Compiler Design

Fatemeh Deldar

Isfahan University of Technology

1402-1403

Conversion of an NFA to a DFA

- **The subset construction**

```
initially, ϵ-closure(s0) 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 = ϵ-closure(move(T, a));
                if ( U is not in Dstates )
                        add U as an unmarked state to Dstates;
                Dtran[T, a] = U;
        }
}
```

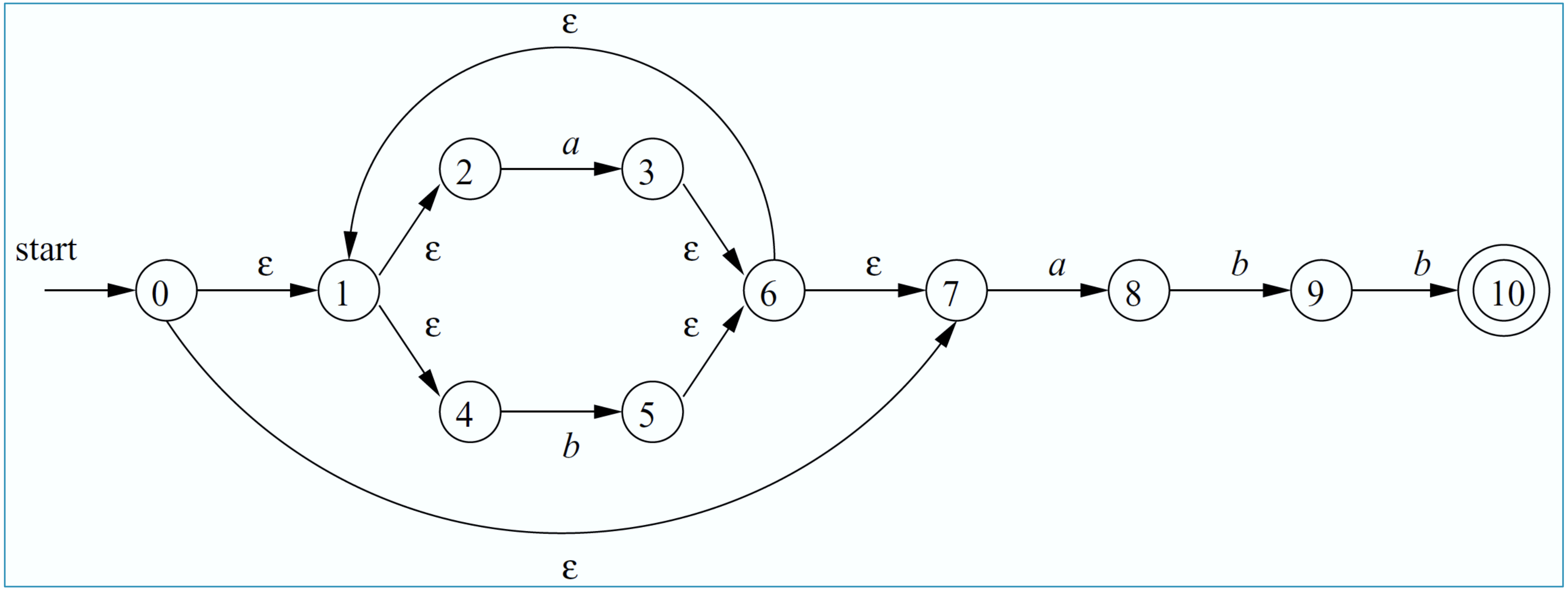
Conversion of an NFA to a DFA

- **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;
                }
}
```

Conversion of an NFA to a DFA

- **Example**



Conversion of an NFA to a DFA

- **Example**
  - The start state A of the equivalent DFA is ϵ-closure(0), or A = {0, 1, 2, 4, 7}

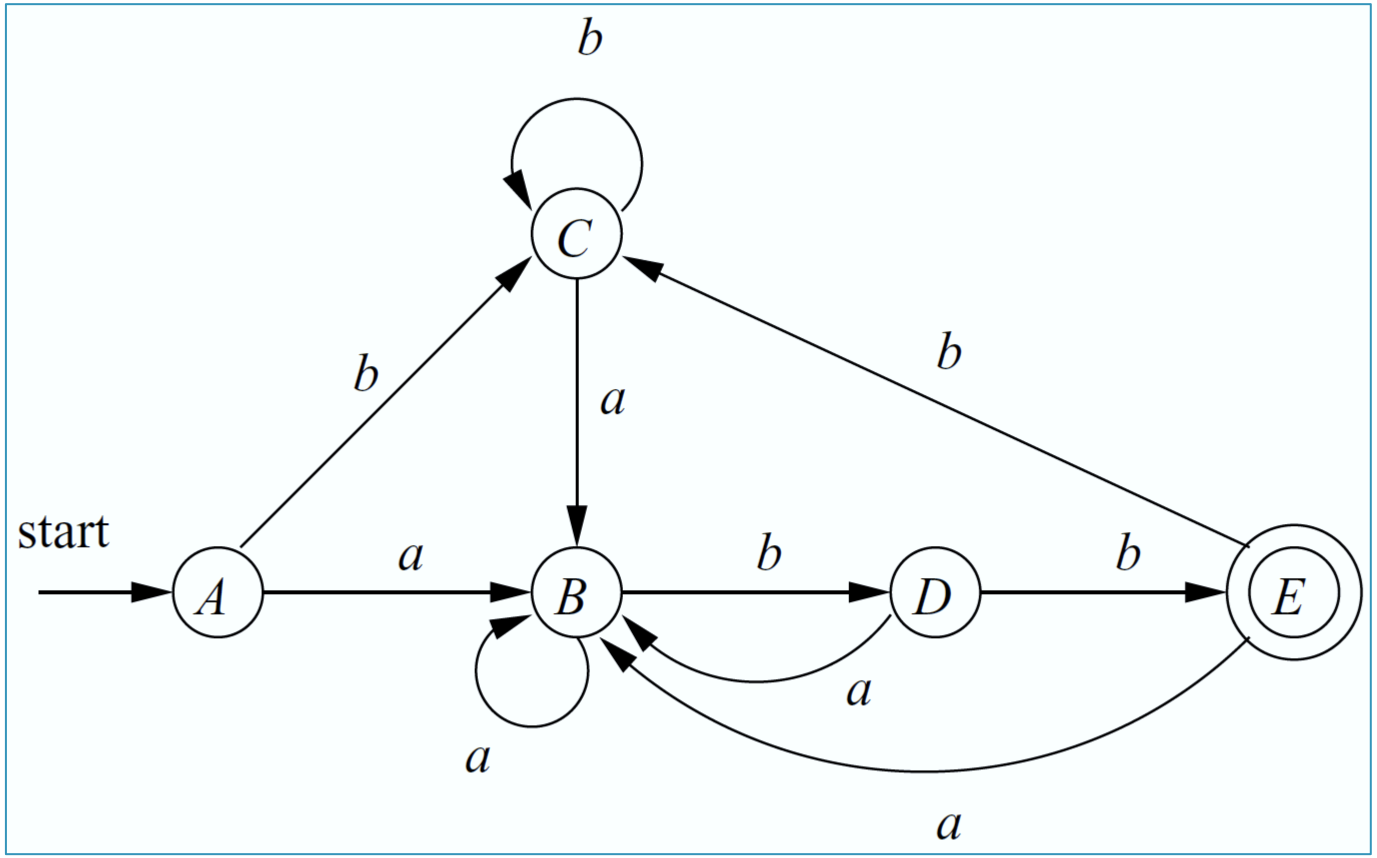
$$Dtran[A, a] = \epsilon\text{-}closure\big(move(A, a)\big) = \epsilon\text{-}closure(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\}$$

$$Dtran[A, b] = \epsilon\text{-}closure(\{5\}) = \{1, 2, 4, 5, 6, 7\}$$

NFA STATE	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

Conversion of an NFA to a DFA

- **Example**

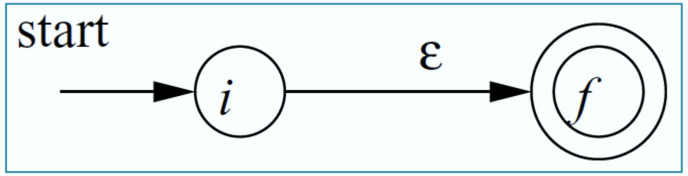


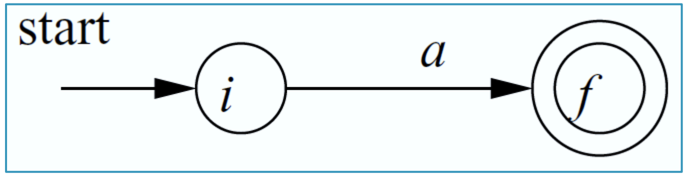
Simulation of an NFA

```
S = ε-closure(s_0);
c = nextChar();
while ( c != eof ) {
        S = ε-closure(move(S, c));
        c = nextChar();
}
if ( S ∩ F != ∅ ) return "yes";
else return "no";
```

Construction of an NFA from a Regular Expression

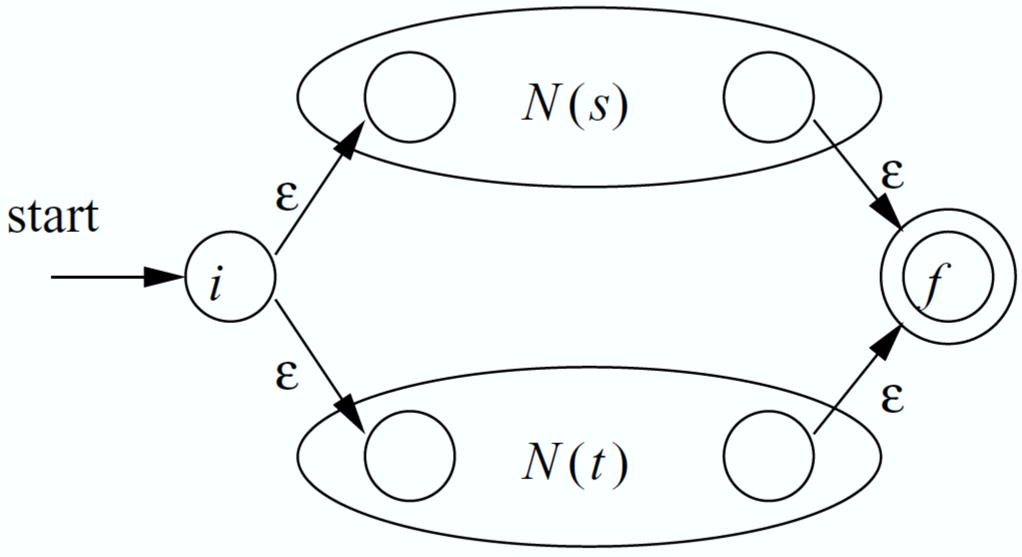
- **Basis**





Construction of an NFA from a Regular Expression

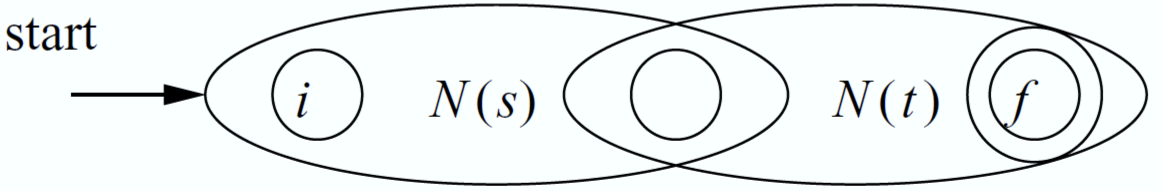
- **Induction**



NFA for the union of two regular expressions

Construction of an NFA from a Regular Expression

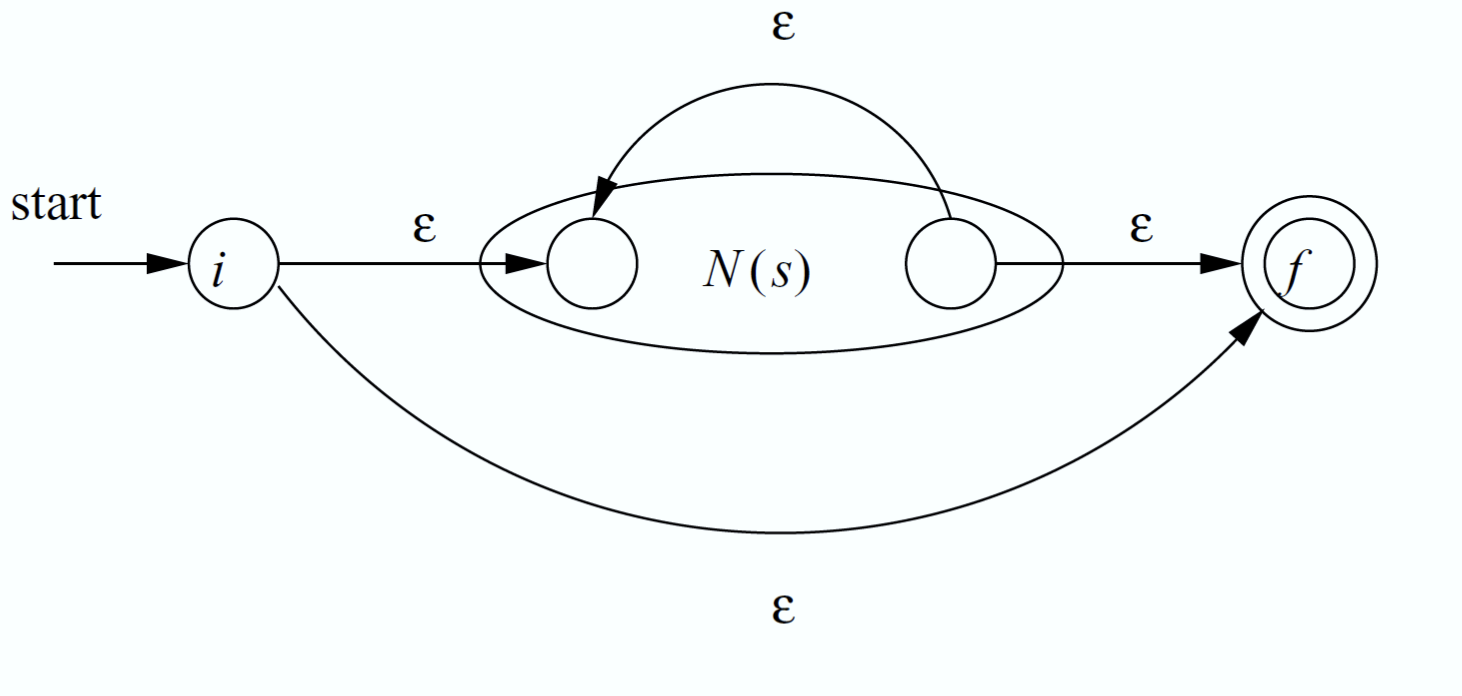
- **Induction**



NFA for the concatenation of two regular expressions

Construction of an NFA from a Regular Expression

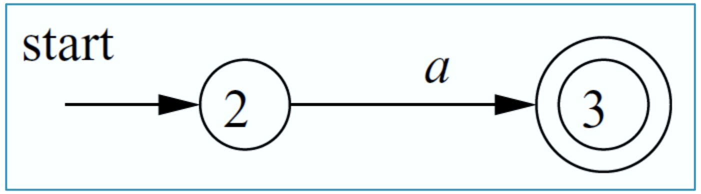
- **Induction**

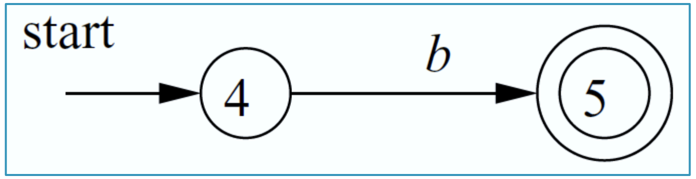


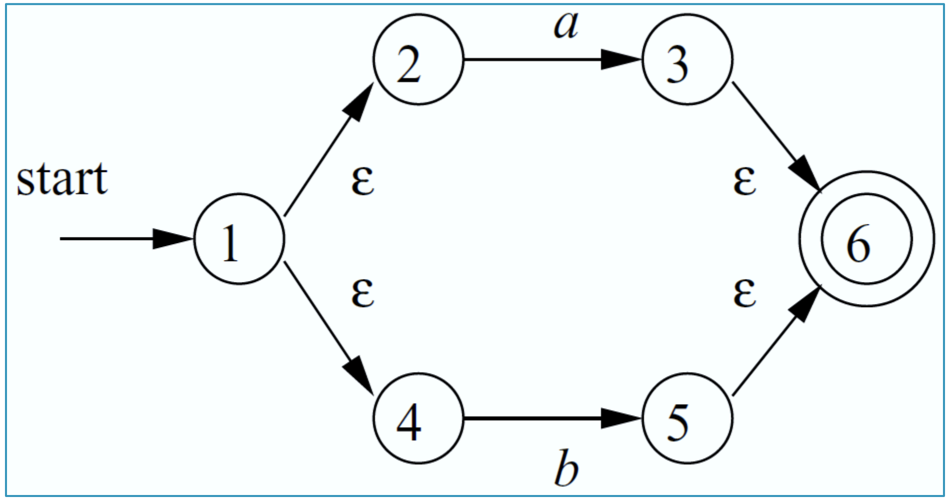
NFA for the closure of a regular expression

Construction of an NFA from a Regular Expression

- **Example:** Construct an NFA for $r = (a|b)^*abb$

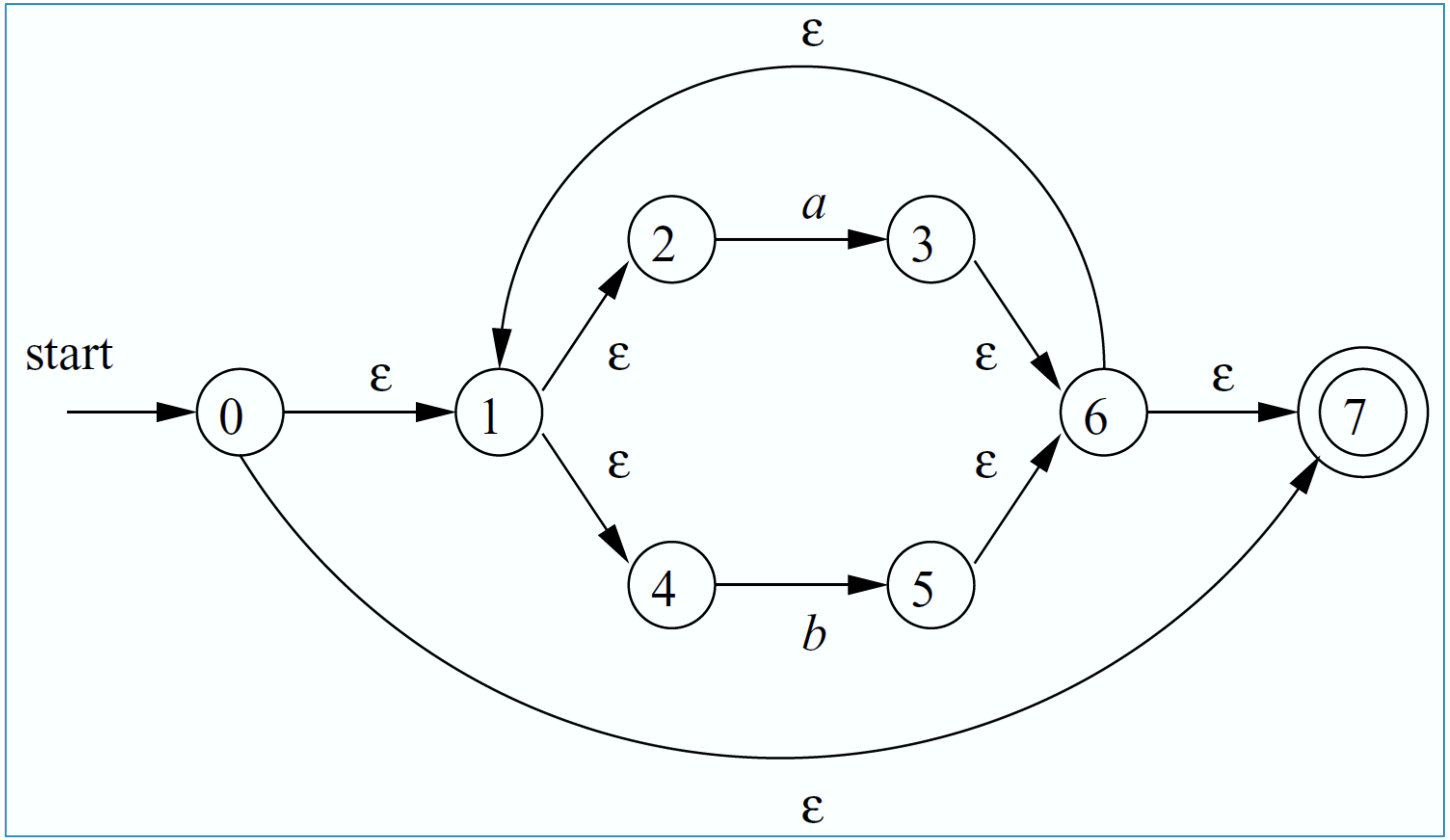






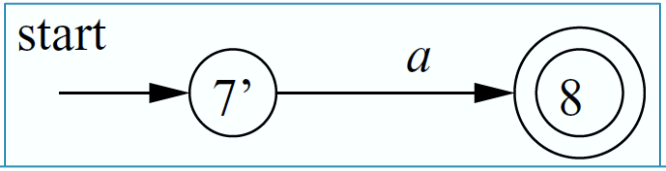
Construction of an NFA from a Regular Expression

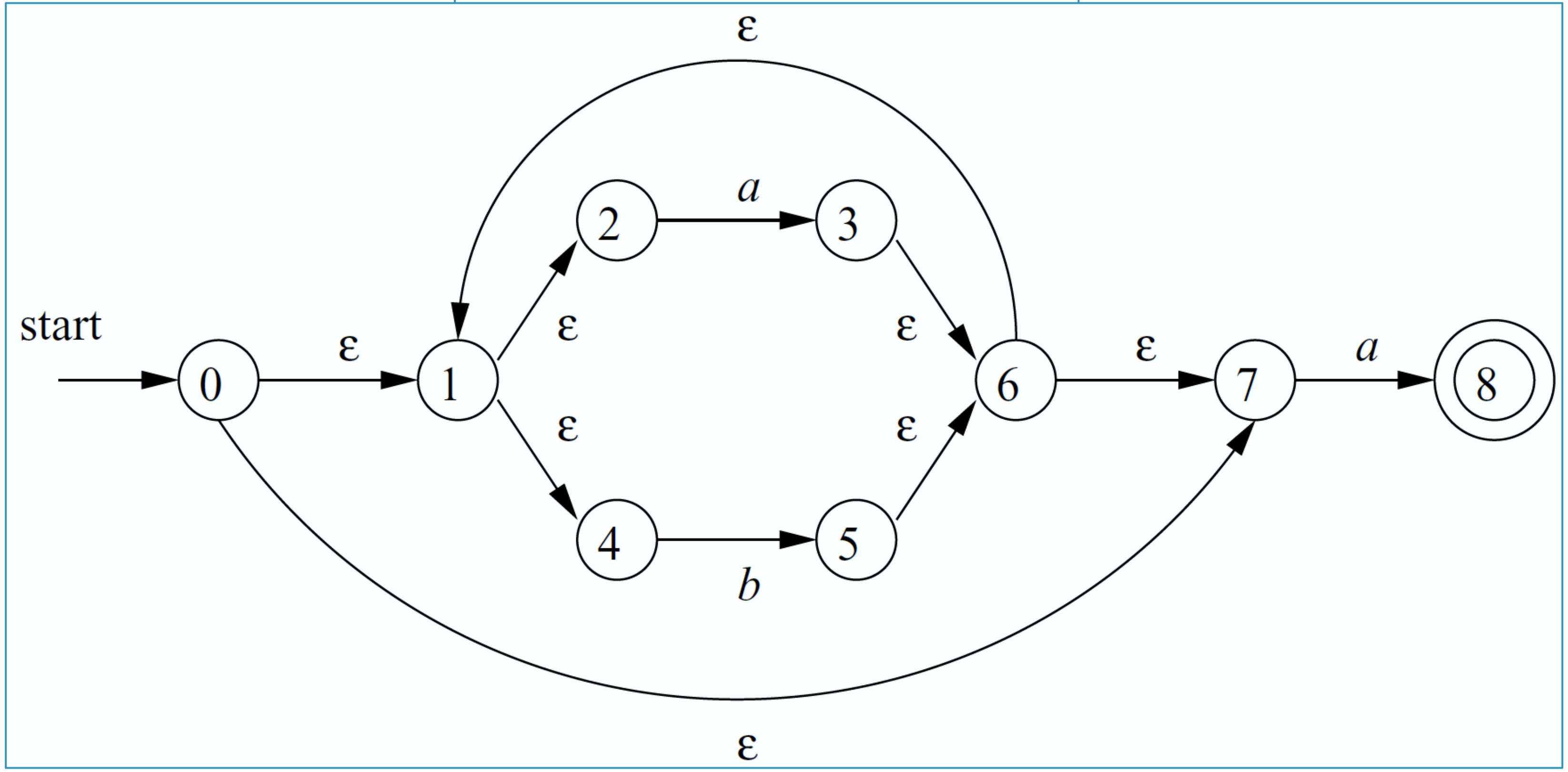
- **Example:** Construct an NFA for $r = (a|b)^*abb$



Construction of an NFA from a Regular Expression

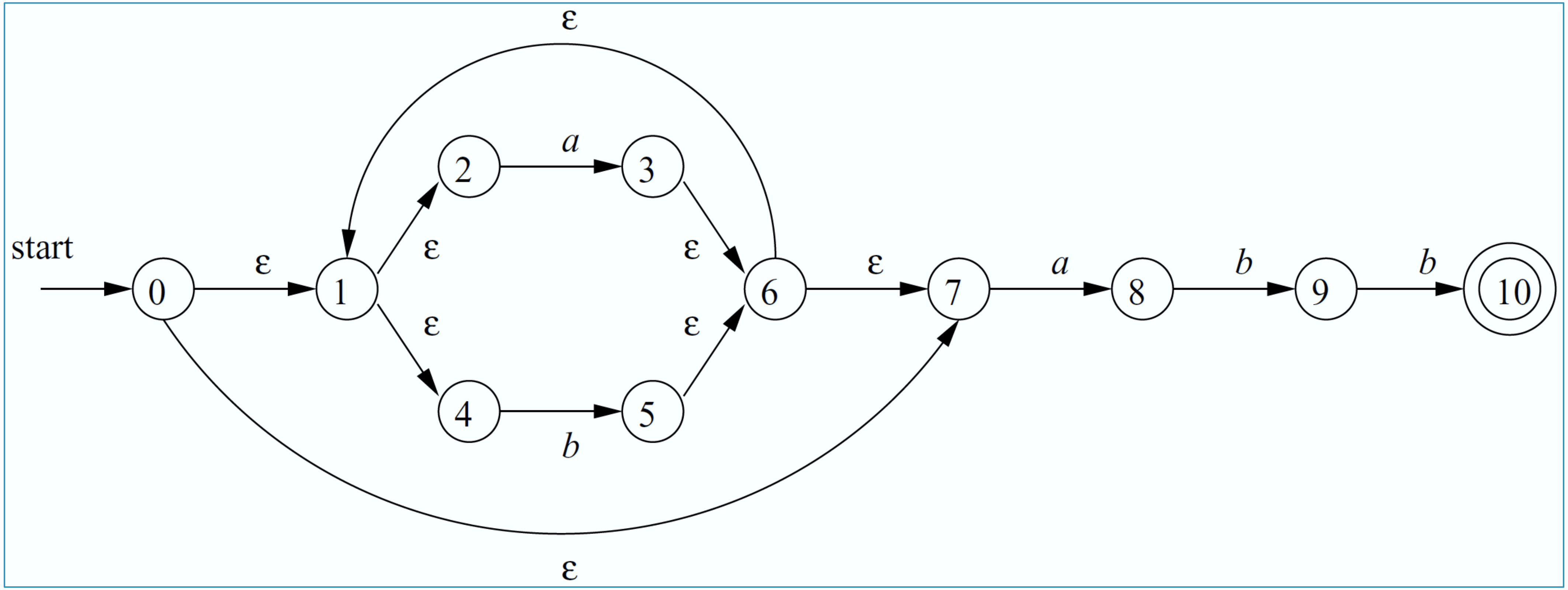
- **Example:** Construct an NFA for $r = (a|b)^*abb$





Construction of an NFA from a Regular Expression

- **Example:** Construct an NFA for $r = (a|b)^*abb$



Construction of an NFA from a Regular Expression

- **Example**

  a) $(\mathbf{a}|\mathbf{b})^*$.

  b) $(\mathbf{a}^*|\mathbf{b}^*)^*$.

  c) $\left((\epsilon|\mathbf{a})\mathbf{b}^*\right)^*$.

  d) $(\mathbf{a}|\mathbf{b})^*\mathbf{abb}(\mathbf{a}|\mathbf{b})^*$.

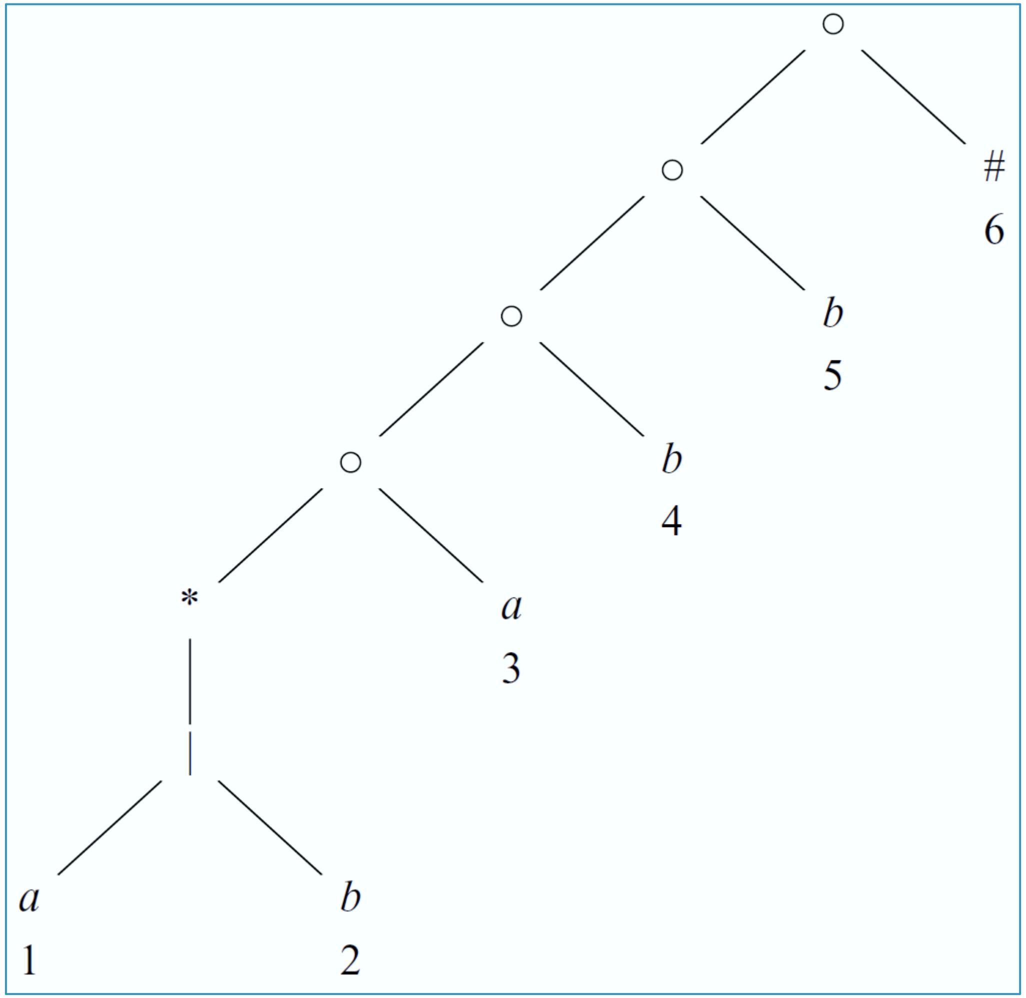
Converting a Regular Expression Directly to a DFA

- **Algorithm**
  1. Construct a syntax tree T from the augmented regular expression $(r)\#$
  2. Compute *nullable*, *firstpos*, *lastpos*, and *followpos* for T
  3. Construct *Dstates*, the set of states of DFA D, and *Dtran*, the transition function for D, using the following algorithm

```
initialize Dstates to contain only the unmarked state firstpos(n0),
        where n0 is the root of syntax tree T for (r)#;
while ( there is an unmarked state S in Dstates ) {
        mark S;
        for ( each input symbol a ) {
                let U be the union of followpos(p) for all p
                        in S that correspond to a;
                if ( U is not in Dstates )
                        add U as an unmarked state to Dstates;
                Dtran[S, a] = U;
        }
}
```

Converting a Regular Expression Directly to a DFA

- **Example:** Regular expression $(a|b)^*abb$
- **Syntax tree for $(a|b)^*abb\#$**
  - To each leaf not labeled ϵ, we attach a unique integer as the position of the leaf



Converting a Regular Expression Directly to a DFA

- **Functions Computed From the Syntax Tree**
  - ***nullable(n)*** is true for a syntax-tree node n if and only if the subexpression represented by n has ϵ in its language
  - ***firstpos(n)*** is the set of positions in the subtree rooted at n that correspond to the first symbol of at least one string in the language of the subexpression rooted at n
  - ***lastpos(n)*** is the set of positions in the subtree rooted at n that correspond to the last symbol of at least one string in the language of the subexpression rooted at n
  - ***followpos(p)***, for a position p, is the set of positions q in the entire syntax tree that can come after p

Converting a Regular Expression Directly to a DFA

NODE n	$nullable(n)$	$firstpos(n)$
A leaf labeled ϵ	**true**	$\emptyset$
A leaf with position i	**false**	$\{i\}$
An or-node $n = c_1 \mid c_2$	$nullable(c_1)$ **or** $nullable(c_2)$	$firstpos(c_1) \cup firstpos(c_2)$
A cat-node $n = c_1c_2$	$nullable(c_1)$ **and** $nullable(c_2)$	**if** ($nullable(c_1)$) $firstpos(c_1) \cup firstpos(c_2)$ **else** $firstpos(c_1)$
A star-node $n = {c_1}^*$	**true**	$firstpos(c_1)$

Converting a Regular Expression Directly to a DFA

- **Example**
  - ***firstpos*** and ***lastpos*** for nodes in the syntax tree for $(a|b)^*abb\#$