u>Deterministic Finite Automata</u> (DFA)

$M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **where**

Q **is a finite nonempty <u>set of states</u>**

Σ **is the <u>input alphabet</u>**

$\delta : Q \times \Sigma \to Q$ **<u>transition function</u>**

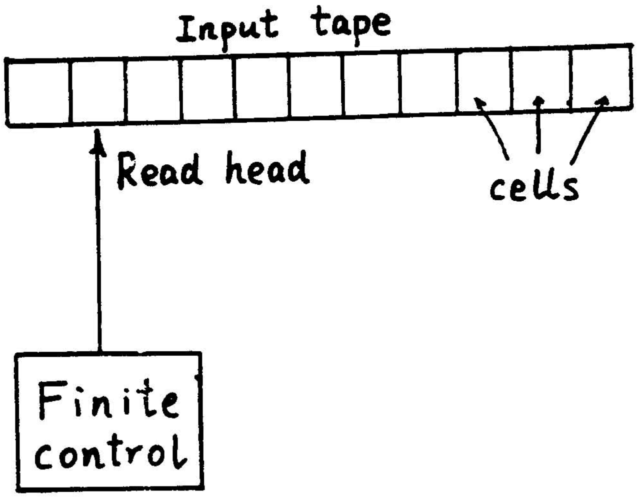
s **<u>start state</u>**

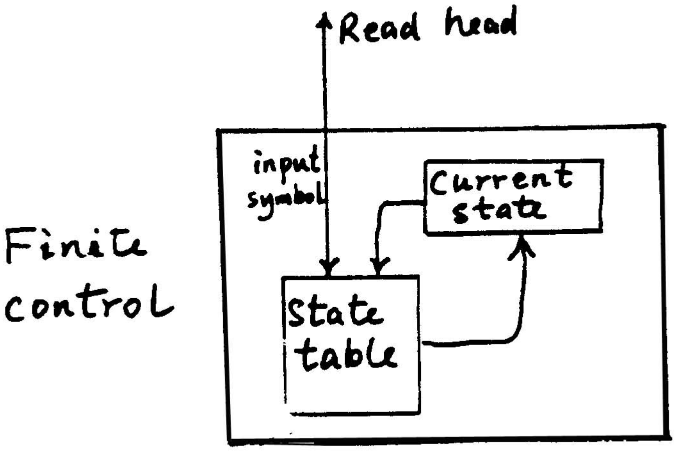
$F \subseteq Q$ **<u>final state set</u>**

A computer is a finite state system (i.e. FA) which has millions of states.

There are many examples of <u>finite state systems</u>. A finite automaton is an <u>abstraction</u> of them.

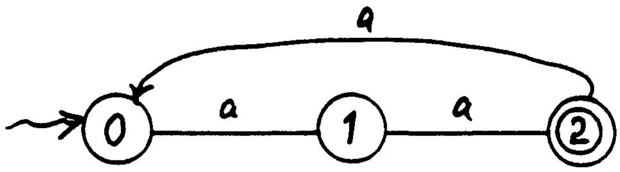
<u>View a DFA as a machine</u>



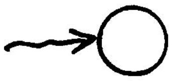


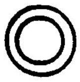
Specifying δ

1)



State diagram
(Transition diagram)

 **Start state**

 **Final state**

2)

Present state \ Present symbol	a
0	1
1	2
2	0

Configurations

a word in $Q\Sigma^*$

$$px$$

where p **is the present state, and**
x **is the remaining input**

Example:

$0aa \quad \ldots \quad 1a \quad \ldots \quad 2$

(start configuration) **(final configuration)**

Moves of a DFA

$0aa \vdash 1a$ $\qquad$ $px \vdash qy$

$1a \vdash 2$ $\qquad$ **if** $x = ay$ **and** $\delta(p, a) = q$

Configuration sequence

$0aa \vdash 1a \vdash 2$

$\vdash^+$ and $\vdash^*$

$\vdash$ **is a binary relation over** $Q\Sigma^*$**.**

$\vdash^+$: **transitive closure of** $\vdash$**.**

$\vdash^*$: **reflexive transitive closure of** $\vdash$**.**

$$0aa \vdash^+ 2$$
$$0aa \vdash^* 2$$
$$0aa \vdash^* 0aa$$
$$0aa \vdash^2 2$$

$$px \vdash^k qy$$

if $px \underbrace{\vdash p_{i_1}x_{i_1} \vdash p_{i_2}x_{i_2} \vdash \ldots}_{k \textbf{ steps}} \vdash qy$

Accepting Configuration Sequence

$0aa \vdash 1a \vdash 2$

$\vdash$ **can also be viewed as a** **function**

$$\vdash : Q\Sigma^* \rightarrow Q\Sigma^*,$$

since the next configuration is determined uniquely for a given configuration.

The DFA stops when:

(i) we have no more input, or

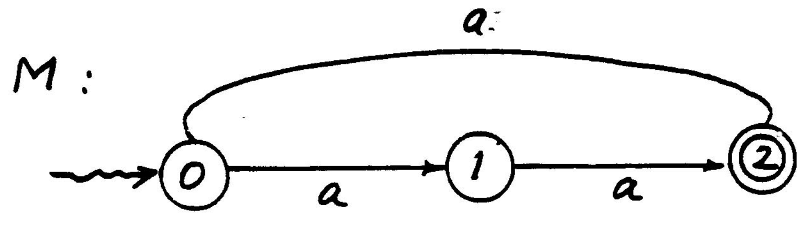
(ii) the next configuration is undefined.

A word x **is said to be accepted by a DFA** M **if** $sx \vdash^* f,\ f \in F$**.**

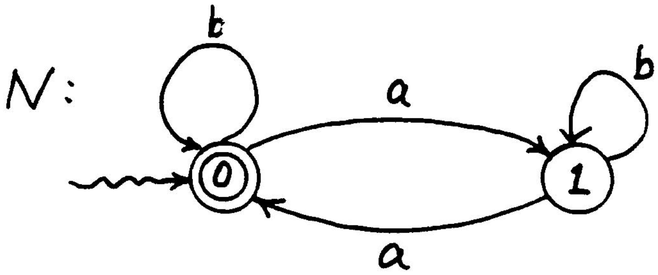
The language of a DFA M**,** $L(M)$**, is defined as:**

$$L(M) = \{x \mid sx \vdash^* f, \text{ for some } f \in F\}$$

Examples



$L(M) =$



$L(N) =$

DFA membership problem

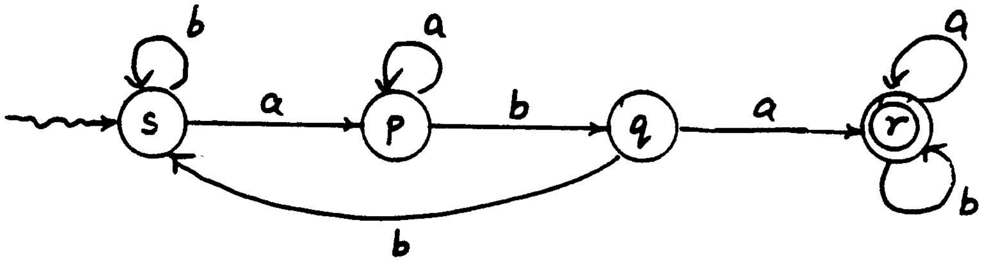
DFA MEMBERSHIP

INSTANCE: A DFA, $M = (Q, \Sigma, \delta, s, F)$
and a word $x \in \Sigma^*$.

QUESTION: Is x **in** $L(M)$ **?**

Run the DFA M **with input** x**.**
In at most $|x|$ **steps it accepts, rejects or aborts.**

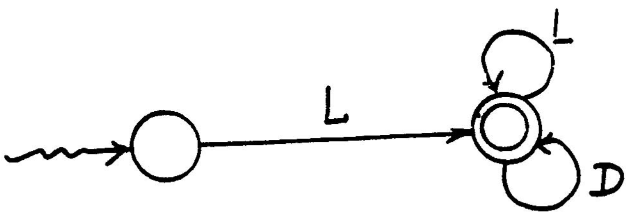
Examples



Checking for words that contain $\underline{aba}$ **as subword.**

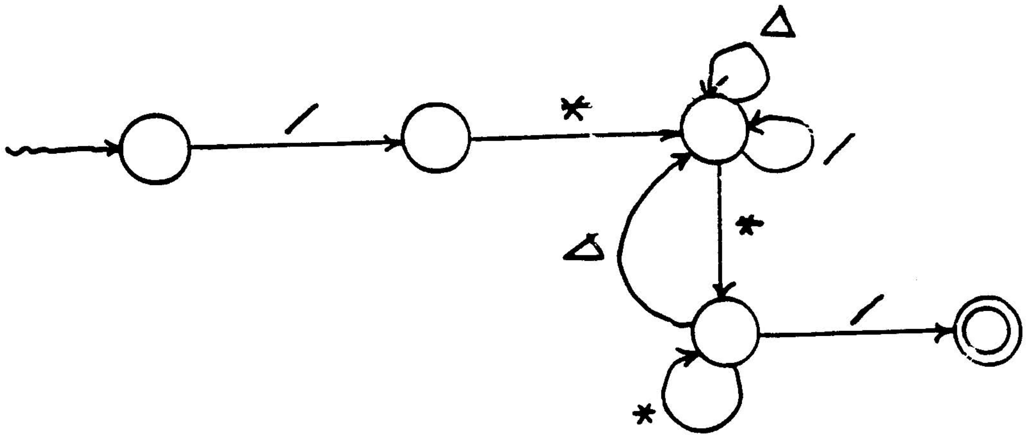
Check: $ababba$
$abbaabbbab$

Let L denote any letter of English alphabet and D any decimal digit; the form of PASCAL IDENTIFIERS can be specified by



Recognizing comments that may go over several lines.

/*...............*/

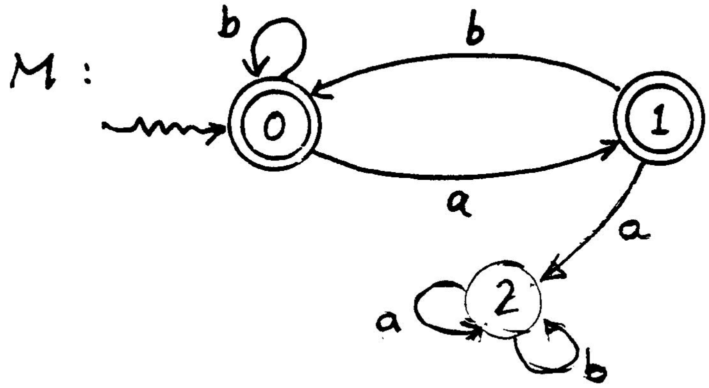


Δ: symbols other than "$*$" and "/"

A DFA which has a total δ is said to be complete; if δ is nontotal it is incomplete.

Theorem. Every incomplete DFA M can be "completed" by adding one new state ("sink") to give DFA M' such that $L(M') = L(M)$.

Example:



$L(M)$ is the set of all words that do not contain two consecutive a's.

$\triangle$ **Two DFA M_1 and M_2 are equivalent if $L(M_1) = L(M_2)$.**

$\triangle$ **The collection of languages accepted by DFA's is denoted by**

$$\underline{\mathcal{L}_{DFA}}.$$

It is called the family of DFA languages and it is defined as:

$$\mathcal{L}_{DFA} = \{L \mid L = L(M) \textbf{ for some DFA } M\ \}$$

$\triangle$ $K = \{a^i b^i \mid i \geq 1\}$ **is not accepted by any DFA.**

Proof: Use contradiction and Pigeonhole principle.

Assume $K = L(M)$, for some DFA

$$M = (Q, \{a, b\}, \delta, s, F).$$

Let $n = \#Q$. Consider the accepting configuration sequence for $a^n b^n$,

$$s_0 a^n b^n \vdash s_1 a^{n-1} b^n \vdash \ldots \vdash s_n b^n \vdash \ldots \vdash s_{2n}$$

where $s_0 = s$ and $s_{2n} \in F$. Now $n+1$ states appear during the reading of a^n, but there are only n distinct states in Q. By Pigeonhole principle at least one state must appear at least twice during the reading of a's.

Assume $s_i = s_j, 0 \leq i < j \leq n$. Then

$$\begin{aligned} s_0 a^{n-(j-i)} b^n \vdash \ldots \vdash s_i a^{n-j} b^n & \\ & s_j a^{n-j} b^n \vdash \ldots \vdash s_n b^n \vdash \ldots \vdash \\ & \quad \vdash s_{2n} \end{aligned}$$

Therefore $a^{n-(j-i)} b^n \in K$. This is a contradiction.

$\triangle$ $L_i = \{a^i b^i\}, i \geq 1.$

For any $i \geq 1$, is L_i a DFA language?

$\triangle$ $K_j = \{a^i b^i : 0 \leq i \leq j\}, j \geq 1.$

For any $j \geq 1$, is K_j a DFA language ?

<u>Nondeterministic Finite Automata</u> (NFA)

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

same as a DFA except

$$\delta \subseteq Q \times \Sigma \times Q.$$

Here, δ is a finite <u>transition relation</u>.

In a DFA, δ is a transition function:

$$\delta : Q \times \Sigma \rightarrow Q.$$

It can be viewed as a relation

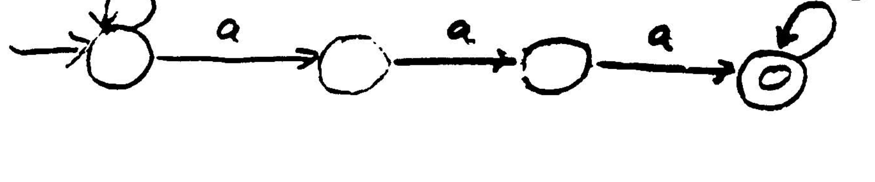
$$\delta : Q \times \Sigma \times Q.$$

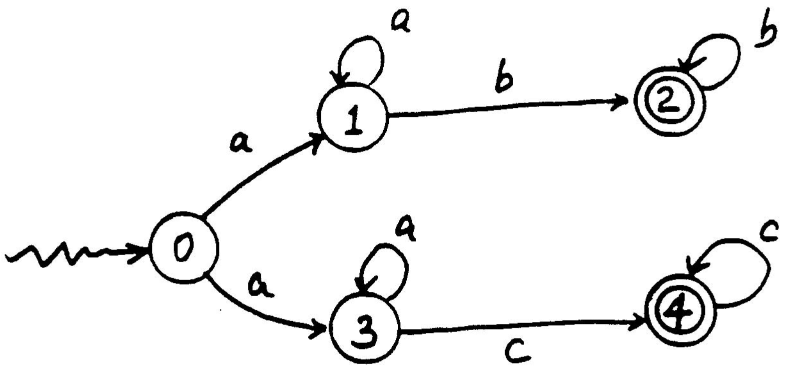
In a NFA, δ can be be viewed as a function:

$$\delta : Q \times \Sigma \rightarrow 2^Q.$$

<u>Examples</u>:

NFA for words in $\{a, b\}^*$ that contain three consecutive a's.





Both $(0, a, 1)$ **and** $(0, a, 3)$ **are in** δ**.**

We define acceptance by existence of a computation that leads to a final state.

Conversely, we define rejection by the nonexistence of any computation that leads to a final state.

The language of an NFA $M = (Q, \Sigma, \delta, s, F)$ **is defined by**

$L(M) = \{x \mid sx \vdash^* f$**, for some** f **in** F $\}$**.**

The family of NFA languages $\mathcal{L}_{NFA}$

is defined by:

$\mathcal{L}_{NFA} = \{L \mid L = L(M)$, **for some NFA** M $\}$.

Two NFAs M_1, **and** M_2 **are equivalent if** $L(M_1) = L(M_2)$.

Why NFA?

(i) easy to construct;
(ii) useful theoretically;
(iii) are of same power as DFA.

Note:

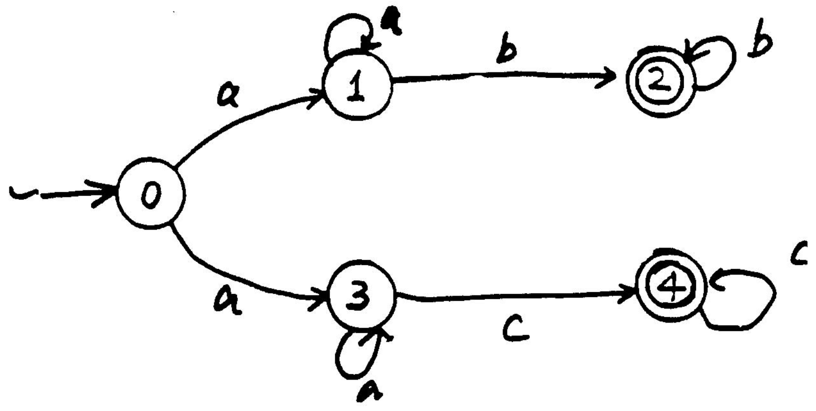
configurations are defined in the same way
Transition (move)

$$px \vdash qy$$

if $x = ay$, **for some** $a \in \Sigma$, **and** $(p, a, q) \in \delta$.

Transforming NFA to DFA

Consider the NFA M_1 again



At each step, there are only limited number of choices totally. For example:

$0\underline{a}ab \vdash 1ab \vdash 1b \vdash 2$

$0aab \vdash 3ab \vdash 3b$

Why <u>limited</u> number of choices?

The state set is finite.

We summarize the choices at each step by combining all configuration sequences into one "<u>super-configuration sequence</u>".

$\{0\}aab \vdash \{1,3\}ab \vdash \{1,3\}b \vdash \{2\}.$

We now have a set of all possible states at each step. From this point of view the computation of the NFA on an input word is <u>deterministic</u>.

A <u>super-configuration</u> has the form

$$Kx$$

where $K \subseteq Q$ and $x \in \Sigma^*$.

Note that $\underline{\emptyset x}$ is a super-configuration, it means that the NFA cannot be in any state at that point, i.e., an abort has occured.

We say that

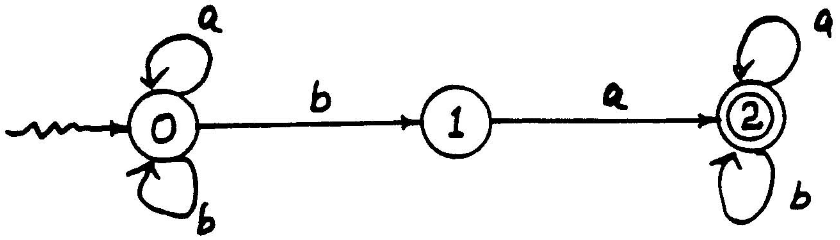
$$Kx \vdash Ny$$

if $x = ay$, for some $a \in \Sigma$, and

$$N = \{q \mid (p, a, q) \in \delta, \text{ for some } p \in K\}.$$

More <u>examples</u> on super-configurations

M: $L(M)$ is the set of all words that have "ba" as a subword.



The super-configuration sequence on input word "$abbaa$" is as follows:

$\{0\}abbaa \vdash \{0\}bbaa \vdash \{0,1\}baa \vdash \{0,1\}aa$
$\vdash \{0,2\}a \vdash \{0,2\}$

Notice that given a set $K \subseteq Q$ and an input symbol $a \in \Sigma$, the set $N \subseteq Q$ s.t. $Ka \vdash N$ is uniquely determined.

<u>Lemma</u> (2.3.1) (Determinism Lemma)
Let $M = (Q, \Sigma, \delta, s, F)$ be an NFA. Then for all words $\underline{x}$ in Σ^* and for all $\underline{K} \subseteq Q$.

$$Kx \vdash^* N \textbf{ and } Kx \vdash^* P$$

implies

$$\underline{P = N}.$$

<u>Lemma</u> (2.3.2) Let $M = (Q, \Sigma, \delta, s, F)$ be an NFA. Then for all words $\underline{x}$ in Σ^* and for all $\underline{q}$ in Q,

$$qx \vdash^* p$$

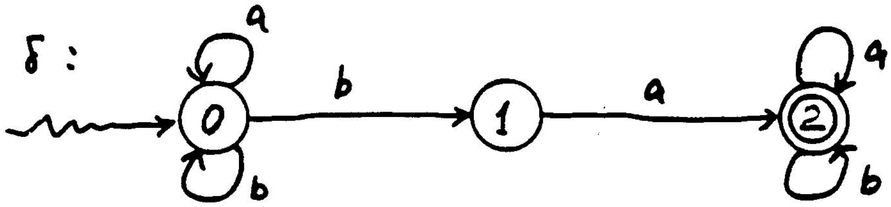
iff $\underline{\{q\}x \vdash^* P}$, for some P with p in P.

Example (Transformation of an NFA to a DFA)

$M = (Q, \Sigma, \delta, s, F)$ **where**

$Q = \{0, 1, 2\}, \quad \Sigma = \{a, b\} \qquad s = 0, \qquad F = \{2\}$

δ :



$M' = (Q', \Sigma, \delta', s', F')$ **where**

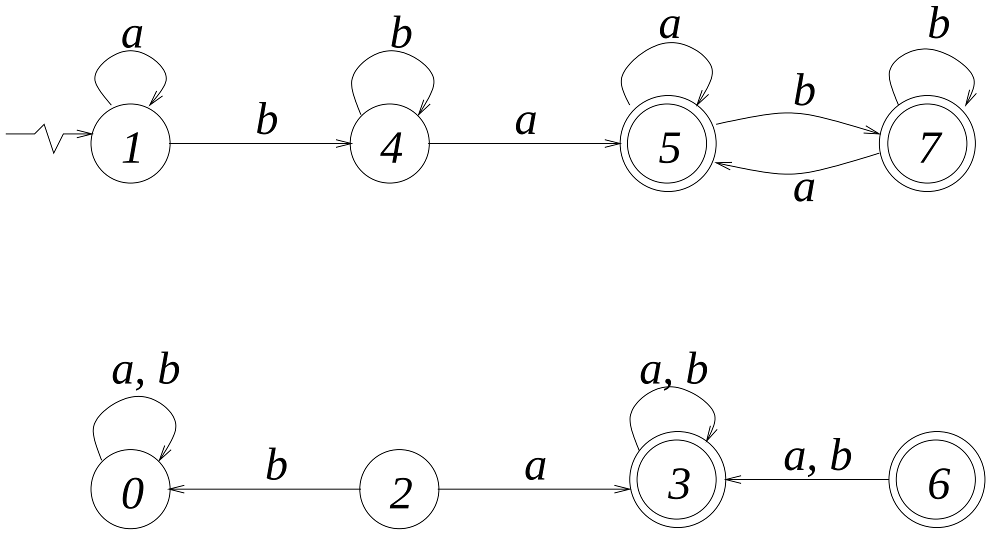
$Q' = 2^Q =$

$$\{\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\}$$

δ':

	current state \ input symbol	a	b
0	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{0\}$	$\{0\}$	$\{0,1\}$
2	$\{1\}$	$\{2\}$	$\emptyset$
3	$\{2\}$	$\{2\}$	$\{2\}$
4	$\{0,1\}$	$\{0,2\}$	$\{0,1\}$
5	$\{0,2\}$	$\{0,2\}$	$\{0,1,2\}$
6	$\{1,2\}$	$\{2\}$	$\{2\}$
7	$\{0,1,2\}$	$\{0,2\}$	$\{0,1,2\}$

$\underline{\delta'(P, a) = \{q \mid (p, a, q) \in \delta \textbf{ and } p \in P\}}$



$s' = \{0\}$

$F' = \{$

Algorithm NFA to DFA — The Subset Construction

On entry: **An NFA** $M = (Q, \Sigma, \delta, s, F)$.

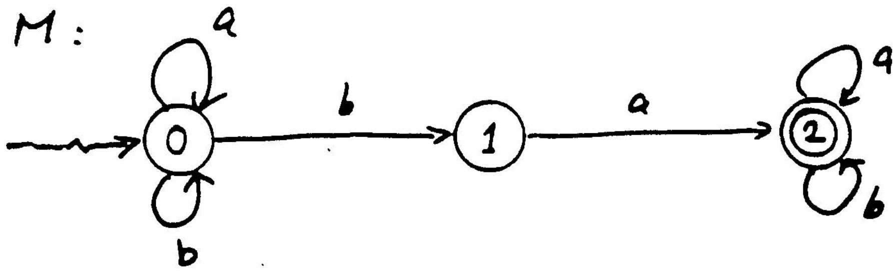
On exit: **A DFA** $M' = (Q', \Sigma, \delta', s', F')$ **satisfying** $L(M) = L(M')$**.**

begin

Let $Q' = 2^Q, s' = \{s\}$ **and** $F' = \{K \mid K \in Q'$ **and** $K \cap F \neq \emptyset\}$**.**

We define $\delta' : Q' \times \Sigma \to Q'$ **by**
for all $K \in Q'$ **and for all** $a \in \Sigma$**,**
$\delta'(K, a) = N$**, where**
$N = \{q \mid (p, a, q) \in \delta$ **and** $p \in K\}$**.**

end of Algorithm

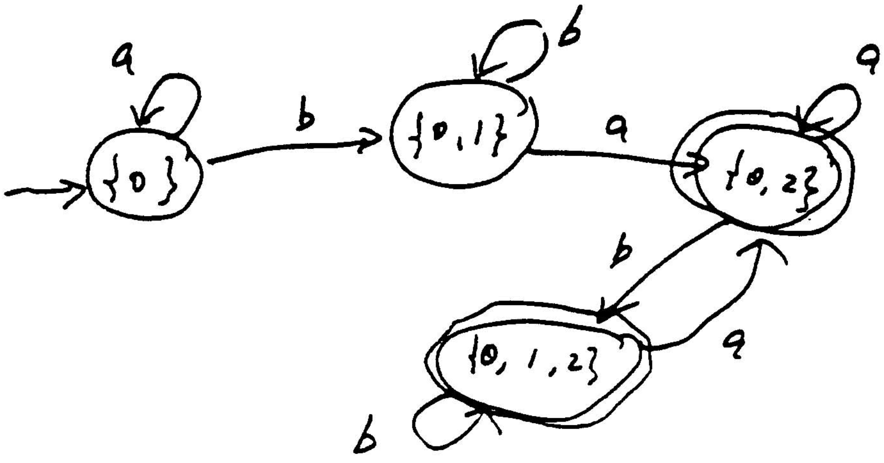


$s' = \{0\}$

input symbol / current state	a	b
$\{0\}$	$\{0\}$	$\{0, 1\}$
$\{0,1\}$	$\{0,2\}$	$\{0,1\}$
$\{0,2\}$	$\{0,2\}$	$\{0,1,2\}$
$\{0,1,2\}$	$\{0,2\}$	$\{0,1,2\}$

	a	b
0	$\{0\}$	$\{0, 1\}$
1	$\{2\}$	ϕ
2	$\{2\}$	$\{2\}$

Algorithm NFA to DFA 2
—<u>The Iterative Subset Construction</u>



<u>Theorem</u> Given an NFA $M = (Q, \Sigma, \delta, s, F)$, then the DFA $M' = (Q', \Sigma', \delta', s', F')$ obtained by either subset construction satisfies $L(M') = L(M)$.

<u>Proof:</u>

By Lemma 2.3.2, for all $x \in \Sigma^*$ in M

$sx \vdash^* p,$ **iff** $\{s\}x \vdash^* P$ **for some** P **with** $p \in P$

By the construction of M',
$\{s\}x \vdash^* P$ **in** M **(**P **is a super-conf. in** M**) iff** $\{s\}x \vdash^* P$ **in** M' **(**P **is a state in** M'**).**

$$
\begin{aligned}
x \in L(M) &\Leftrightarrow sx \vdash^* f, \text{ for some } f \in F \\
&\Leftrightarrow \{s\}x \vdash^* P,\ f \in P, \text{ in } M \\
&\Leftrightarrow \{s\}x \vdash^* P, \text{ in } M' \text{ and } P \cap F \neq \emptyset \\
&\Leftrightarrow s'x \vdash^* P,\ P \in F' \\
&\Leftrightarrow x \in L(M')
\end{aligned}
$$

<u>Theorem</u> Every NFA Language is a DFA language and conversely.
$(\mathcal{L}_{NFA} = \mathcal{L}_{DFA})$

<u>Example</u>
Every <u>finite</u> language is accepted by a DFA.

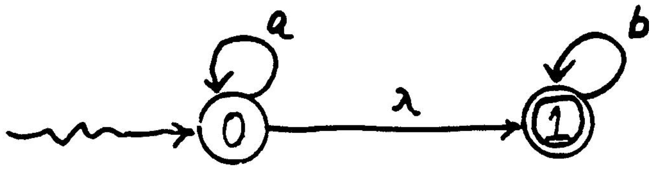
ε–NFA

It is useful to loosen the definition of NFA even more by allowing the <u>read head</u> to remain over the same symbol of the input and <u>read nothing</u>.

Example

$L_1 = \{a^i b^j \mid i \geq 0,\ j \geq 0\}$

M:



$$L(M) = L_1$$

$$\begin{array}{ll} \delta: & (0, a, 0) \\ & \underline{(0, \varepsilon, 1)} \quad \varepsilon - transition \\ & (1, b, 1) \end{array}$$

$$\begin{array}{rl} 0aab & \vdash\ 0ab\ \vdash\ 0b\ \vdash\ 1b\ \vdash\ 1 \\ 0bb & \vdash\ 1bb\ \vdash\ 1b\ \vdash\ 1 \\ 0a & \vdash\ 0\ \vdash\ 1 \end{array}$$

Formally, a ε-NFA $M = (Q, \Sigma, \delta, s, F)$ where Q, Σ, s, F are as before, but δ is a finite transition relation for which

$$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$$

Configurations are as before.

$\vdash$ is defined by

$$px \vdash qy$$

if either $\underline{x = ay}$ for $a \in \Sigma$ and $(p, a, q) \in \delta$ or $\underline{x = y}$ and $(p, \varepsilon, q) \in \delta$

<u>Example</u>

Given FA M_1 and M_2, construct a FA M_3 such that

$$L(M_3) = L(M_1) \cup L(M_2)$$

Transforming ε-NFA to NFA

Two steps:

Step I: ε-completion

Step II: ε-transition removal

(I). ε-Completion

Given a ε-NFA $M = (Q, \Sigma, \delta, s, F)$ **perform the following process:**

> **For** all $p, q, r \in Q$:
>
> **whenever** $(p, \varepsilon, q), (q, \varepsilon, r)$ **are in** δ
>
> **add** (p, ε, r) **to** δ
>
> **until** no new transitions are added to δ
>
> **and let this be** $\underline{\delta'}$.

Let the new $\varepsilon-$NFA be

$M' = (Q, \Sigma, \delta', s, F')$

where $\delta' = \delta \cup \{(p, \varepsilon, q) \mid p \vdash^{+} q\}$ **and**

$F' = F \cup \{p \mid (p, \varepsilon, f) \in \delta'$ **and** $f \in F\}$

Claim 1: **For any** $p,\ q \in Q$,

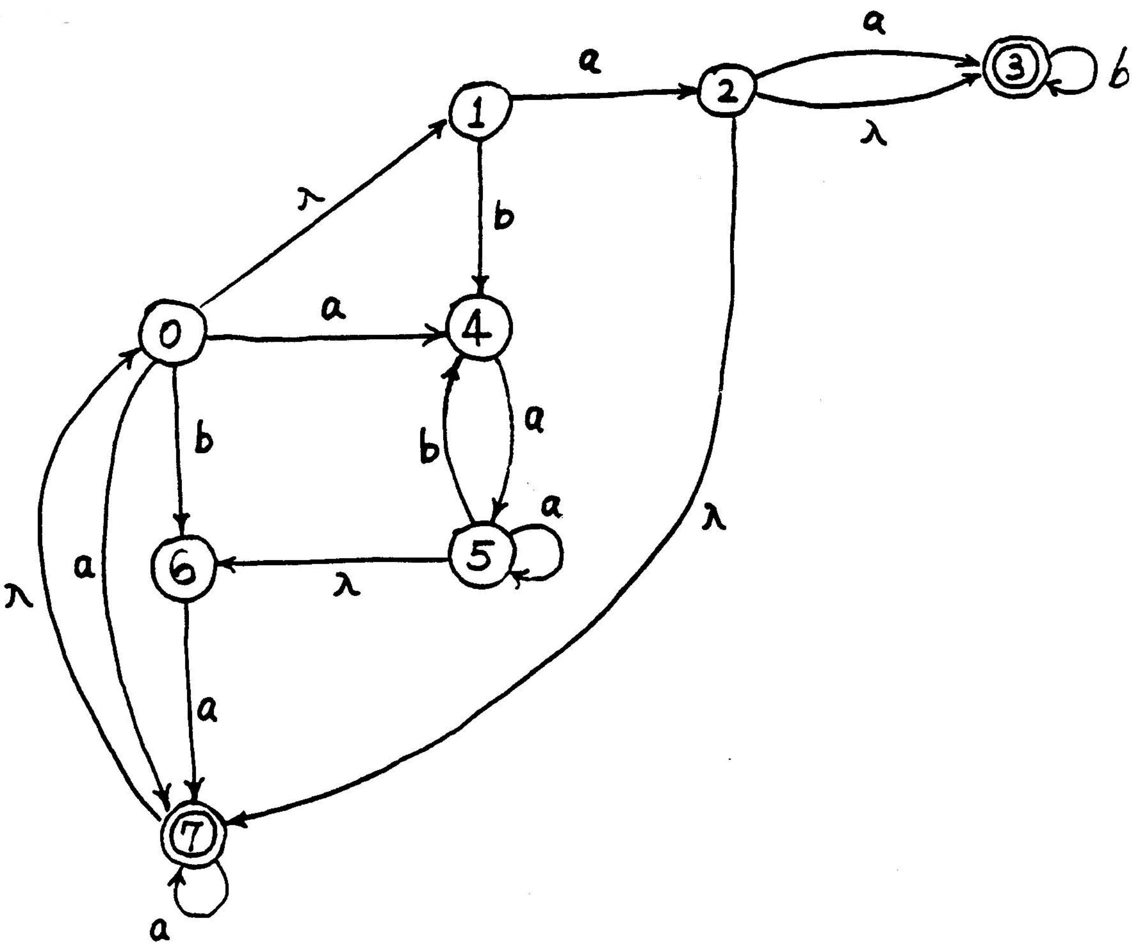
$$p \ \vdash_M^+ \ q \text{ if and only if } p \ \vdash_{M'} \ q$$

Claim 2: **For any** $p,\ q \in Q,\ x \in \Sigma^*$,

$$px \ \vdash_M^* \ q \text{ if and only if } px \ \vdash_{M'}^* \ q$$

Theorem: $L(M') = L(M)$

Example:



(II) ε–Transition Removal

Given a ε-completed ε-NFA

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

perform the following process:

(0) $\delta' = \delta$;

(i) **For all** $p, q, r \in Q$,

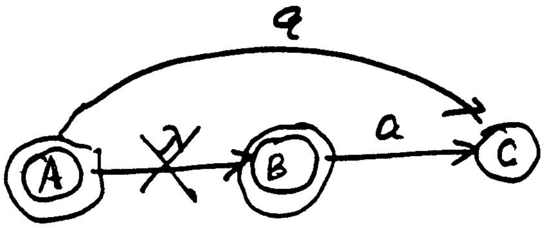
if $\underline{(p, \varepsilon, q)}$ **and** $\underline{(q, a, r)}$ **in** δ

then **<u>add</u>** $\underline{(p, a, r)}$ **to** δ';

(ii) **<u>Delete</u> all <u>ε-transitions</u> from** δ'.

Now we got $M' = (Q,\ \Sigma,\ \delta',\ s,\ F)$
where $\delta' = (\delta \cup \{(p, a, r) \mid (p, \varepsilon, q), (q, a, r) \in \delta\})$
$-\{(p, \varepsilon, q) \mid p, q \in Q\}$

Example



Claim **Whenever**

$$sx \;\vdash_M^*\; f$$

for some $f \in F$**, we have**

$$sx \;\vdash_{M'}^*\; f$$

and vice versa.

Claim $L(M') = L(M)$

Theorem

$$\mathcal{L}_{\varepsilon-NFA} = \mathcal{L}_{NFA}$$

Properties of FA Languages

I. Closure Properties

1. Union: The union of two DFA languages is a DFA language.

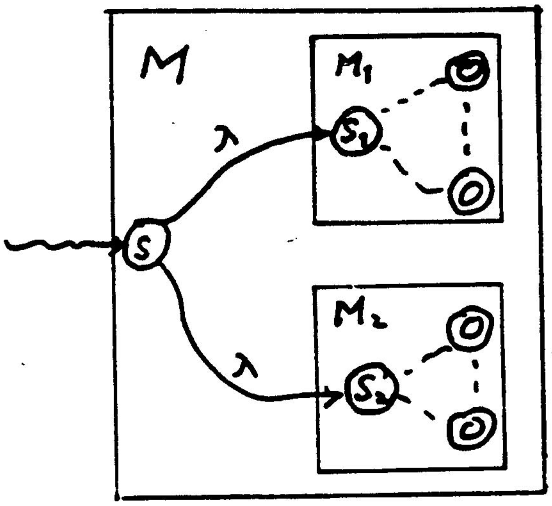
Proof: Given $L_1 = L(M_1)$ and $L_2 = L(M_2)$, $L_1,\ L_2$ are generic DFA languages.

$$M_1 = (Q_1,\ \Sigma_1,\ \delta_1,\ s_1,\ F_1) \text{ and}$$

$$M_2 = (Q_2,\ \Sigma_2,\ \delta_2,\ s_2,\ F_2).$$

Construct an ε-NFA $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ where

$$\begin{aligned}
Q &= Q_1 \cup Q_2 \cup \{s\} \\
\Sigma &= \Sigma_1 \cup \Sigma_2 \\
\delta &= \delta_1 \cup \delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\} \\
F &= F_1 \cup F_2
\end{aligned}$$



Now we claim that $L(M) = L_1 \cup L_2$

(1) Let x be a generic word in $\underline{L(M)}$.

$x \in L(M)$ **implies**

$sx \vdash_M^+ f$ **for some** $f \in F_1$ **or** $f \in F_2$**.**

If $f \in F_1$, then $sx \vdash_M s_1x \vdash_M^* f$ **and then** $s_1x \vdash_{M_1}^* f$**. We know that** $x \in L(M_1)$**.**

If $f \in F_2$, then $sx \vdash_M s_2x \vdash_M^* f$**. Since** $s_2x \vdash_{M_2}^* f$**, we know that** $x \in L(M_2)$**, therefore** $x \in L(M_1) \cup L(M_2)$**, i.e.,** $x \in L_1 \cup L_2$**. So** $L(M) \subseteq L_1 \cup L_2$**.**

(2) Let x be a generic word in $L_1 \cup L_2$.

Without losing of generality we can assume that $x \in L_1$, i.e., $x \in L(M_1)$.

Then $s_1 x \vdash^*_{M_1} f$, for some $f \in F_1$.

Then $sx \vdash_M s_1 x \vdash^*_M f$. Therefore $x \in L(M)$. So, $L_1 \cup L_2 \subseteq L(M)$.

(3) By the results of (1) and (2), we conclude $L(M) = L_1 \cup L_2$.

Since every ε-NFA language is a DFA language, $L(M)$ is a DFA language.

2. Complementation:

The complement of a DFA language $L \subseteq \Sigma^*$ is a DFA language.

Proof: There is a complete DFA $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **with** $L = L(M)$**. Define** $\overline{M} = (Q,\ \Sigma,\ \delta,\ s,\ Q - F)$

3. Intersection:

L_1 **and** L_2

$$L \;=\; L_1 \cap L_2 \;=\; \overline{\overline{L_1} \cup \overline{L_2}}$$

4. Catenation:

The catenation of two DFA languages is a DFA language:

$$M_1 = (Q_1,\ \Sigma_1,\ \delta_1,\ s_1,\ F_1);$$
$$M_2 = (Q_2,\ \Sigma_2,\ \delta_2,\ s_2,\ F_2).$$

$M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **where**

$$\begin{array}{rcl} Q & = & Q_1 \cup Q_2 \\ \Sigma & = & \Sigma_1 \cup \Sigma_2 \\ \delta & = & \delta_1 \cup \delta_2 \cup \{(f, \varepsilon, s_2) \mid f \in F_1\} \\ s & = & s_1 \\ F & = & F_2 \end{array}$$

5. Star:

The star of a DFA language is a DFA language

$L = L(M)$ **where** $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$

$L^* = L(M')$

$M' = (Q',\ \Sigma',\ \delta',\ s',\ F')$ **where**

$$\begin{aligned} Q' &= Q \cup \{s'\} \\ \Sigma' &= \Sigma \\ \delta' &= \delta \cup \{(s', \varepsilon, s)\} \cup \{(f, \varepsilon, s') \mid f \in F\} \\ F' &= F \cup \{s'\} \quad \mathit{or}\ F' = \{s'\} \end{aligned}$$

6. Plus:

II. Decidability Properties

A decision problem is a problem each instance of which is either false or true.

A decision algorithm is an algorithm whose result for each possible input is either false or true.

A decision problem is decidable if there exists a decision algorithm for it. Otherwise it is undecidable.

1. **DFA membership**
   INSTANCE: A DFA $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ and a word $x \in \Sigma^*$.
   QUESTION: Is $x \in L(M)$?

   Theorem: DFA membership is decidable.

Proof: Compute for the given M and x the terminating state q such that $sx \ \vdash^* \ q$. If $q \in F$ then answer "True" else answer "false".

2. DFA Emptiness

INSTANCE: A DFA $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$.
QUESTION: Is $L(M) = \emptyset$?

Theorem DFA emptiness is decidable.

Proof: $L(M) = \emptyset$ iff there is no path in the state diagram of M from s to a final state. If $F = \emptyset$, then this holds immediately.
Otherwise, enumerate the states that can be reached from s. (Mark s. Now mark all states reachable by one transition from one of the marked states. Repeat this until no newly marked state is introduced. The marked states are the reachable states).

3. <u>**DFA Universality**</u>

INSTANCE: A DFA $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$**.**
QUESTION: Is $L(M) = \Sigma^*$ **?**

<u>Theorem</u> **DFA universality is decidable.**
<u>**Proof:**</u>

4. <u>**DFA Containment**</u>

INSTANCE: Two DFA
$M_1 = (Q_1,\ \Sigma_1,\ \delta_1,\ s_1,\ F_1)$ **and**
$M_2 = (Q_2,\ \Sigma_2,\ \delta_2,\ s_2,\ F_2)$**.**
QUESTION: Is $L(M_1) \subseteq L(M_2)$ **?**

<u>Theorem</u> **DFA containment is decidable.**
<u>**Proof:**</u>

$L(M_1) \cap \overline{L(M_2)} = \emptyset$ **means that** $L(M_1) \subseteq L(M_2)$

5. <u>DFA Equivalence</u>

INSTANCE: Two DFA

$M_1 = (Q_1,\ \Sigma_1,\ delta_1,\ s_1,\ F_1)$ **and**

$M_2 = (Q_2,\ \Sigma_2,\ delta_2,\ s_2,\ F_2)$**.**

QUESTION: Is $L(M_1) = L(M_2)$ **?**

<u>**Theorem**</u> **DFA equivalence is decidable.**

$L(M_1) \subseteq L(M_2)$ **? yes**

$L(M_2) \subseteq L(M_1)$ **? yes**

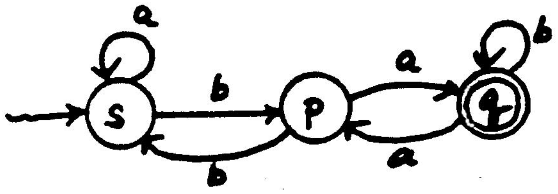
Pumping Lemma and Non-DFA Language.

The DFA Pumping Lemma

Let $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **be a DFA and let** $p = \#Q$**. For** **<u>all</u>** **words** x **in** $L(M)$ **such that** $|x| \geq p$, x **can be decomposed into** uvw, **for some** u, v, **and** w **in** Σ^* **such that**

- **(i)** $|uv| \leq p$;
- **(ii)** $|v| \geq 1$; **and**
- **(iii)** **for all** $i \geq 0$, uv^iw **is in** $L(M)$.

<u>Example</u> $M = (Q, \Sigma, \delta, s, F)$ **where**
$Q = \{s,\ p,\ q\}$, $\Sigma = \{a,\ b\}$,
$F = \{q\}$. $\#Q = 3$



Let x **be an arbitrary word of length** $\geq \#Q$ **in** $L(M)$**, e.g.** $x = baaa$**. The accepting conf. sequence for** x **is:**
$\underline{s}baaa \ \vdash\ paaa \ \vdash\ \underline{q}aa \ \vdash\ pa \ \vdash\ \underline{q}$**.**
So, $u = b,\ v = aa,\ w = a$ **and** $b(aa)^i a \in L(M)$ **for all** $i \geq 0$**.**

Proof of DFA Pumping Lemma:

Let x be in $L(M)$ with $|x| \geq p$.
Then M has an accepting configuration sequence

$$q_0 a_1 a_2 \ldots a_r \;\vdash\; q_1 a_2 \ldots a_r \;\vdash\; \ldots \;\vdash q_{r-1} a_r \;\vdash\; q_r$$

where $q_0 = s,\; q_r \in F$.
Consider the first p transitions. $s,\; q_1,\; \ldots,\; q_p$ **cannot all be distinct since there are only p distinct states (by pigeonhole principle).**
This means that $q_i = q_j$ for some $0 \leq i < j \leq p$.
Let

$$u = a_1 \ldots a_i,$$

$$v = a_{i+1} \ldots a_j,$$

$$w = a_{j+1} \ldots a_r.$$

So, we have $q_0 u \;\vdash^*\; q_i$

$$q_i v \;\vdash^*\; q_j \qquad (q_i = q_j)$$

$$q_j w \;\vdash^*\; q_r.$$

Since $q_i = q_j$, $q_i v^k \;\vdash^*\; q_j$ for any $k \geq 0$

$(i)\ \ |uv| \le p,$ **since** $j \le p$

$(ii)\ \ |v| \ge 1,$ **since** $i < j$

$(iii)\ \ uv^k w \in L(M)$ **for all** $k \ge 0,$ **since**

$$q_0 uv^k w \ \vdash^* \ q_i v^k w \ \vdash^* \ q_j w \ \vdash \ q_r \ \in F$$

qed.

<u>Re-state Pumping Lemma</u>

<u>If</u> L **is a <u>DFA Language</u>**

<u>then</u>

($L = L(M)$ **for some** M **of** p **states;)**

for <u>every</u> word x **of length** $\ge p$ **in** L**,**

there <u>exists one</u> decomposition $x = uvw$

which satisfies

$(i)\ \ |uv| \le p,$

$(ii)\ \ |v| \ge 1,$

$(iii)\ \ uv^k w \in L(M)$ **for all** $k \ge 0.$

Comments: The DFA pumping lemma shows a property of DFA languages. It can be used possitively; but it is mainly used negatively to show that some languages are not in $\mathcal{L}_{DFA}$. The DFA pumping lemma gives a necessary condition for DFA languages. The condition is not sufficient.

Review of logic:

if A **then** $B \Leftrightarrow$ **if** $\neg B$ **then** $\neg A$

if A **then** $B \not\Leftrightarrow$ **if** B **then** A

if A **then** $B \not\Leftrightarrow$ **if** $\neg A$ **then** $\neg B$

Example

if it is sugar then it is sweet.

if it is not sweet then it is not sugar.

if it is sweet then it is sugar.

So, pumping lemma can be used to show that a language is not a DFA language, but cannot be used to show that a language is a DFA language.

Non-DFA Language

$L = \{a^i b^i \mid i \geq 0\}$ **is not a DFA Language**

Proof (I):

Assume L **is a DFA language. Then** $L = L(M)$ **for some DFA** $M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **with** $|Q| = p$**. Consider** $x = a^t b^t,\ \ |x| \geq p$. **By DFA pumping lemma, there is a decomposition**

$$x = uvw$$

which satisfies (i), (ii), and (iii).

Consider <u>**all the possible**</u> **decompositions**

$$\begin{array}{ll} \underline{\text{Case 1}}: & v = a^k,\ \ k \geq 1.\ k \leq t \\ & \textbf{But } uv^0w = a^{t-k}b^t \notin L. \\ \underline{\text{Case 2}}: & v = b^k,\ \ k \geq 1. \\ & uv^0w = a^t b^{t-k} \notin L. \\ \underline{\text{Case 3}}: & v = a^k b^l,\ \ k, l \geq 1. \\ & \textbf{Then } uv^2w = a^t b^l a^k b^t \notin L. \end{array}$$

All the possible decompositions fail. Therefore, L **is not a DFA language.**

Proof (II):

. . . .

$|Q| = p$.

Consider $x = a^p b^p$ **in** L. **Obviously,** $|x| \geq p$.

By DFA pumping lemma, there is a decomposition

$$x = uvw$$

that satisfies (i), (ii), and (iii).

The only decompositions that satisfy (i) and (ii) are the following:

$$v = a^k \textbf{ for } p \geq k \geq 1.$$

But $uv^0w = a^{p-k}b^p \notin L$.
So, there is no such decomposition.
L **is not a DFA language.**

Notes on proving that a language is not a DFA language by pumping lemma

1) **Assume L is a DFA language.**
   Then we have a constant p.
2) **Choose a word in** L **of length** $\geq p$.
3) **Consider all the possible decompositions of** x. **If none of them satisfy (i), (ii), and (iii) at the same time, then conclude that such decomposition does not exist.**

So, L **is not a DFA language.**

$L = \{a^{2^n} \mid n \geq 0\}$ **is not a DFA language.**

Proof:

(1) Assume $L = L(M)$ **for some** $M = (Q, \Sigma, \delta, s, F)$ **with** $|Q| = p$**.**

(2) Consider $x = a^{2^{p+1}},\ |x| \geq p.$

(3) By DFA P.L., there is a decomposition

$$x = uvw$$

satisfying (i), (ii), and (iii).

The only decompositions of x **which satisfy (i) and (ii) are the following:**

$$v = a^k, \quad p \geq k \geq 1.$$

Notice that $uw = a^{2^{p+1}-k}$**.**

$2^{p+1} > 2^{p+1} - k \geq 2^{p+1} - p > 2^{p+1} - 2^p = 2^p$

i.e., $2^{p+1} > 2^{p+1} - k > 2^p.$

$$|a^{2^{p+1}}| > |uw| > |a^{2^p}|.$$

So, $uw \notin L$**. It is contrary to (iii), so** L **is not a DFA language.**

6. DFA Finiteness

INSTANCE: A DFA $M = (Q, \Sigma, \delta, s, F)$
QUESTION: Is $L(M)$ **finite?**
It is decidable.

Theorem $L(M)$ **is finite iff**
M **accepts no words** x **that satisfy**

$$|Q| \leq |x| < 2|Q|$$

Proof: If: Assume M **accepts no words** x **with** $|Q| \leq |x| < 2|Q|$ **and** $L(M)$ **is infinite. Since** $L(M)$ **is infinite, it must accept some word** x **with** $|x| \geq 2|Q|$**. By P.L.,** $x = uvw$ **with** $|Q| \geq |v| \geq 1$**, and** $x' = uw$ **is in** $L(M)$**. Notice that** $|x| > |x'| \geq |x| - |Q|$**. If** $|x'| \geq 2|Q|$**, then repeat this process.**
Otherwise $2|Q| > |x'| \geq |Q|$**. (i.e.,** $2|Q| - |Q|$**).**
This is a contradiction.

Only if: By P.L., $x = uvw,\ |v| \geq 1$**.**
Since $uv^iw \in L(M)$ **for all** $i \geq 0$**,** $L(M)$ **is infinite.**

Theorem **The family of DFA languages is closed under morphisms.**

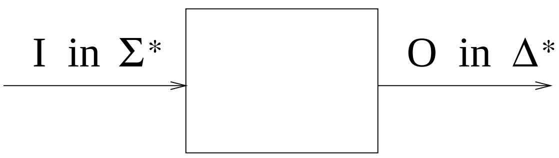
Proof:

Let $L \subseteq \Sigma^*$ **be a DFA language. Then there is a DFA** $M = (Q, \Sigma, \delta, s, F)$ **such that** $L(M) = L$**.**
Let $f : \Sigma^* \to \Delta^*$ **be a morphism.**
We are going to show that
$f(L) = \{f(x) \mid x \in L\}$ **is a DFA language.**
Construct $M' = (Q, \Delta, \delta', s, F)$ **by defining**
$(p, f(a), q)$ **is in** δ' **iff** $\delta(p, a) = q$**.**

M' **is a lazy FA. So, L(M') is a DFA language. It is not difficult to show that** $L(M') = f(L)$**. Therefore,** $f(L)$ **is a DFA language.** *qed*

Finite Transducers and Finite Transductions (Translations)

In automata and language theory, a machine with input and output is called a transducer.



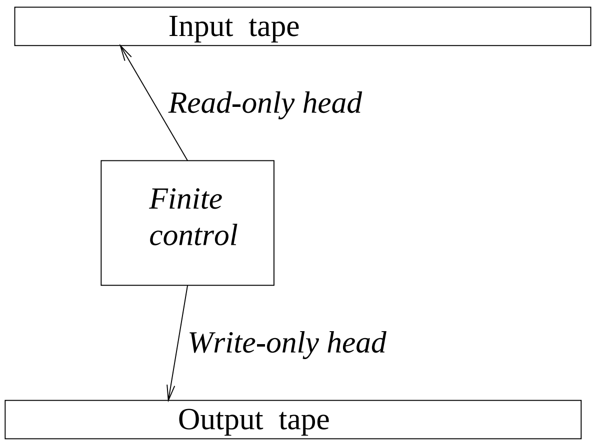
A transducer is single-valued if it produces at most one output word for each input word. It is multi-valued otherwise.

A transducer defines
— a function $f : \Sigma^* \to \Delta^*$, if it is single-valued,
—a relation $R \subseteq \Sigma^* \times \Delta^*$, otherwise.

A function or a relation that is defined by a transducer is called a transduction (translation).

Specifically, a finite automaton with <u>output</u> is called a <u>finite transducer</u>.

FA(DFA, NFA, or ε-NFA)
$+$ output (output alphabet, output tape)
$=$ FT



Formally, a finite transducer (FT) M is a six-tuple $(Q, \Sigma, \Delta, \delta, s, F)$ where

Δ is an output alphabet;

$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Delta^*$ is a finite transition relation;

the others are the same as in a FA.

A configuration vpu is a word in $\Delta^* Q \Sigma^*$.

v — **output till now;**
p — **current state;**
u — **remaining input.**

We write

$$vpau \;\vdash\; vzqu$$

if (p, a, q, z) **is in** δ**.**

$\vdash^i$, $\vdash^+$, $\vdash^*$ **are defined as usual.**

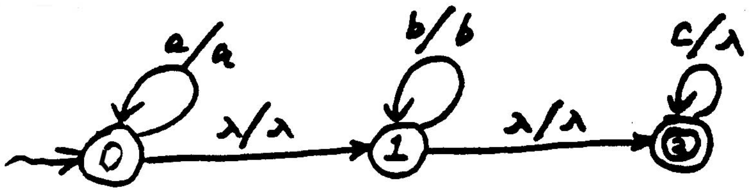
Given $x \in \Sigma^*$**, we say that** y **is an output for** x **w.r.t.** M **if**

$$sx \;\vdash^*\; yf$$

for some $f \in F$**.**

Let $M(x)$ **be the set of all outputs for** x**, then** $M(x) = \{y \mid sx \vdash^* yf$ **for some** $f \in F\}$**.**
The transduction or translation defined by M **is** $T(M) = \{(x, y) \mid x \in \Sigma^*$ **and** $y \in M(x)\}$

Example M:



$$0abbc \ \vdash \ a0bbc \ \vdash \ a1bbc \ \vdash \ ab1bc$$
$$\vdash \ abb1c \ \vdash \ abb2c \ \vdash \ abb2$$

i.e. $0abbc \ \vdash^* \ abb2$

Since $2 \in F$**, we say** $abb \ \in \ M(abbc)$**,**

and $(abbc, abb) \ \in \ T(M)$**.**

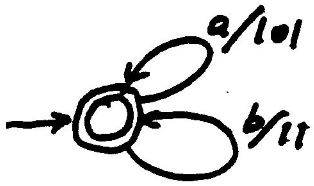
We can see that

$$T(M) = \{(a^i b^j c^k, a^i b^j) \mid i, j, k \geq 0\}$$

Example

Let $h(a) = 101,\ h(b) = 11$.

Encoding:



Decoding: