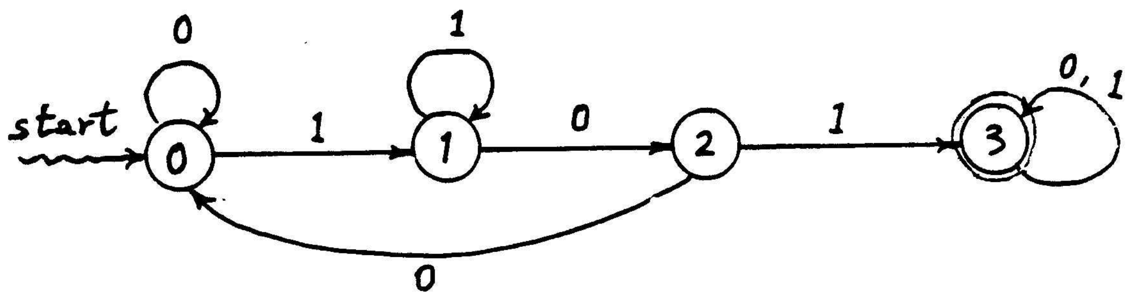
Chapter 2. FINITE AUTOMATA

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)

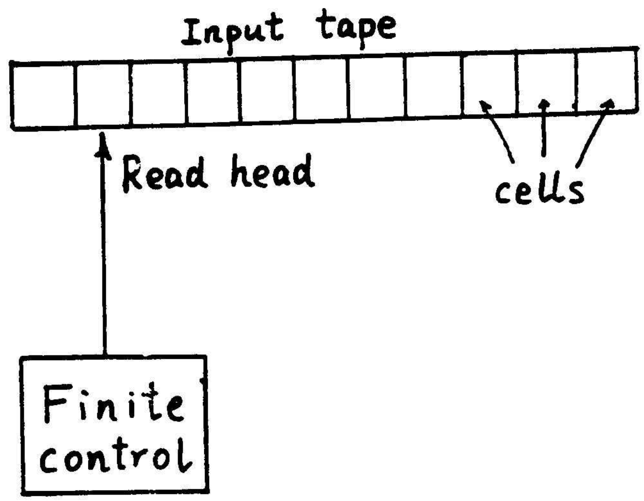
$M = (Q,\ \Sigma,\ \delta,\ s,\ F)$ **where**

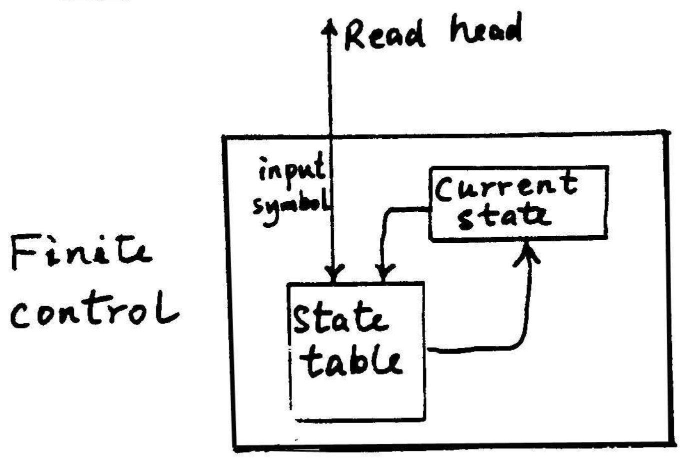
Q	**is a finite nonempty <u>set of states</u>**
Σ	**is the <u>input alphabet</u>**
$\delta : Q \times \Sigma \to Q$	**<u>transition function</u>**
s	**<u>start state</u>**
$F \subseteq Q$	**<u>final state set</u>**

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

There are many examples of FINITE STATE SYSTEMS. A finite automaton is an ABSTRACTION of them.

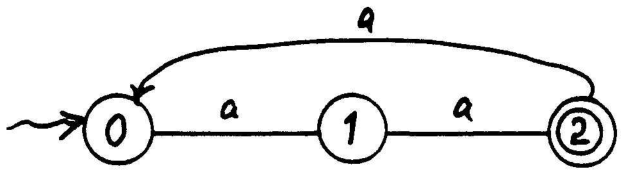
View a DFA as a machine



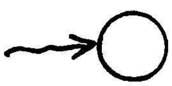


Specifying δ

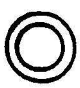
1)



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,

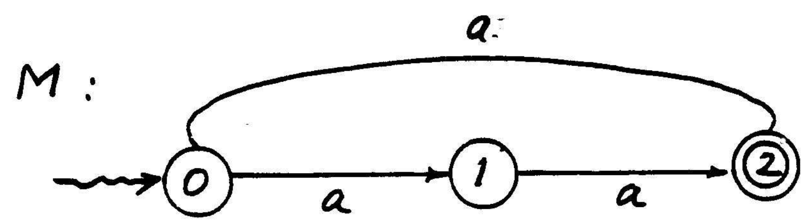
or (ii) the next configuration is undefined.

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

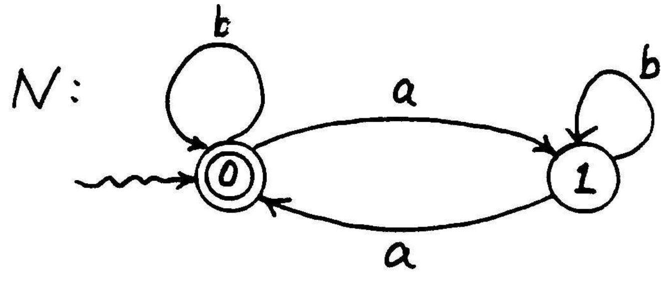
The <u>language</u> of a DFA M, $L(M)$, is defined as:

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

<u>Examples</u>



$L(M) =$



$L(N) =$

DFA membership problem

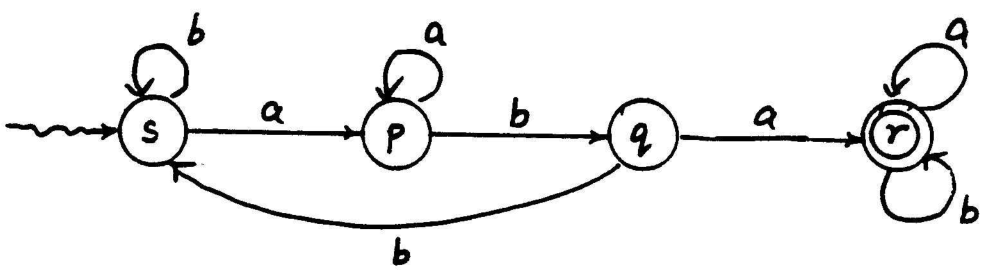
DFA MEMBERSHIP

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.**

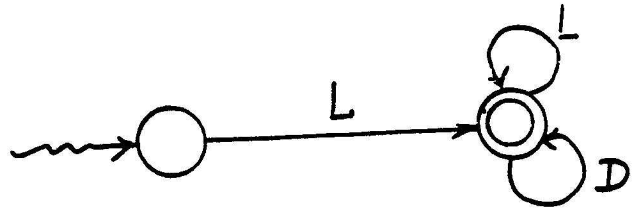
Examples



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

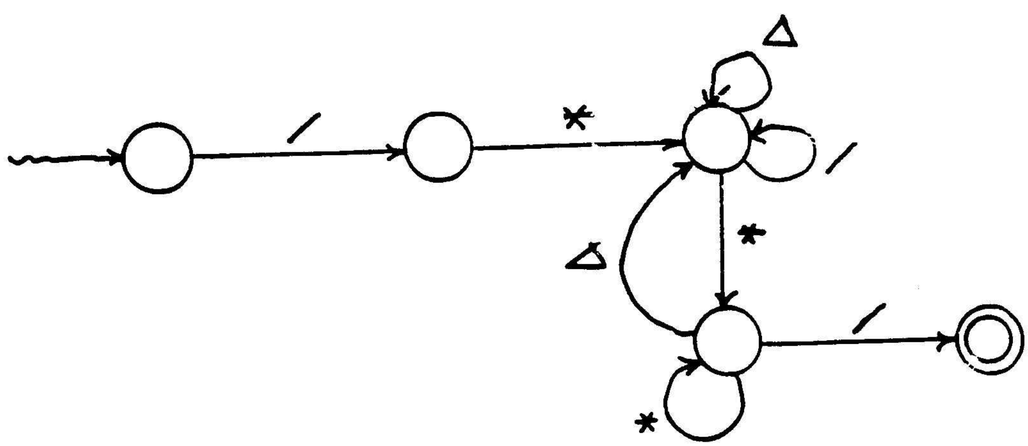
Check: $ababba$
$abbaabbbab$

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.

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

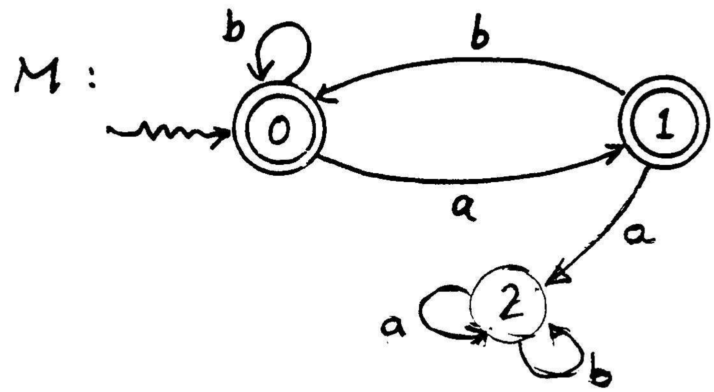


Δ: symbols other than "$*$" and "/"

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

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

<u>Example:</u>



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

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

△ **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:

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

△
$K = \{a^i b^i \mid i \geq 1\}$ **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

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

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

This is a contradiction.

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

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

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

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

Nondeterministic Finite Automata (NFA)

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

same as a DFA except

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

δ **is a finite transition relation.**

In a DFA

δ **is a transition function:**

$\delta : Q \times \Sigma \to Q$

It can be viewed as a relation

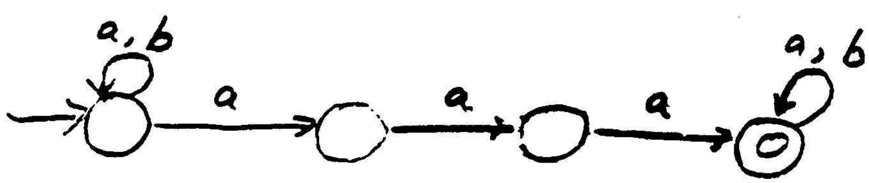
$\delta : Q \times \Sigma \times Q$

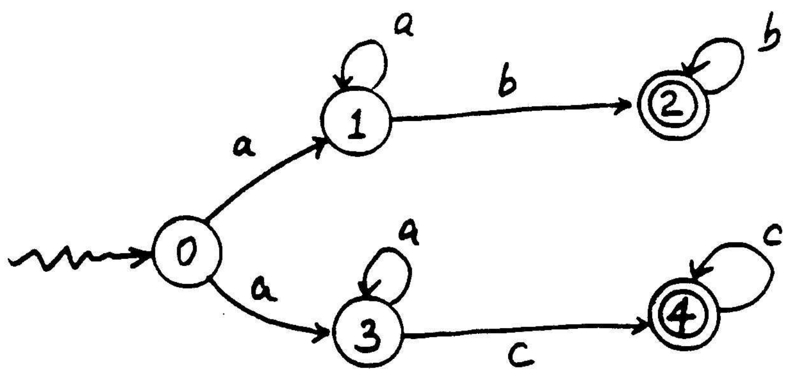
In a NFA, δ **can be be viewed as a function:**

$\delta : Q \times \Sigma \to 2^Q$

Examples:

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:

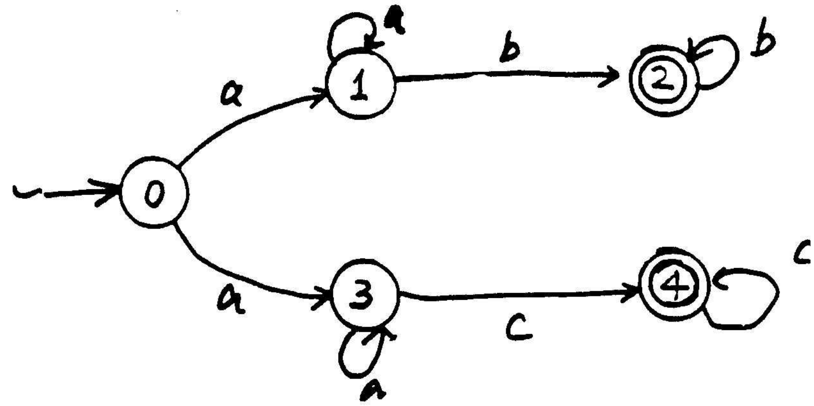
configurations are defined in the same way
Transition (move)

$$px \vdash qy$$

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

Transforming NFA to DFA

Consider the NFA M_1 again



There are only limited number of choices. For example:

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

Why limited number of choices?

The state set is finite.

We summarize the choices at each step by combining all configuration sequences into one "super-conf. sequence".

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

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 deterministic.

A super-configuration has the form

$$Kx$$

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

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

We say that

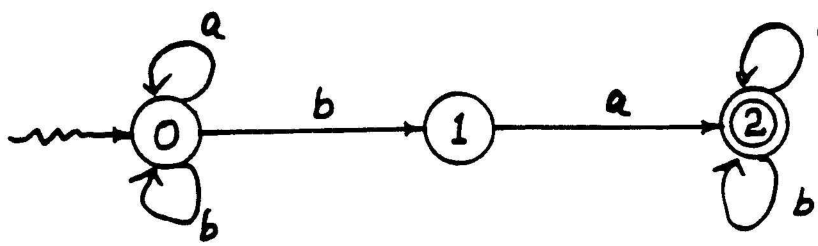
$$Kx \vdash Ny$$

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

$N = \{q \mid (p, a, q) \in \delta$, **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.

Lemma (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 $\underline{K} \subseteq Q$.

$\underline{Kx \vdash^* N \text{ and } Kx \vdash^* P}$

implies

$\underline{P = N}$.

Lemma (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,

$\underline{qx \vdash^* p}$

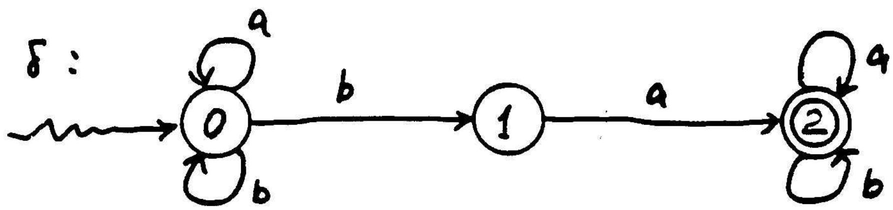
iff $\underline{\{q\}x \vdash^* P}$, for some P with p in P.

<u>Example</u> (Transformation of an NFA to a DFA)

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

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

δ :



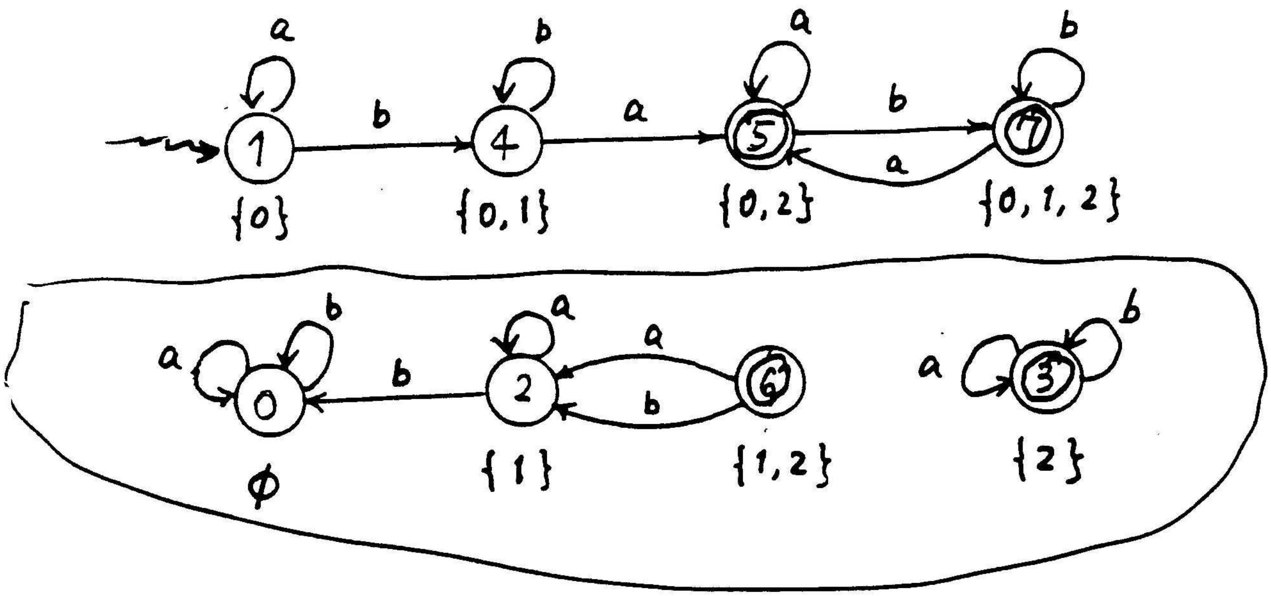
$M' = (Q', \Sigma, \delta', s', F')$ **where**

$$Q' = 2^Q = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\}$$

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

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



$s' = \{0\}$
$F' = \{$

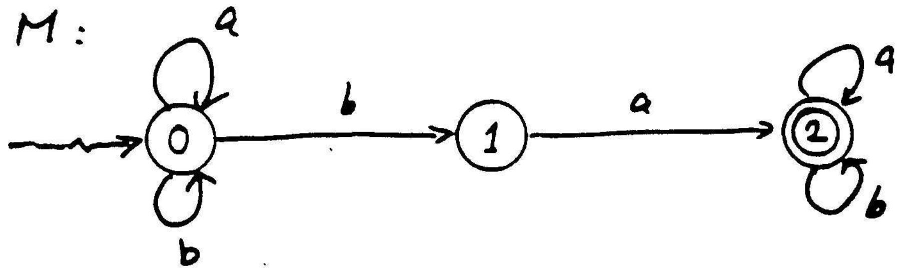
Algorithm NFA to DFA
—The Subset Construction

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** $a \in \Sigma$**,**
$\delta'(K, a) = N$**, if** $Ka \vdash N$ **in** M**.**
end of Algorithm

if $N = \{q \mid (p, a, q) \in \delta$ **and** $p \in K\}$

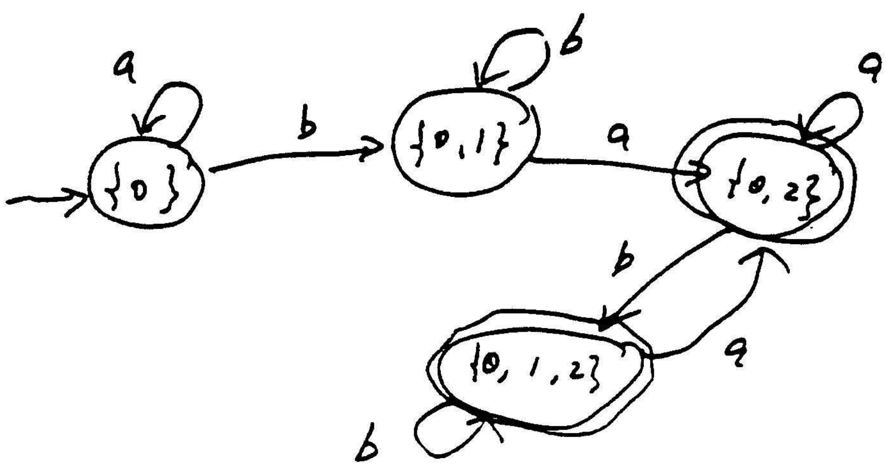


$s' = \{0\}$

current state \ input symbol	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
—The Iterative Subset Construction



<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 **iff**
$\{s\}x \vdash^* P$ **in** M'**.**

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

Theorem

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

Example

Every finite language is accepted by a DFA.