中山大學

Principles of Compiler Construction

Lecture 3 Lexical Analysis (II)

Lecturer: Chang Huiyou

Note that most of these slides were created by:

Prof. Wen-jun LI (School of Software)

Dr. Zhong-mei SHU (Department of Computer Science)

Dr. Han LIN (Department of Computer Science)

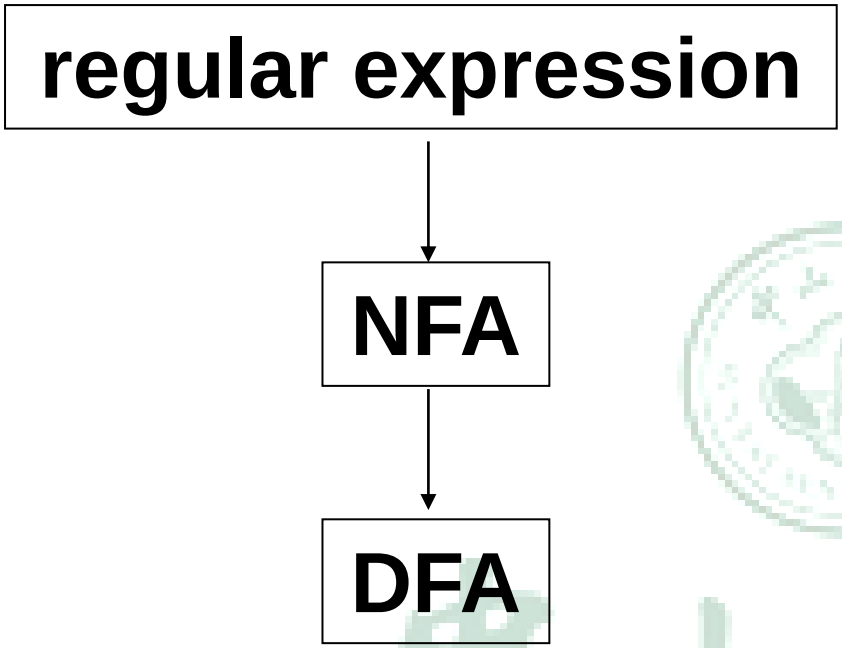
练习

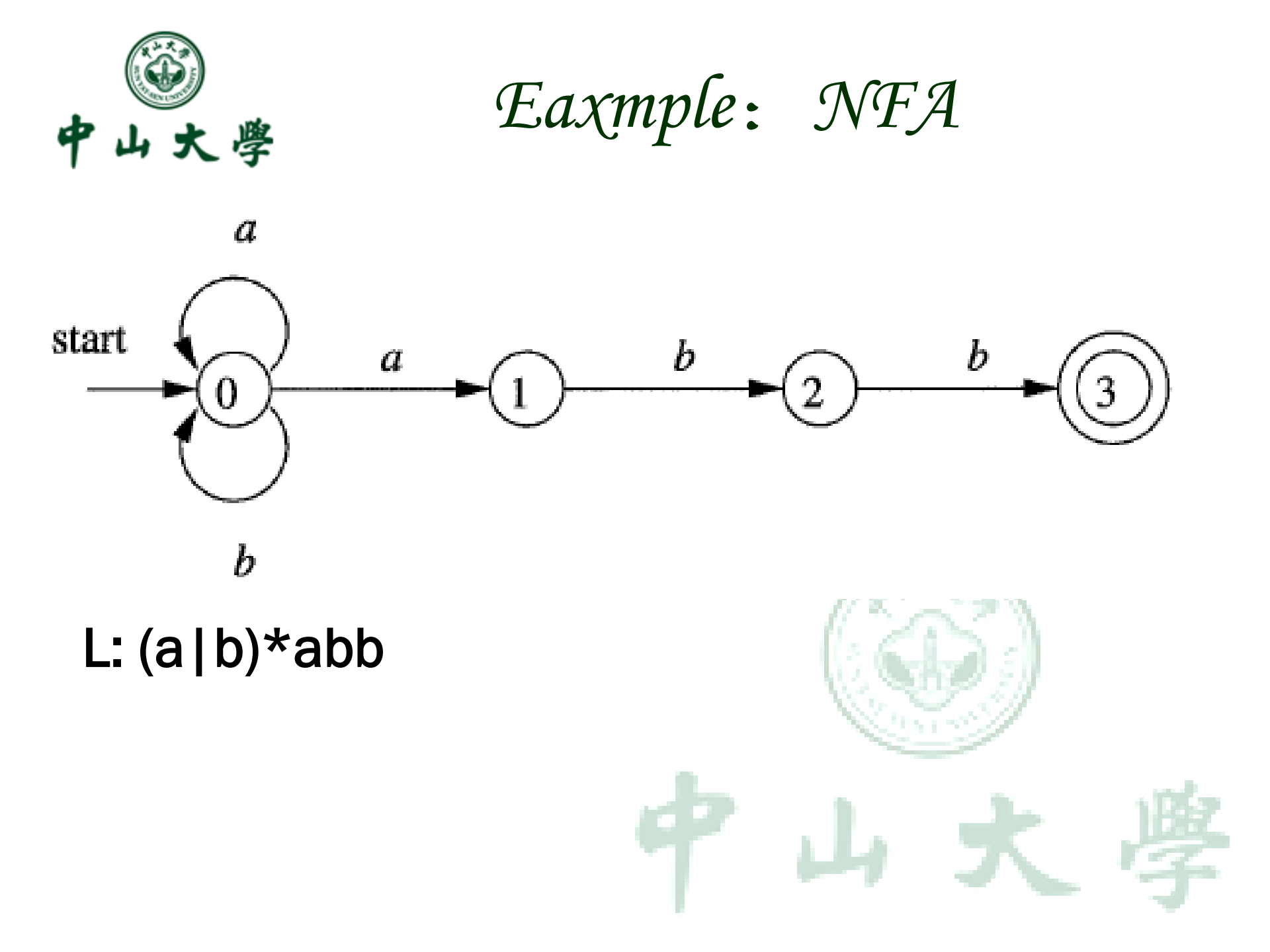
用正则表达式描述下列语言：

- 不以ab开头的所有只含有字母a和b的字符串.

A Big QUESTION

How to identify and process regular expression?





Eaxmple： NFA

L: (a|b)*abb

中山大學

NFA

A *nondeterministic finite automaton* (NFA) consists of:

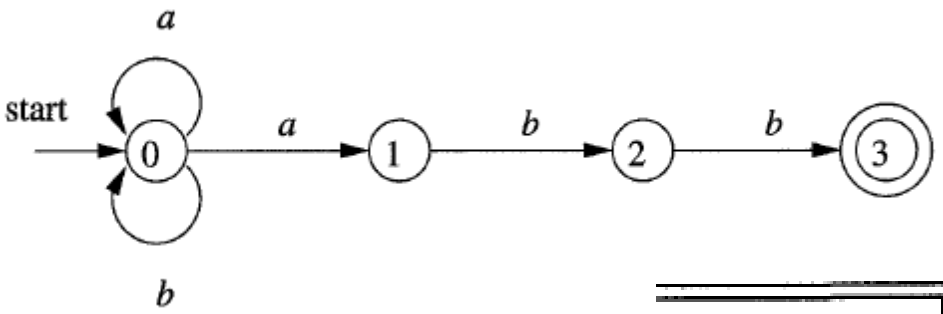
1. A finite set of states S.
2. A set of input symbols Σ, the *input alphabet*. We assume that ϵ, which stands for the empty string, is never a member of Σ.
3. A *transition function* that gives, for each state, and for each symbol in $\Sigma \cup \{\epsilon\}$ a set of *next states*.
4. A state s_0 from S that is distinguished as the *start state* (or *initial state*).
5. A set of states F, a subset of S, that is distinguished as the *accepting states* (or *final states*).

The *language defined (or accepted) by an NFA is the set of strings labeling* some path from the start to an accepting state.

中山大學

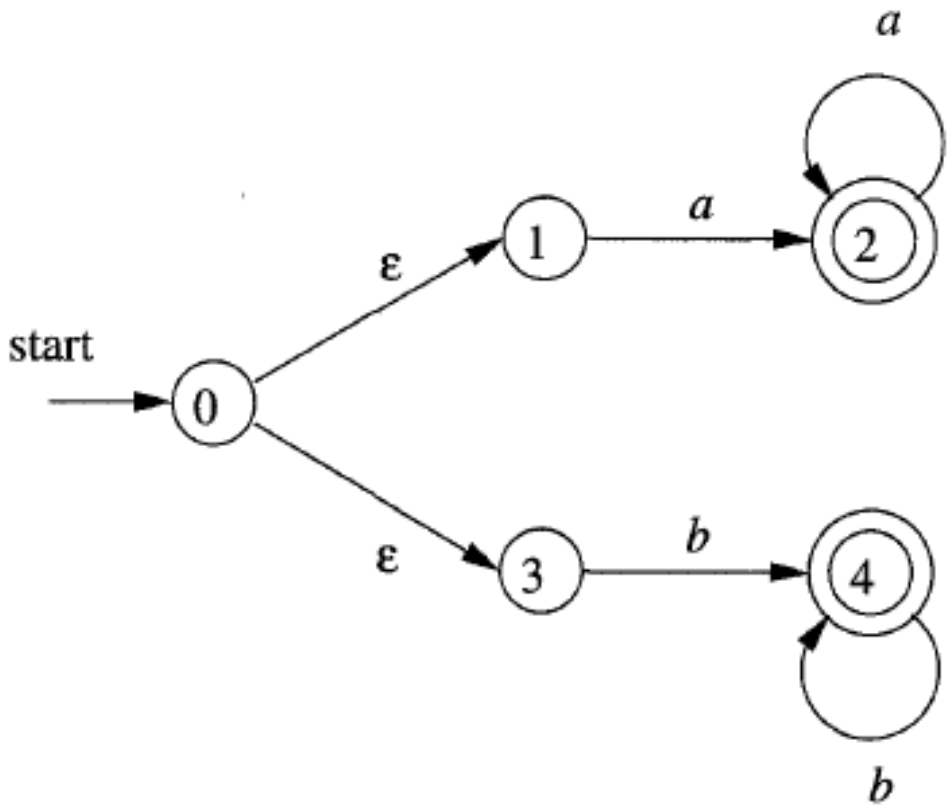
Transition Table

A transition function can be represented by a transition table. Eg:



STATE	a	b	ϵ
0	$\{0, 1\}$	$\{0\}$	$\emptyset$
1	$\emptyset$	$\{2\}$	$\emptyset$
2	$\emptyset$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$

Example: NFA with ϵ



L: aa*|bb*

DFA

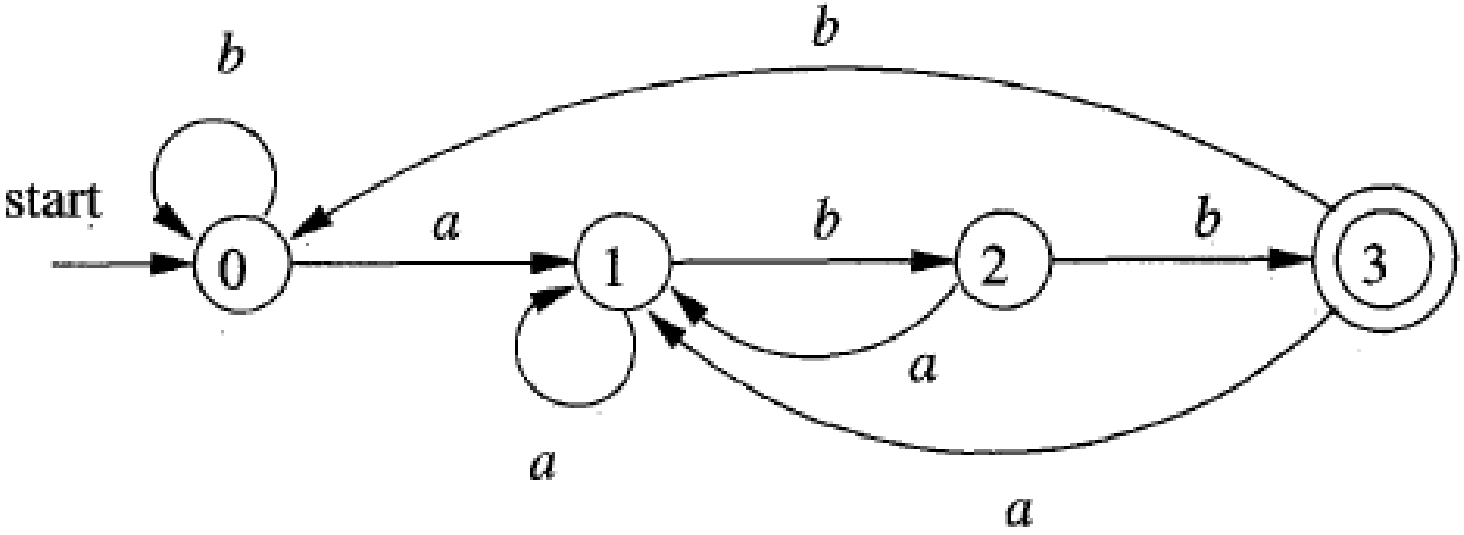
A *deterministic finite automaton (DFA) is a special case of an NFA where:*

1. There are no moves on input ϵ, and
2. For each state s and input symbol a, there is exactly one edge out of s labeled a.

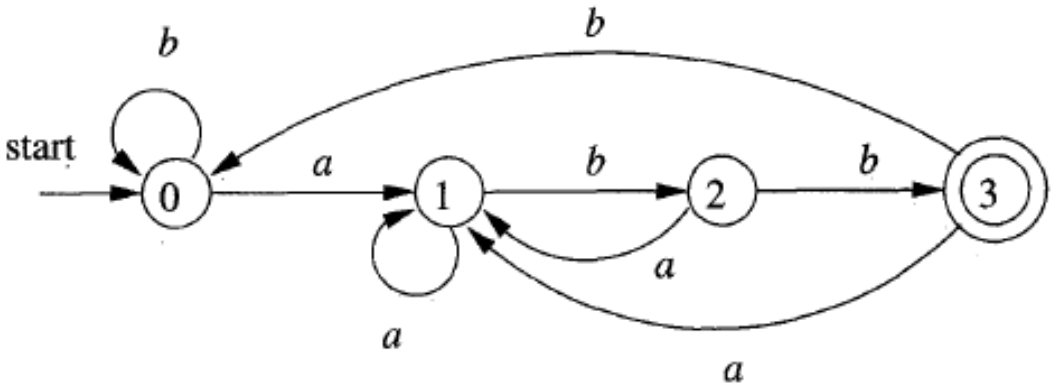
中山大學

Example

(a|b)*abb



Transition Function: Move



Transition function (table): move(s, c)

	a	b
s0	s1	s0
s1	s1	s2
s2	s1	s3
s3	s1	s0

DFA

```
s = s0;
c = nextChar();
while ( c != eof ) {
        s = move(s, c);
        c = nextChar();
}
if ( s is in F ) return "yes";
else return "no";
```

O(|*x*|)，x is input string，|*x*| is the length of x

中山大學

More…

DFA?

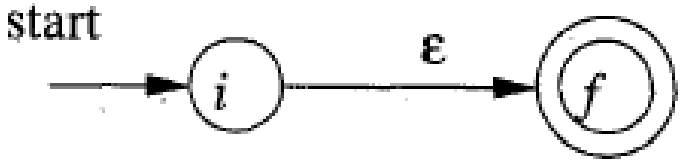
Regular Expression to NFA?

NFA to DFA?

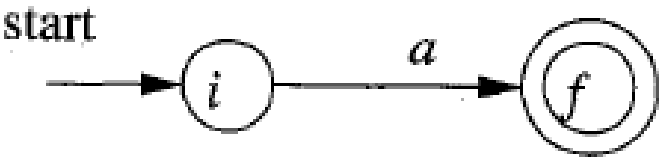
OK, let's try!

From Regular Expression to NFA

BASIS: For expression ϵ construct the NFA



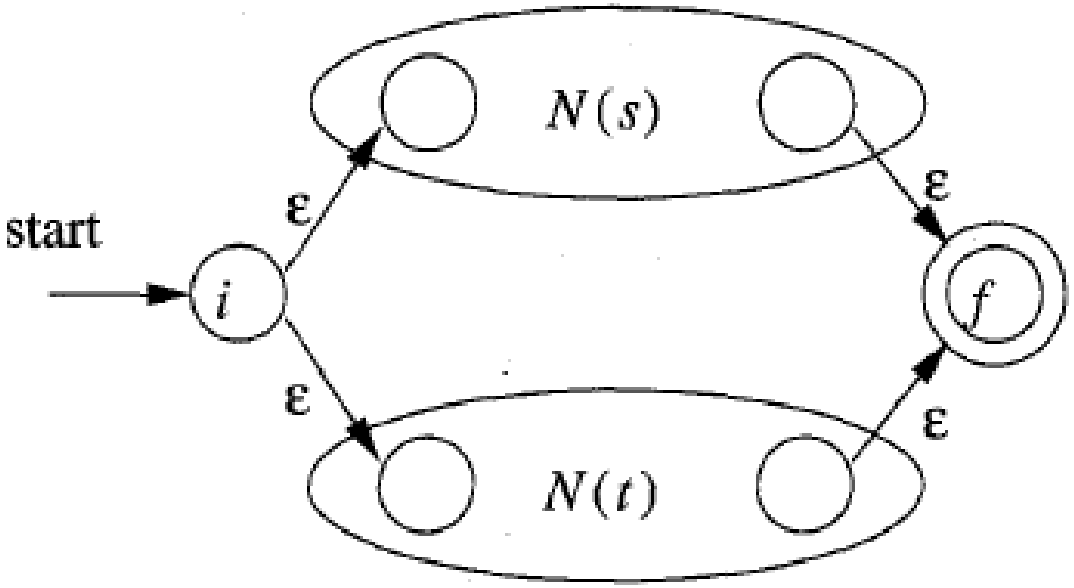
For any subexpression a in Σ, construct the NFA



From Regular Expression to NFA

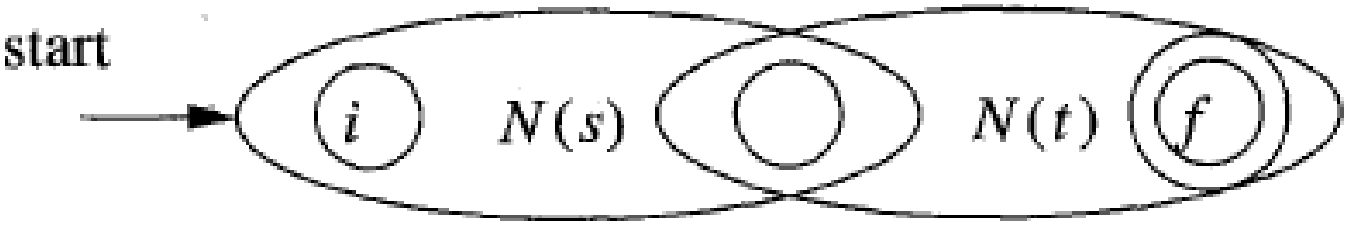
INDUCTION: Suppose $N(s)$ and $N(t)$ are NFA's for regular expressions s and t, respectively.

a) Suppose $r = s|t$. Then $N(r)$, the NFA for r, is constructed as



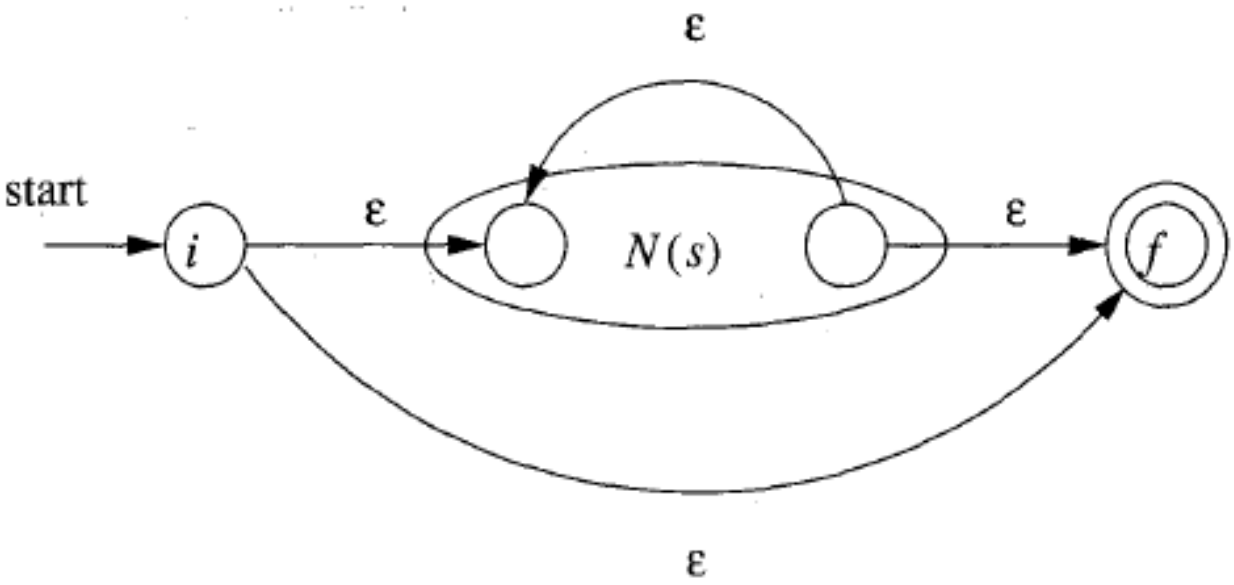
From Regular Expression to NFA

b) Suppose $r = st$. Then construct $N(r)$ as



From Regular Expression to NFA

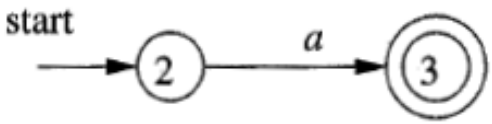
c) Suppose $r = s^*$. Then for r we construct the NFA $N(r)$



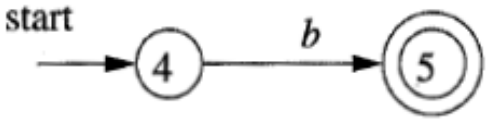
Example

Construct an NFA for (a|b)*abb

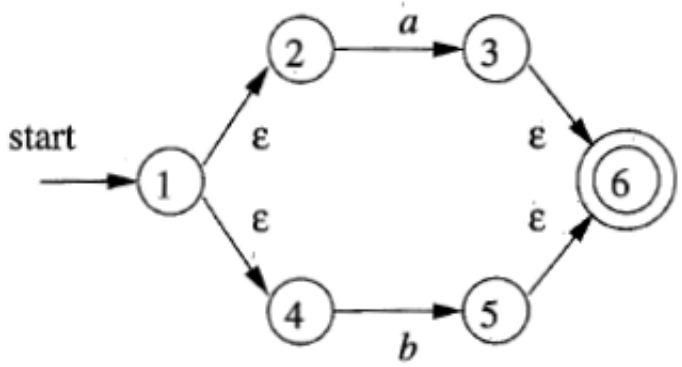
a



b



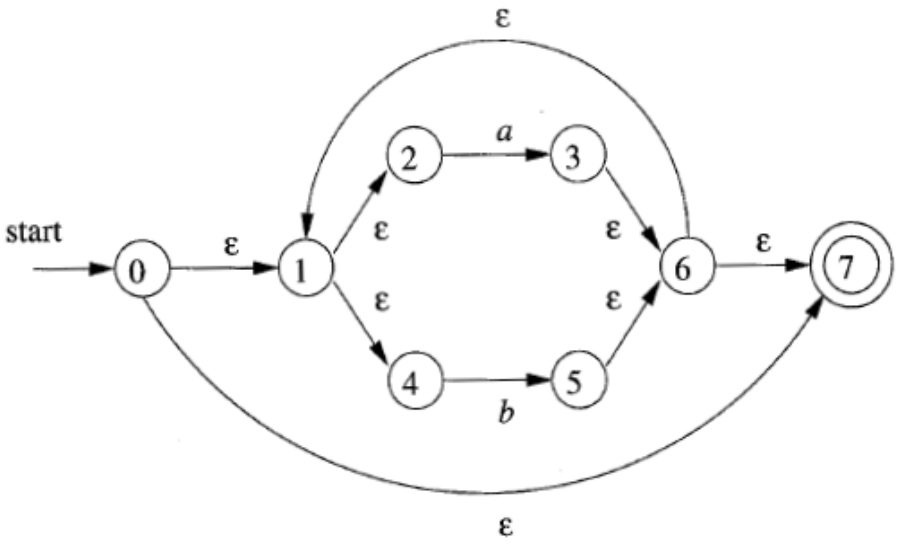
a|b



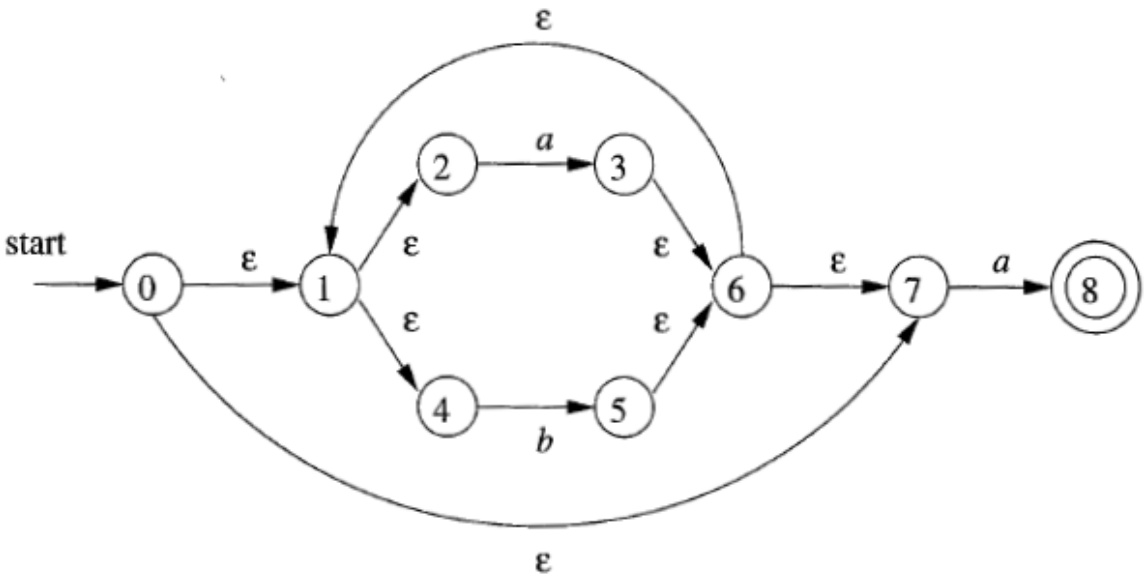
中山大學

Example

(a|b)*

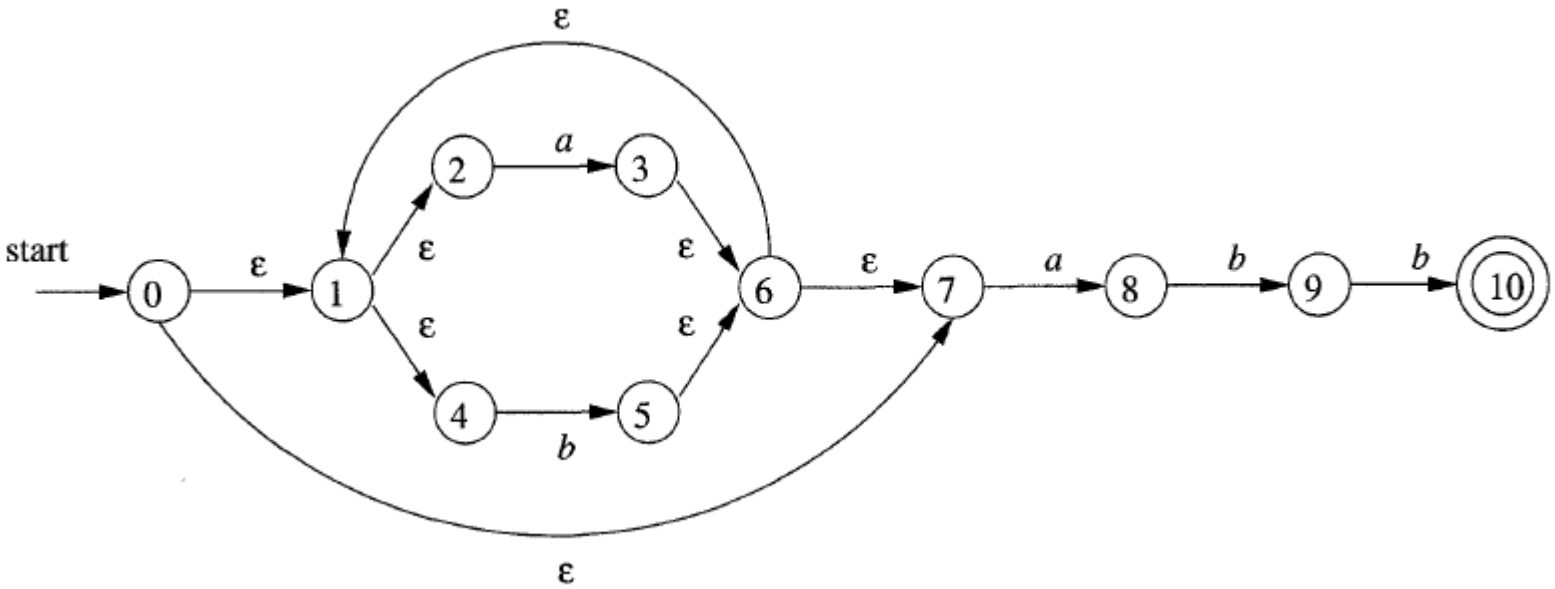


(a|b)*a



Example

(a|b)*abb

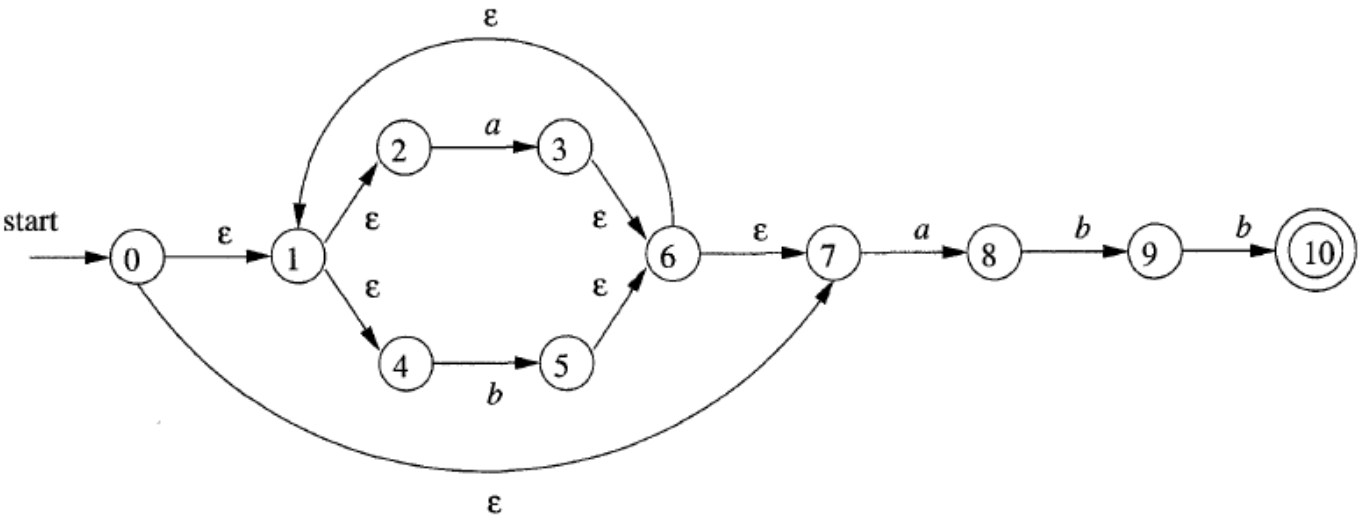


From NFA to DFA

一些记号：

ϵ-*closure*(s)	Set of NFA states reachable from NFA state s on ϵ-transitions alone.

e.g.

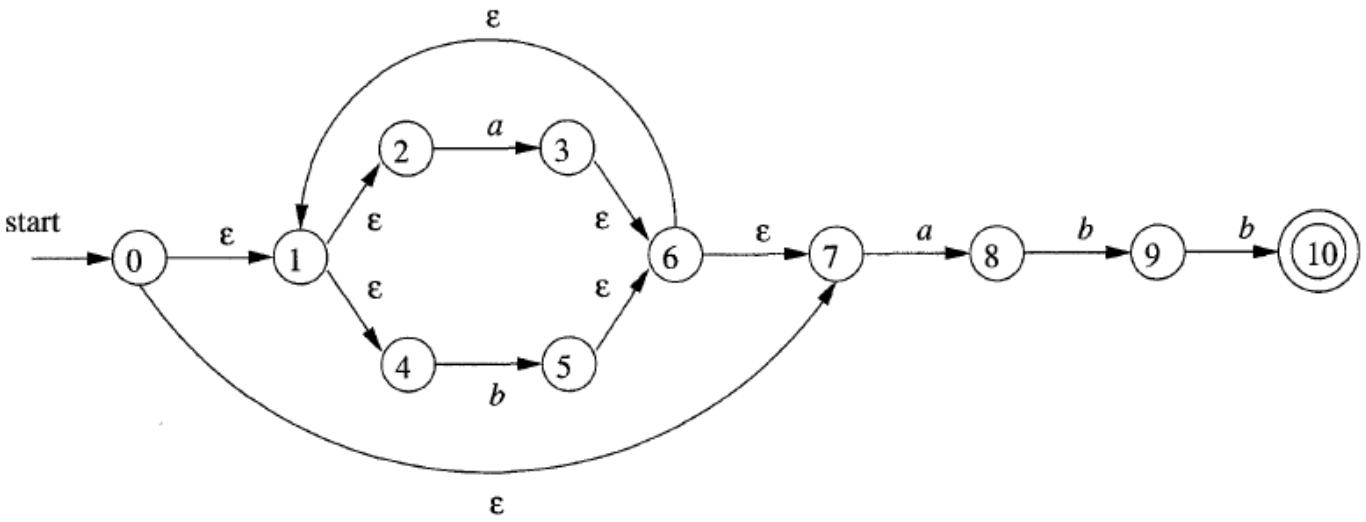


Є-closure(0)={0, 1, 2, 4, 7}

From NFA to DFA

ϵ-*closure*(T)	Set of NFA states reachable from some NFA state s in set T on ϵ-transitions alone; $= \cup_{s \text{ in } T}$ ϵ-*closure*(s).

e.g.

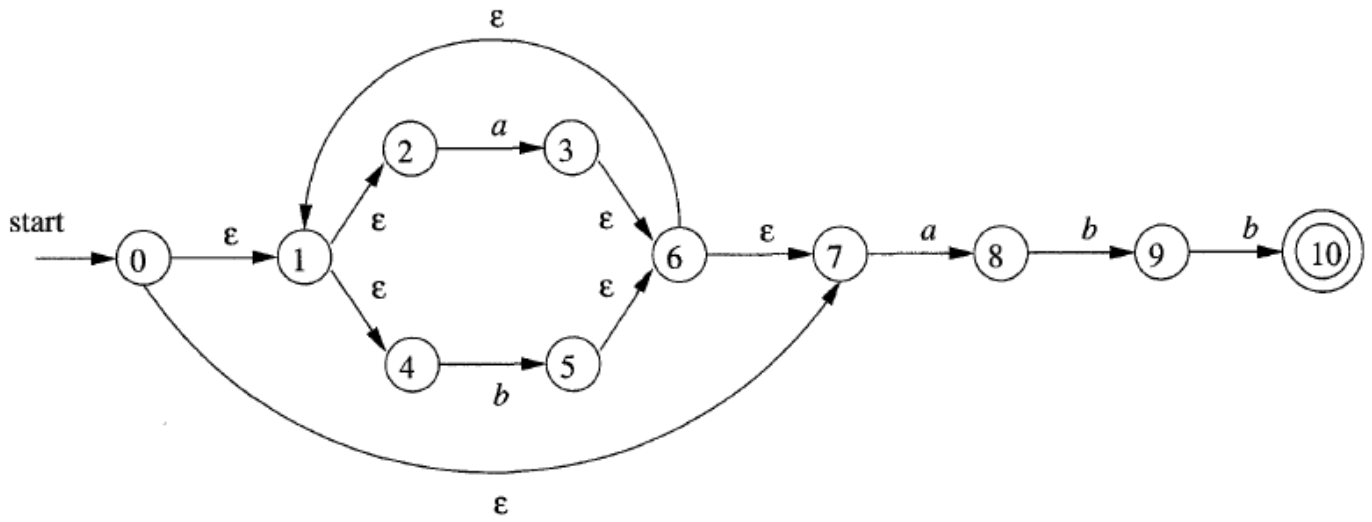


$$\epsilon\text{-}closure(\{3,8\}) = \{1,2,3,4,6,7,8\}$$

From NFA to DFA

$move(T, a)$	Set of NFA states to which there is a transition on input symbol a from some state s in T.

e.g.



move({0, 1, 2, 4, 7}, a) = {3, 8}

Subset Construction

Algorithm:Subset Construction

Input: An NFA *N*

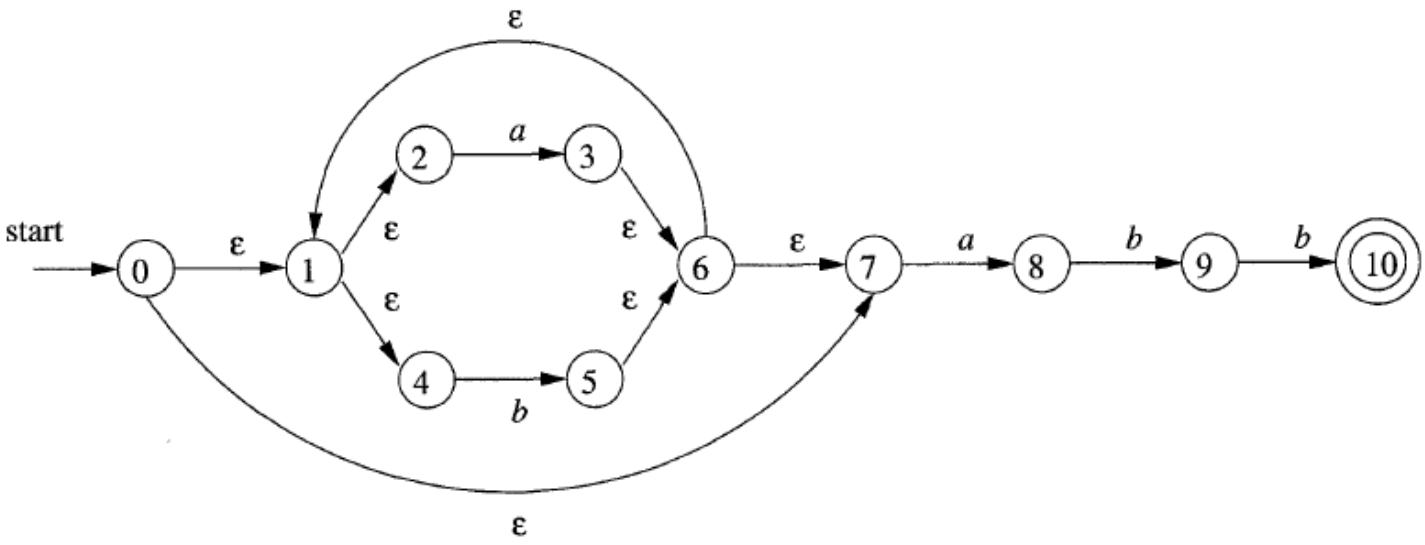
Output: A DFA *D* accepting the same language as *N*

Method: Each state of *D* is a set of NFA states. The algorithm constructs a transition table *Dtran* for *D*.

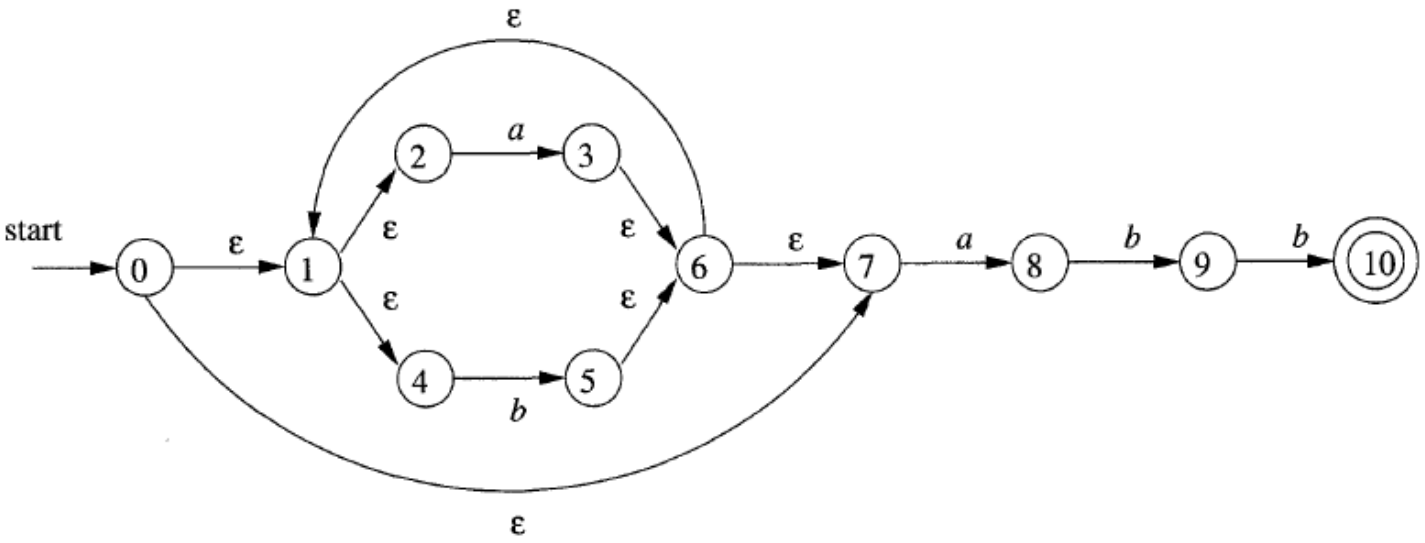
initially, ϵ-*closure*(s_0) is the only state in *Dstates*, and it is unmarked;
while (there is an unmarked state T in *Dstates*) {
    mark T;
    for (each input symbol a) {
        $U = \epsilon$-*closure*(*move*(T, a));
        if (U is not in *Dstates*)
            add U as an unmarked state to *Dstates*;
        Dtran[T, a] = U;
    }
}

Example

initially, ϵ-*closure*(s_0) is the only state in *Dstates*, and it is unmarked;
while (there is an unmarked state T in *Dstates*) {
    mark T;
    for (each input symbol a) {
        $U = \epsilon$-*closure*$(move(T, a))$;
        if (U is not in *Dstates*)
            add U as an unmarked state to *Dstates*;
        $Dtran[T, a] = U$;
    }
}



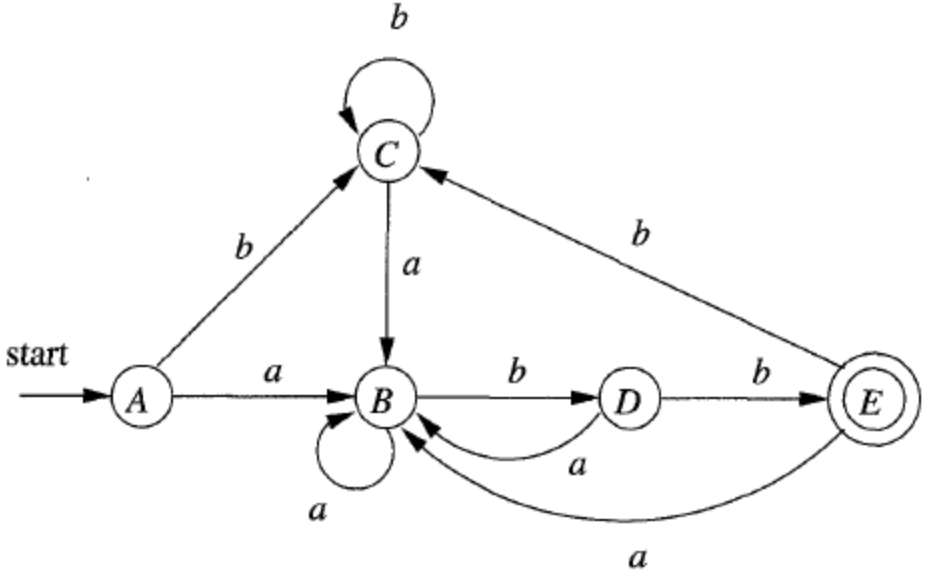
Example



NFA states	DFA state	a	b
{0, 1, 2, 4, 7}	A	B	C
{1, 2, 3, 4, 6, 7, 8}	B	B	D
{1, 2, 4, 5, 6, 7}	C	B	C
{1, 2, 4, 5, 6, 7, 9}	D	B	E
{1, 2, 4, 5, 6, 7, 10}	E	B	C

Example

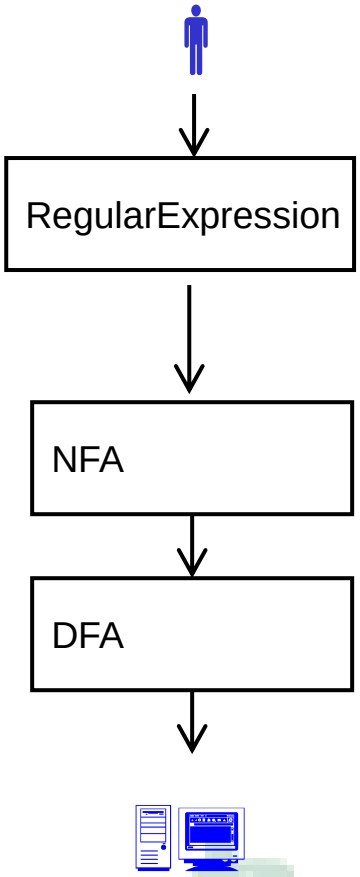
NFA states	DFA state	a	b
{0, 1, 2, 4, 7}	A	B	C
{1, 2, 3, 4, 6, 7, 8}	B	B	D
{1, 2, 4, 5, 6, 7}	C	B	C
{1, 2, 4, 5, 6, 7, 9}	D	B	E
{1, 2, 4, 5, 6, 7, 10}	E	B	C



Computing ϵ*-closure*(T)

```
push all states of T onto stack;
initialize ϵ-closure(T) to T;
while ( stack is not empty ) {
        pop t, the top element, off stack;
        for ( each state u with an edge from t to u labeled ϵ )
                if ( u is not in ϵ-closure(T) ) {
                        add u to ϵ-closure(T);
                        push u onto stack;
                }
}
```

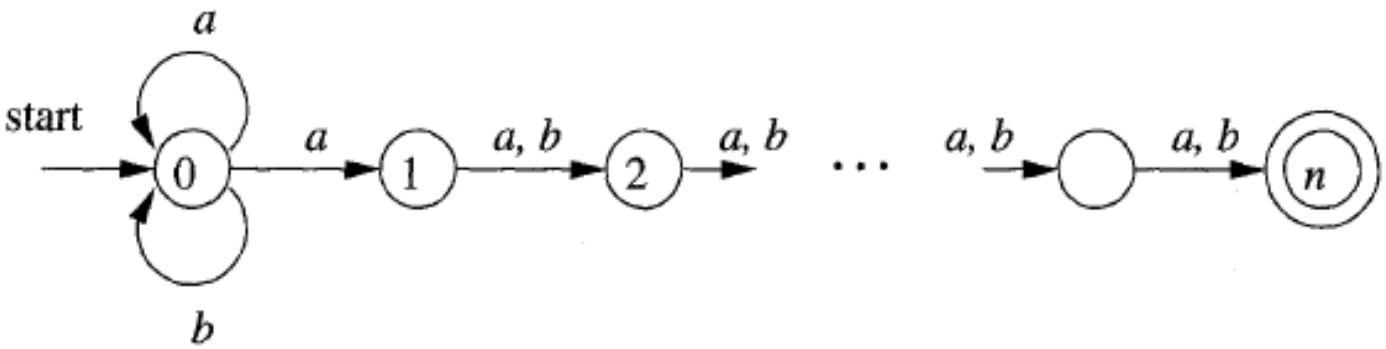
conclusion



From NFA to DFA

Disadvantage

$$L_n = (\mathbf{a}|\mathbf{b})^*\mathbf{a}(\mathbf{a}|\mathbf{b})^{n-1}$$



2^n ?

中山大學

NFA

INPUT: An input string x terminated by an end-of-file character **eof**. An NFA N with start state s_0, accepting states F, and transition function *move*.

OUTPUT: Answer "yes" if M accepts x; "no" otherwise.

```
1)  S = ε-closure(s0);
2)  c = nextChar();
3)  while ( c != eof ) {
4)          S = ε-closure(move(S, c));
5)          c = nextChar();
6)  }
7)  if ( S ∩ F != ∅ ) return "yes";
8)  else return "no";
```

Discussion

DFA?

NFA?

中山大學

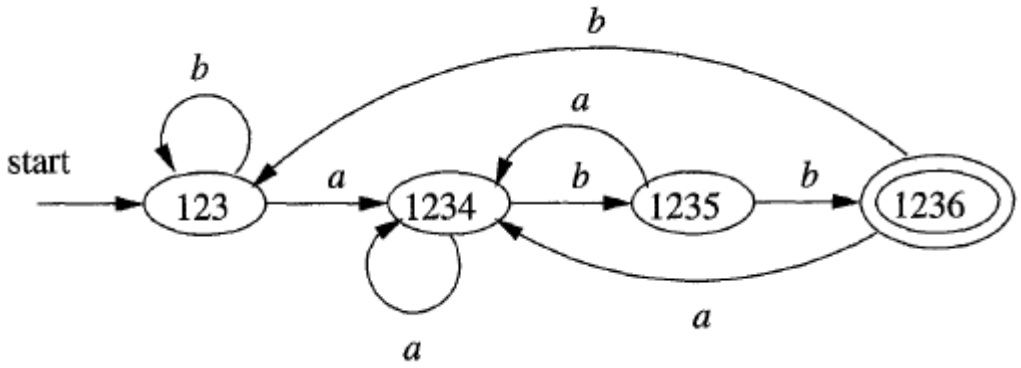
重要结论

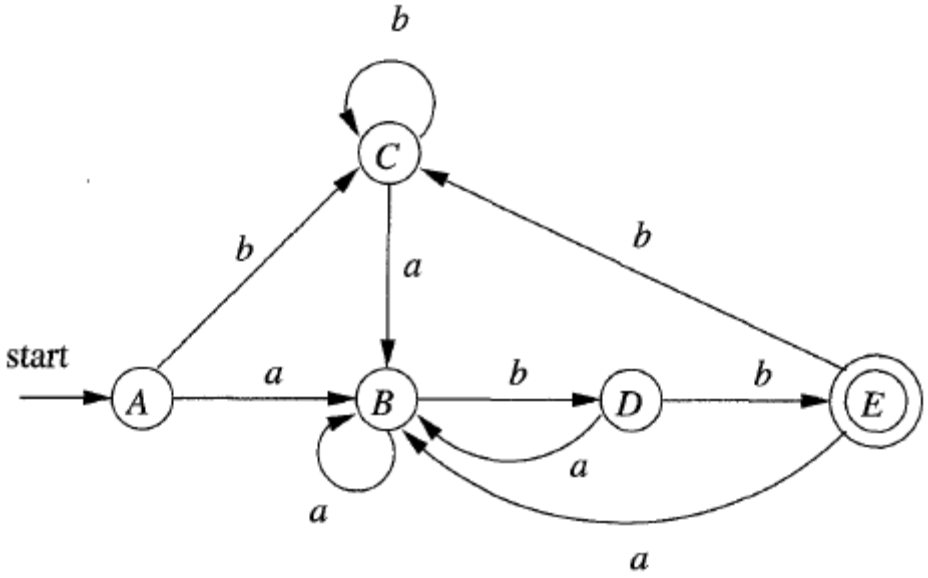
定理：对任何DFA *D*都存在一个正则表达式*r*，使得L(D)=L(r).

推论：对任何NFA *N*都存在一个正则表达式*r*，使得L(N)=L(r).

结论：DFA，NFA和正则表达式三者的描述能力是一样的.

比较





上图比下图少了一个状态

问题：如何使得DFA的状态数最少？

中山大學

DFA的最小化

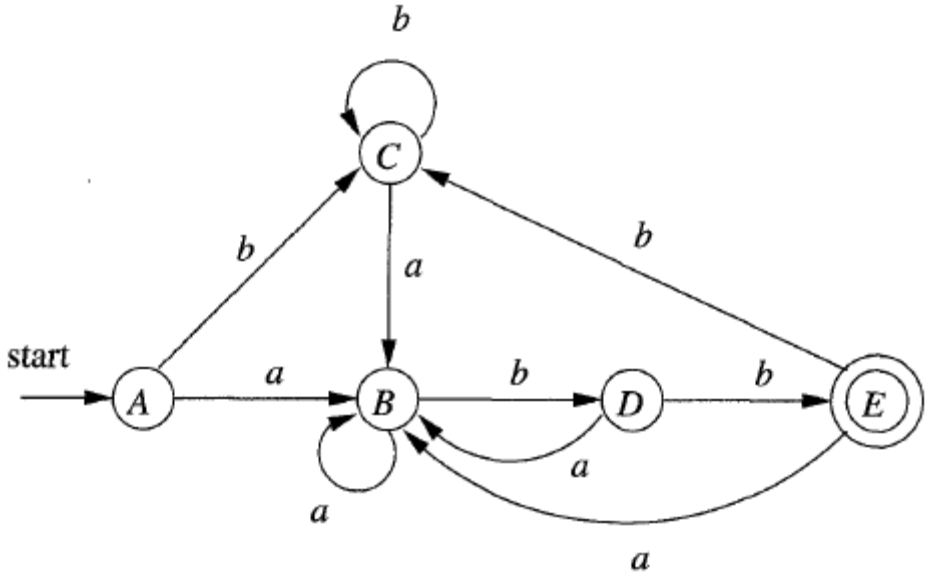
INPUT: A DFA D with set of states S, input alphabet Σ, state state s_0, and set of accepting states F.

OUTPUT: A DFA D' accepting the same language as D and having as few states as possible.

Basic Idea

String x distinguishes state s from state t if exactly one of the states reached from s and t by following the path with label x is an accepting state.

State s is distinguishable from state t if there is some string that distinguishes them.



例如，左图中字符串bb区分状态B和C

中山大學

最小化DFA的算法

1. Start with an initial partition Π with two groups, F and $S - F$, the accepting and nonaccepting states of D.

2. Apply the procedure of Fig. 3.64 to construct a new partition Π_{new}.

```
initially, let Π_new = Π;
for ( each group G of Π ) {
        partition G into subgroups such that two states s and t
                are in the same subgroup if and only if for all
                input symbols a, states s and t have transitions on a
                to states in the same group of Π;
        /* at worst, a state will be in a subgroup by itself */
        replace G in Π_new by the set of all subgroups formed;
}
```

Figure 3.64: Construction of Π_{new}

3. If $\Pi_{\text{new}} = \Pi$, let $\Pi_{\text{final}} = \Pi$ and continue with step (4). Otherwise, repeat step (2) with Π_{new} in place of Π.

中山大學

最小化DFA的算法（续）

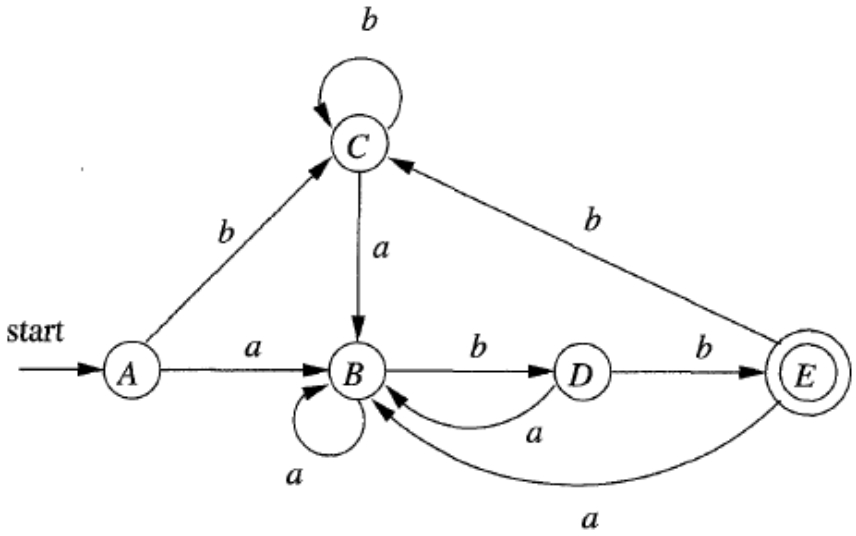
4. Choose one state in each group of Π_{final} as the *representative* for that group. The representatives will be the states of the minimum-state DFA D'. The other components of D' are constructed as follows:

   (a) The state state of D' is the representative of the group containing the start state of D.

   (b) The accepting states of D' are the representatives of those groups that contain an accepting state of D. Note that each group contains either only accepting states, or only nonaccepting states, because we started by separating those two classes of states, and the procedure of Fig. 3.64 always forms new groups that are subgroups of previously constructed groups.

   (c) Let s be the representative of some group G of Π_{final}, and let the transition of D from s on input a be to state t. Let r be the representative of t's group H. Then in D', there is a transition from s to r on input a. Note that in D, every state in group G must go to some state of group H on input a, or else, group G would have been split according to Fig. 3.64.

例子



最小化得

STATE	a	b
A	B	A
B	B	D
D	B	E
E	B	A

中山大學

作业

See you next time!