UNIVERSITY OF NICOSIA
ΠΑΝΕΠΙΣΤΗΜΙΟ ΛΕΥΚΩΣΙΑΣ

COMP-421 Compiler Design

Presented by
Dr Ioanna Dionysiou

Administrative

- [ASU07] Chapter 3 - Lexical Analysis

- Reading for next time…
  - [ASU07] Chapter 3

Lecture Outline

- Construction of NFA from a regular expression
  - Thompson's algorithm
- Conversion of NFA into DFA
  - Subset construction algorithm

Construction of NFA from r

- There are many techniques for building a recognizer from a regular expression
  - We will examine Thompson's construction

Thompson's Construction

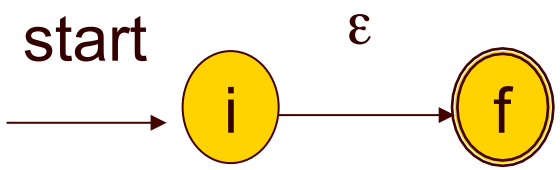
- Given a regular expression r over Σ we will produce an NFA accepting L(r)
- We will show how to construct automata
  - to recognize ε
  - to recognize any symbol in the alphabet
  - for expressions containing
    - Alternation
    - Concatenation
    - Kleene closure
- As the construction proceeds, each step introduces at most 2 new states

Thompson's Construction Steps

1. Parse (break down) r into its constituent subexpressions
2. Using rules (1),(2) construct NFAs for each of the basic symbols in r
   - If a symbol occurs several times in r, a separate NFA is constructed for each occurrence
3. Combine NFAs inductively using rule (3) until we obtain the NFA for the entire expression
   - Each intermediate NFA has exactly one final state, no edge enters the start state and no edge leaves the final state

Rule 1

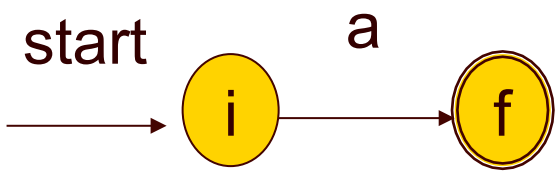
For ε, construct the NFA



i is the new initial state
f is the new final state

Rule 2

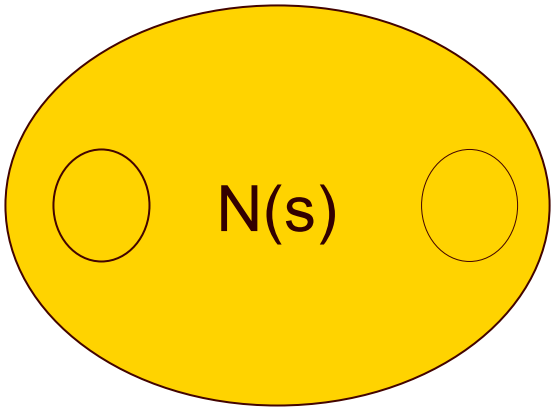
For symbol a in Σ, construct the NFA

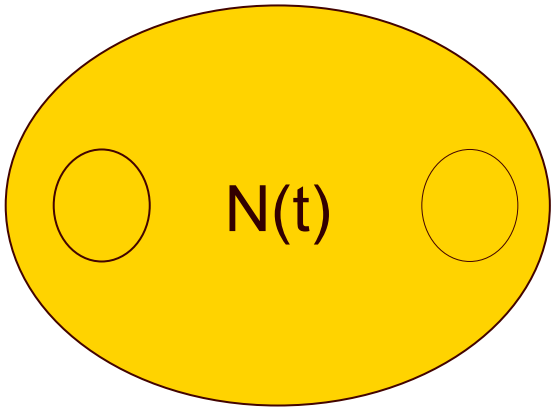


i is the new initial state
f is the new final state

Rule 3

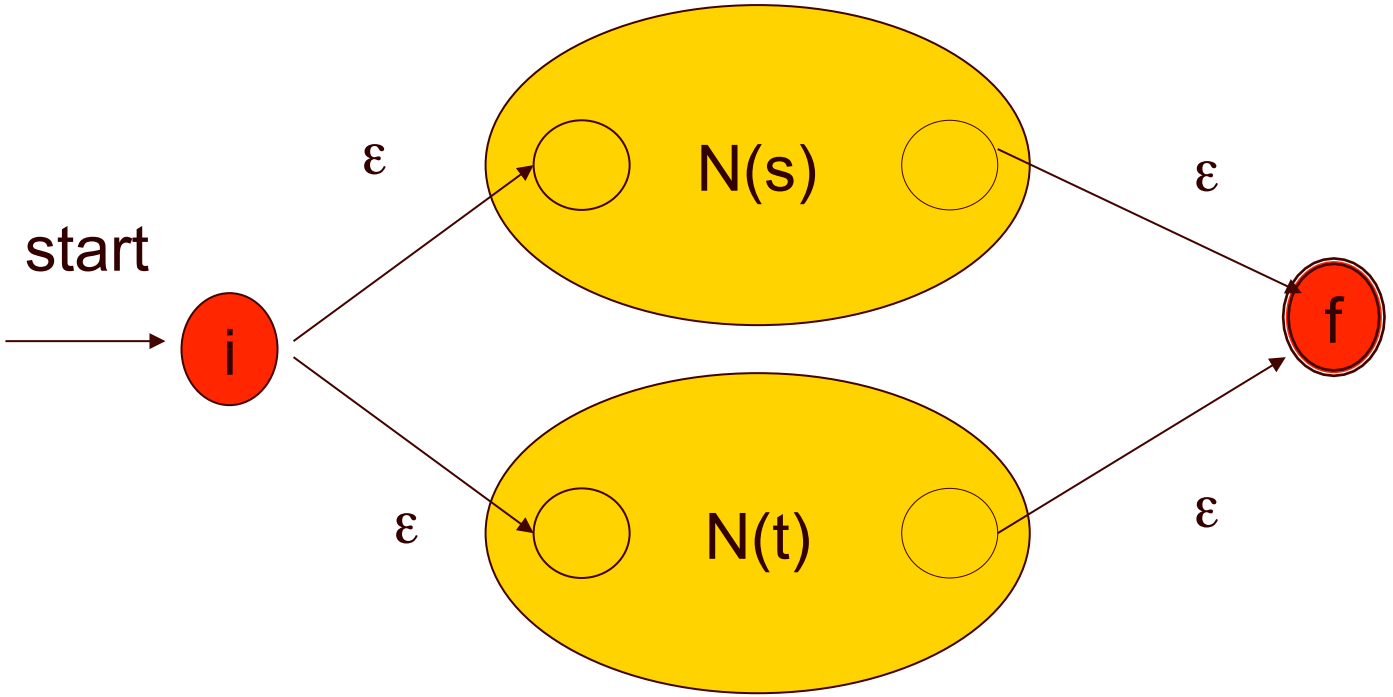
Suppose that N(s) and N(t) are NFA's for regular expressions s and t respectively





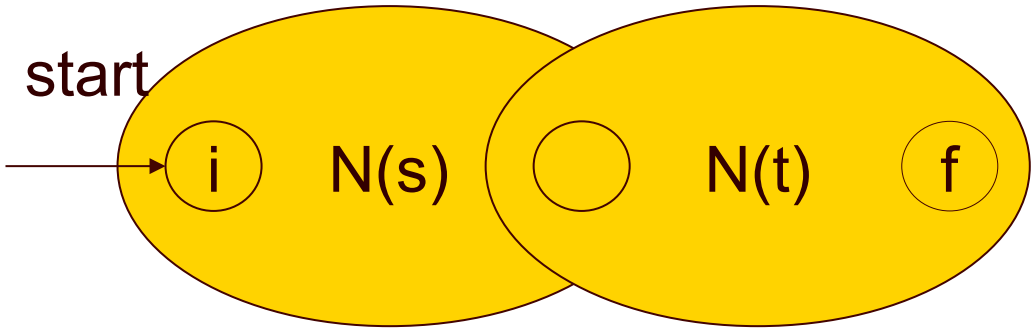
Rule 3 for N(s|t)

For regular expression s|t construct NFA N(s|t)



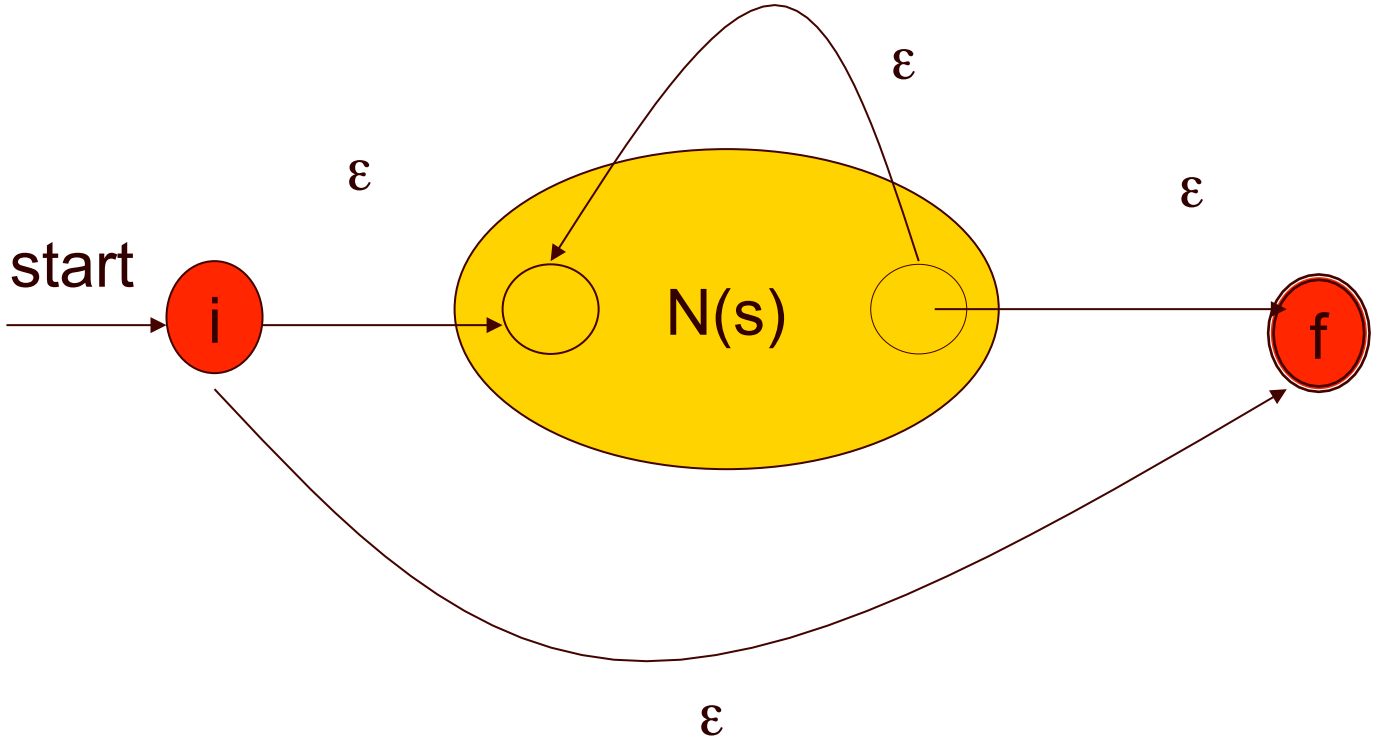
Rule 3 for N(st)

For regular expression st construct NFA N(st)



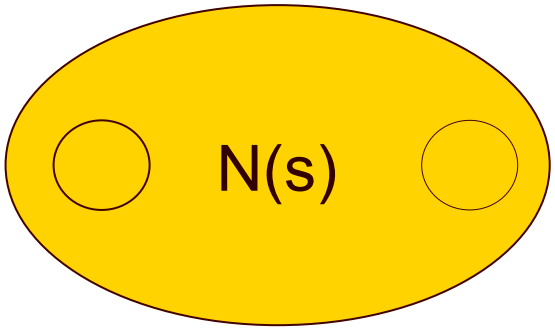
Rule 3 for N(s*)

For regular expression s* construct NFA N(s*)



Rule 3 for N((s))

For regular expression (s) construct NFA N((s)) = N(s)

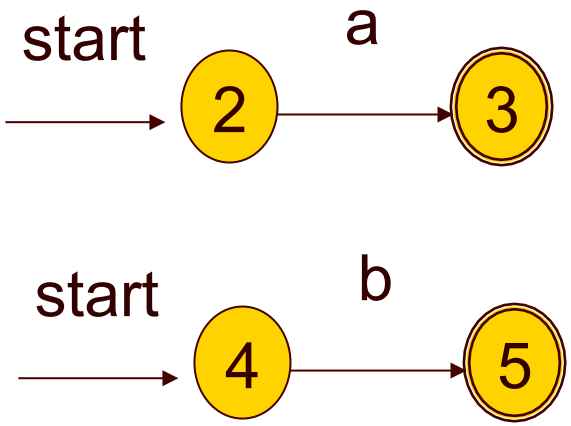


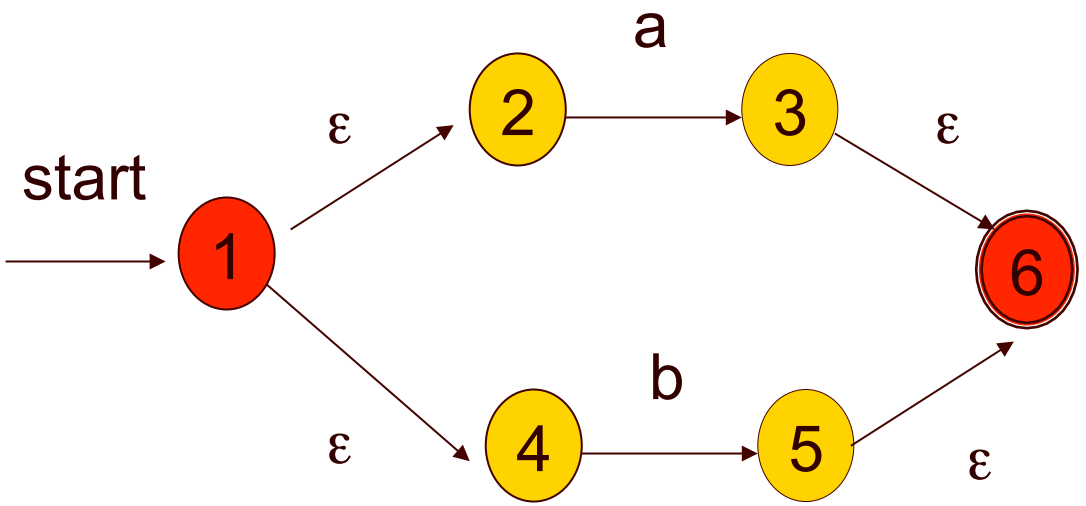
Exercise

- Construct N(r) for r = (a|b)*a
  - First break down the expression into smaller ones
    - r1 = a
    - r2 = b
    - r3 = a|b
    - r4 = (a|b)
    - r5 = (a|b)*
    - r6 = a
    - r7 = (a|b)*a
  - Next, construct all the intermediate NFAs using rules 1,2, and 3 to reach the final NFA.

N(r1), N(r2), N(r3)

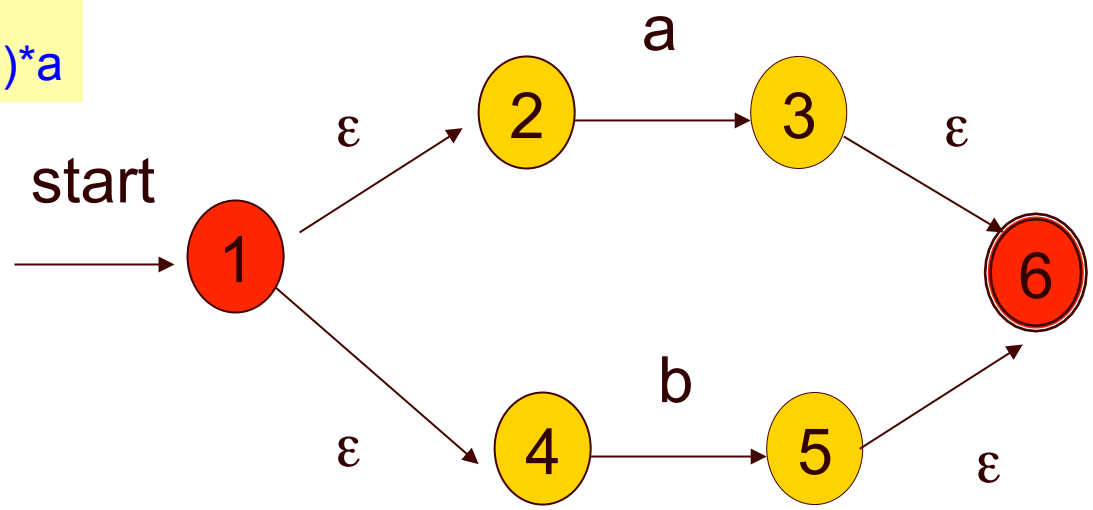
r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a





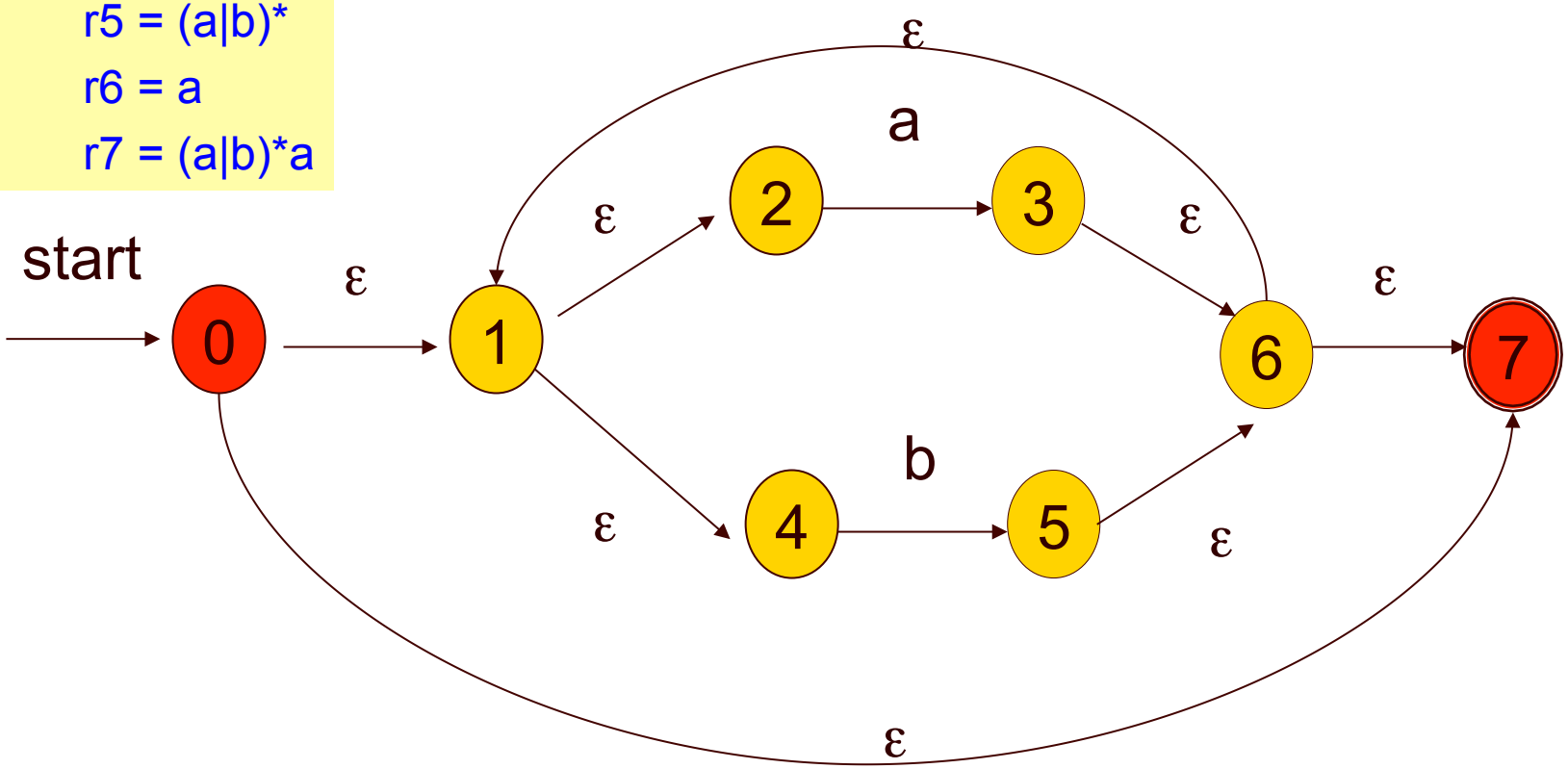
N(r4) = N((r3)) = N((a|b))

r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a



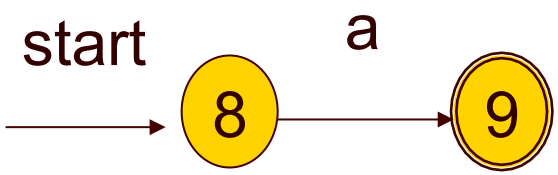
N(r5) = N((a|b)*)

r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a



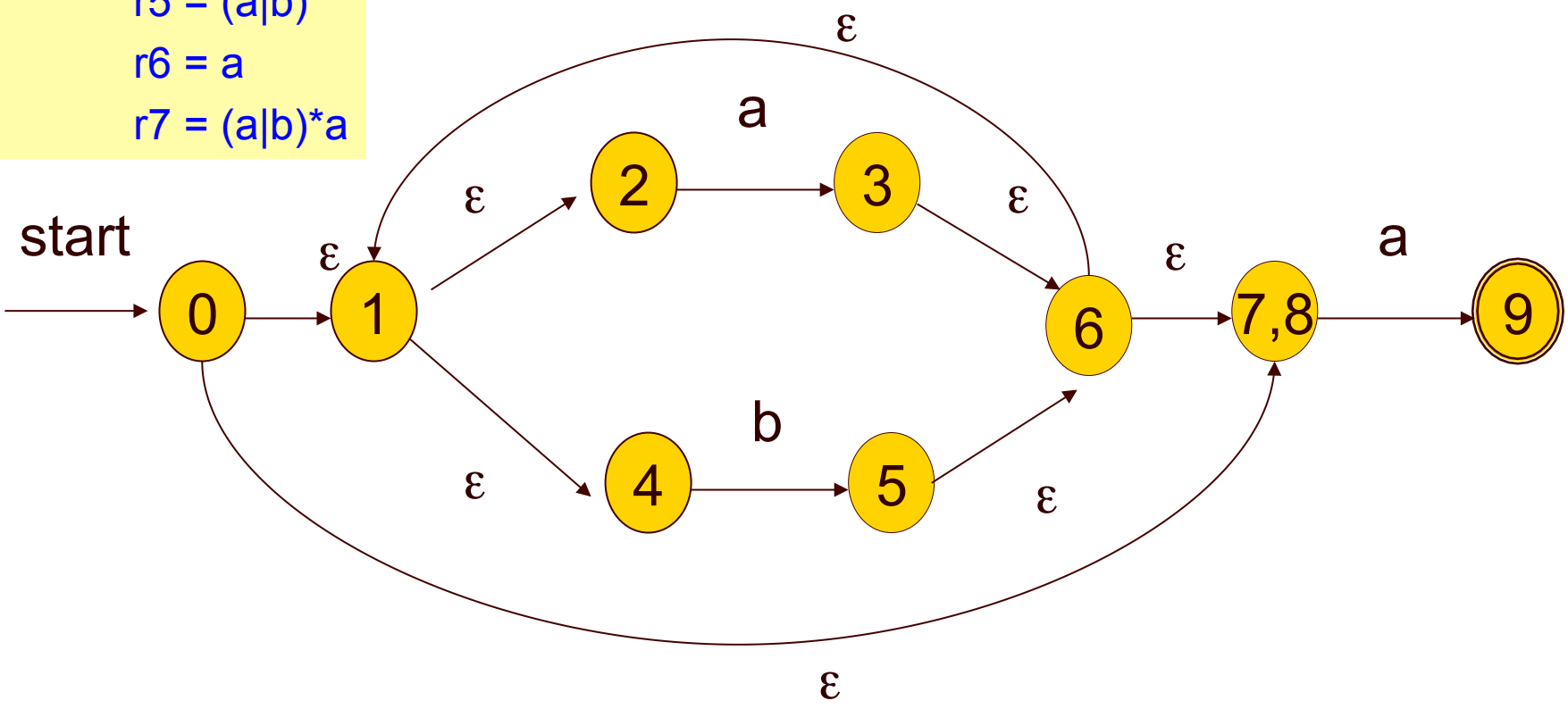
N(r6)

r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a



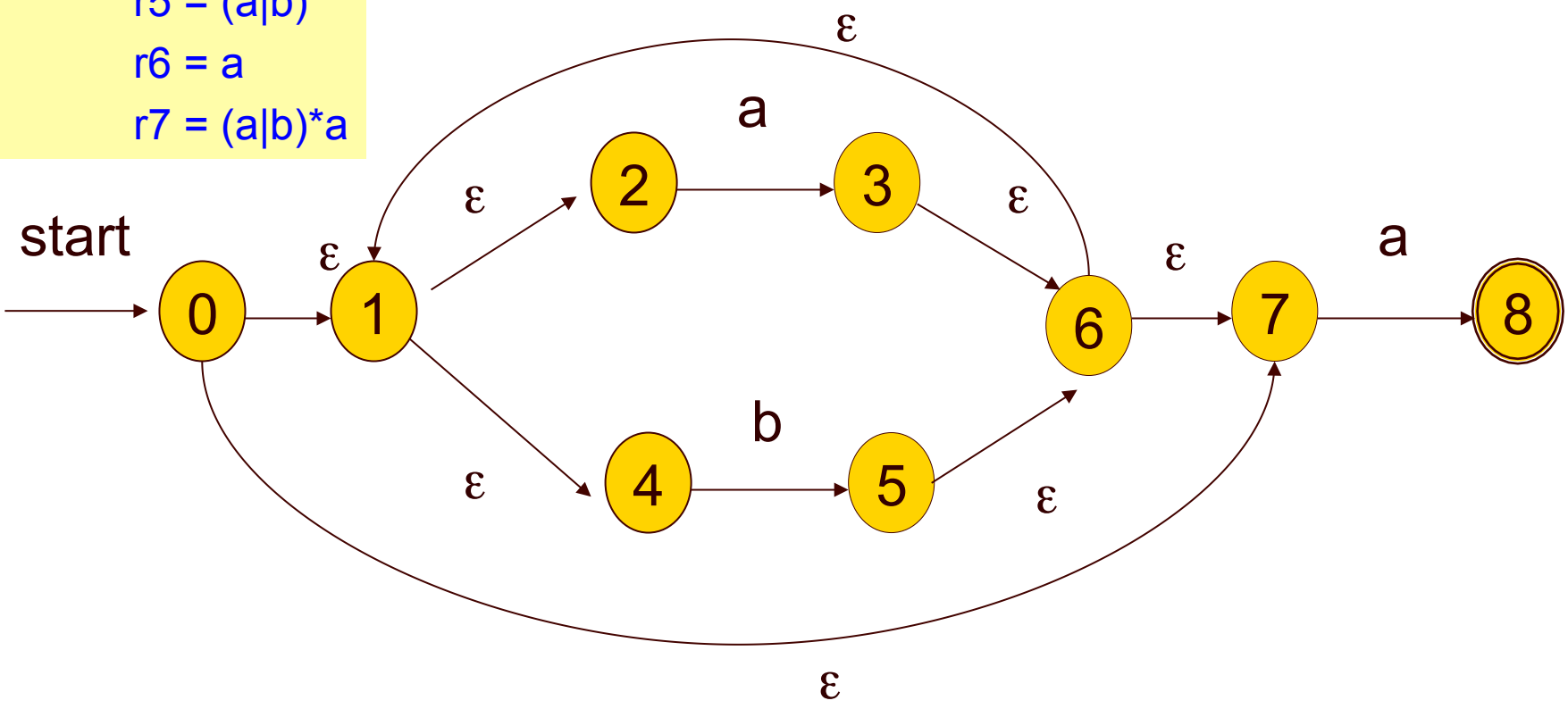
N(r7) = N(r5r6)

r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a



N(r7) = N(r5r6)

r1 = a
r2 = b
r3 = a|b
r4 = (a|b)
r5 = (a|b)*
r6 = a
r7 = (a|b)*a



In-class Exercise

- Using Thompson's construction algorithm construct NFA for
  - a(ab)*a
  - (a|b|c)a
  - 01*0

Lecture Outline

- Construction of NFA from a regular expression
  - Thompson's algorithm
- Conversion of NFA into DFA
  - Subset construction algorithm

Conversion of NFA into DFA

- NFA
  - More than one transition from a state on input α
  - Multivalued transition function makes it hard to simulate a NFA
    - If there are many paths that spell out the same input string, we may have to consider them all before we find one that leads to acceptance or rejection
- Subset construction algorithm
  - Constructs from a NFA a DFA that recognizes the same language
  - Similar ideas to construct LR parsers (chapter 4)

General Idea

- Each DFA state corresponds to a set of NFA states
  - DFA uses its state to keep track of all possible states the NFA can be in after reading each input symbol

Subset Construction Algorithm

Constructing a DFA from a NFA

Input: a NFA N

Output: a DFA D accepting the same language

Method: construct transition table Dtran so that D will simulate "in parallel" all possible moves N can make on a given input string

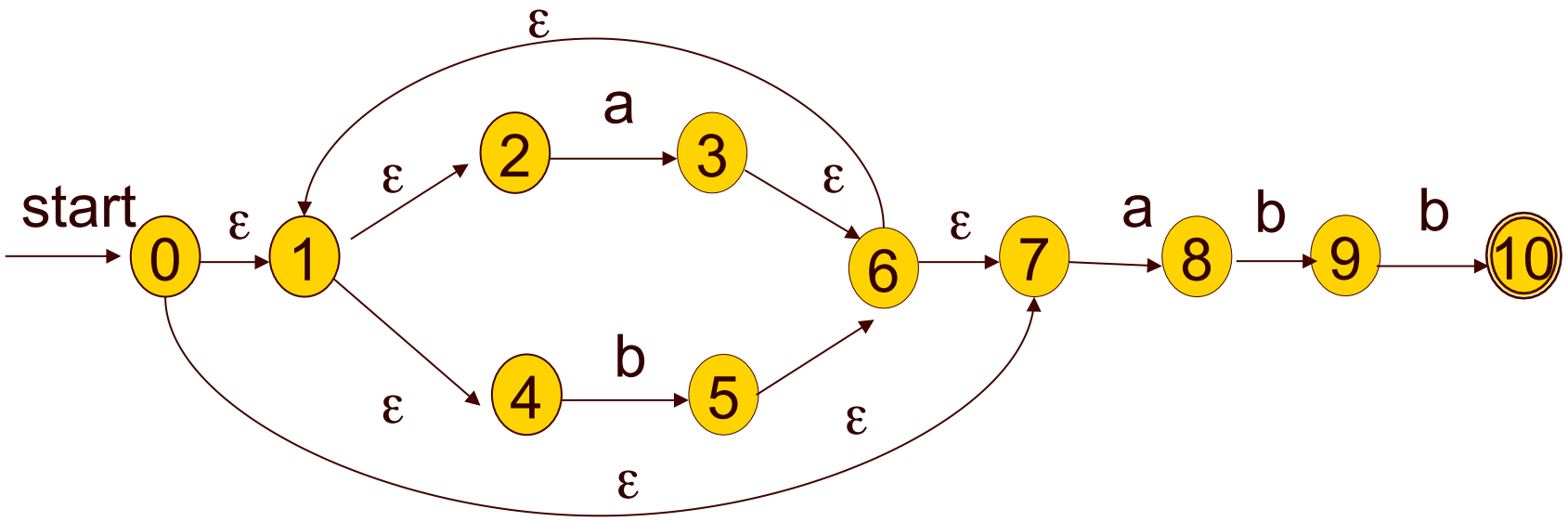
Operations on NFA states

OPERATION	DESCRIPTION
ε-closure(s)	Set of NFA states reachable from NFA state s on ε-transition alone
ε-closure(T)	Set of NFA states reachable from NFA state s in T on ε-transition alone
move(T,α)	Set of NFA states to which there is a transition on input symbol α from some NFA state s in T

s represents an NFA state
T represents a set of NFA states

Subset Construction 1

Before it sees any input symbol, N can be in any of the states in the set ε-closure(s_0), where s_0 is the start state of N



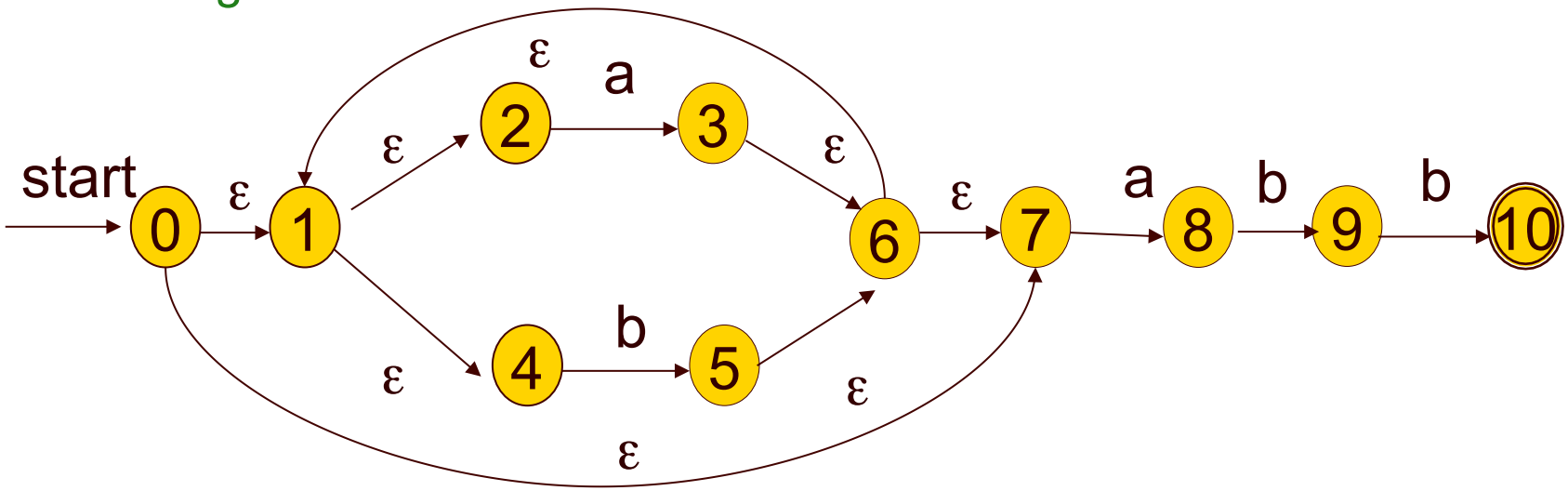
ε-closure(0) = {0,1,2,4,7} = T

Computation of ε-closure

```
push all states in T onto stack
initialize ε-closure(T) to T
while stack is not empty do
begin
        pop t, top element from stack
        for each state u with an edge from t to u labeled  ε do
        begin
                if u is not in ε-closure(T) then
                        add u to ε-closure(T)
                        push u onto stack
         end
end
```

Subset Construction 2

Suppose that exactly the states in set T are reachable from s_0 on a given sequence of input symbols, and let α be the next input symbol. On seeing α, N can move to any of the states in the set move(T, α). When we allow for transitions, N can be in any one the states in ε-closure(move(T, α)) after seeing α



ε -closure(move(T,a)) =

ε -closure(move({0,1,2,4,7},a)) =

ε -closure({3,8}) = {1,2,3,4,6,7,8}

Subset Construction 3

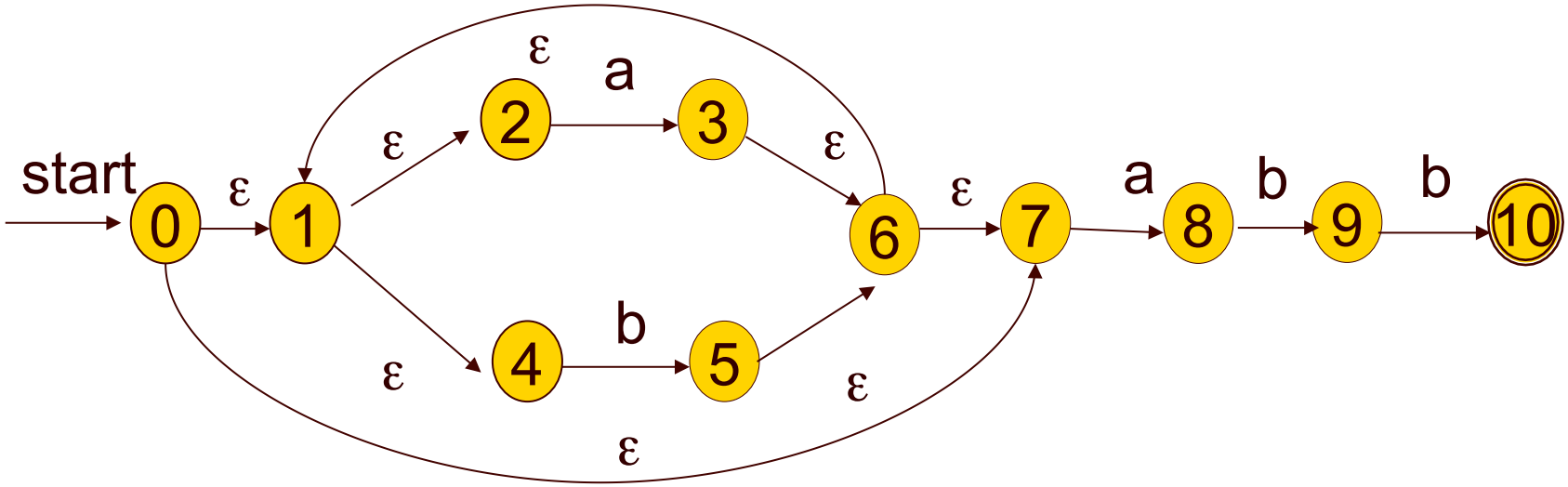
We construct Dstates (the set of states of D) and Dtran (the transition table of D) in the following manner:

1. Each state of D corresponds to a set of NFA states that N could be in after reading some sequence of input symbols, including all possible e transitions before or after symbols are read.
2. The start state of D is e-closure(s0)
3. States and transitions are added to D using the algorithm of next slide
4. A state of D is an accepting state if it is a set of NFA states containing at least one accepting state of N.

Subset Construction 4

Initially, ε-closure(s_0) is the only state in Dstates and it is unmarked
while there is unmarked state T in Dstates do begin
    mark T
    for each input symbol α do begin
        U := ε-closure(move(T, α))
        if U is not in Dstates then
            add U as an unmarked state to Dstates
        Dtran[T, α] := U
    end
end

Example



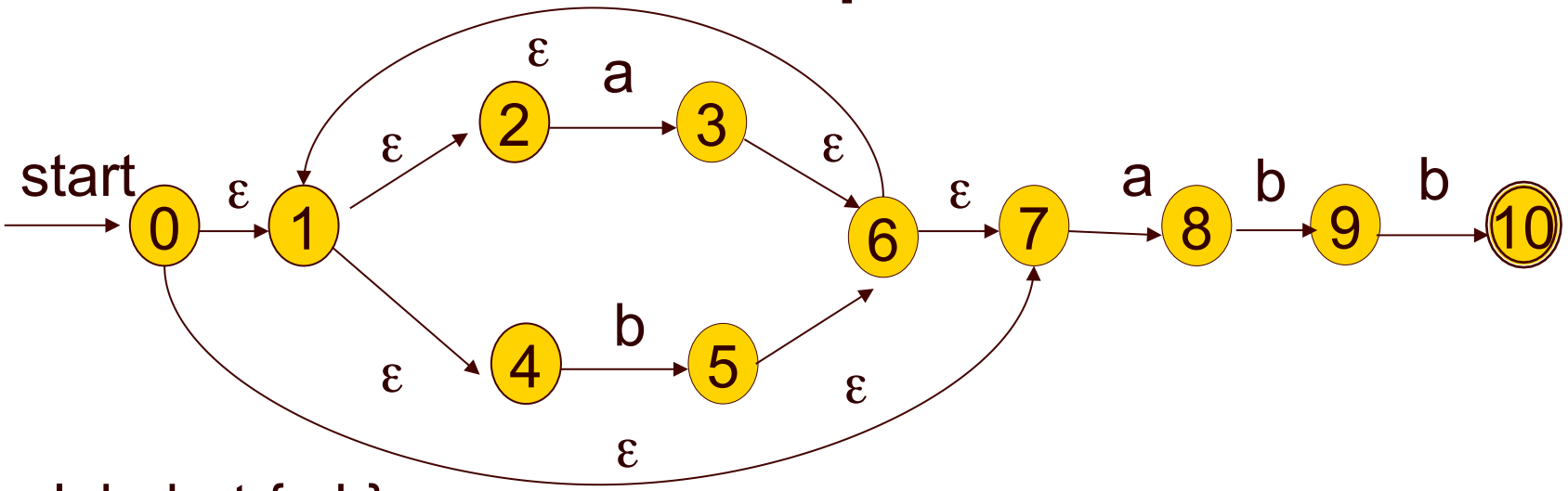
First, need to compute ε-closure(0)

ε-closure(0) = {0,1,2,4,7}

Let's call this A = {0,1,2,4,7}, and this is the start state of the equivalent DFA

Unmarked set = {A}

Example



Input alphabet {a,b}

Mark A and compute ε-closure(move(A,a))

move(A,a) = {3,8}

ε-closure(move(A,a)) =

ε-closure(move({0,1,2,4,7},a)) =

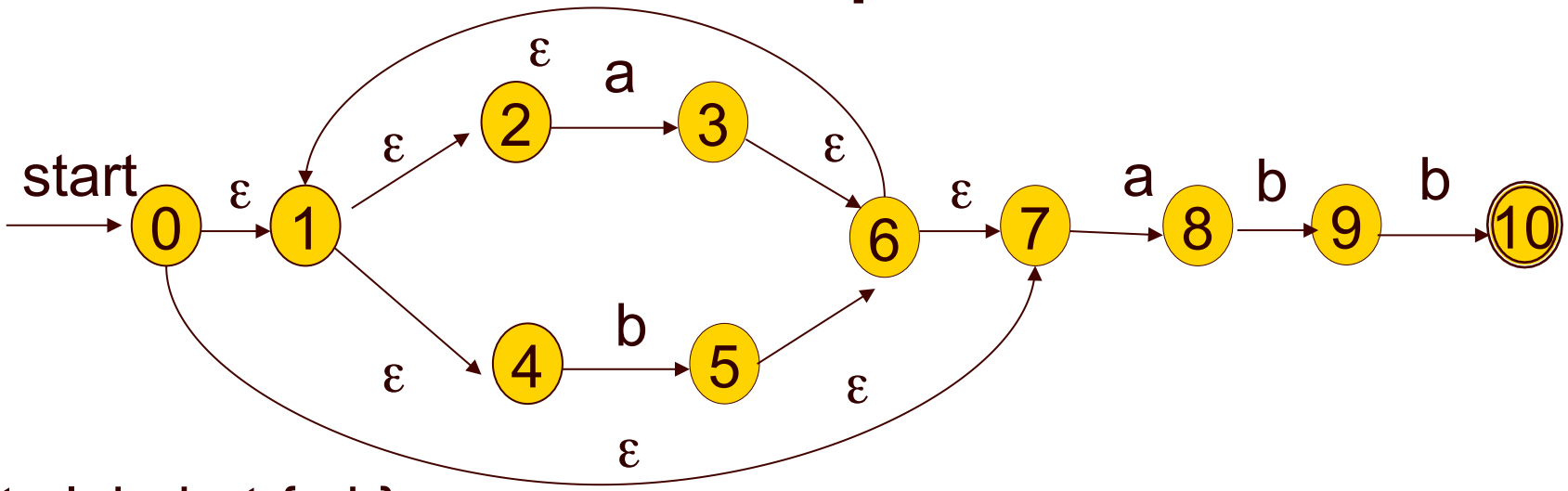
ε-closure({3,8}) = {1,2,3,4,6,7,8}

STATE	SYMBOL a	b
A	B	

Let's call this B = {1,2,3,4,6,7,8}, and this is another state of the equivalent DFA Dtran[A,a] = B

Unmarked set = {B}

Example



Input alphabet {a,b}

Mark A and compute ε-closure(move(A,b))

move(A,b) = {5}
ε-closure(move(A,b)) =
ε-closure(move({0,1,2,4,7},b)) =
ε-closure({5}) = {1,2,4,5,6,7}

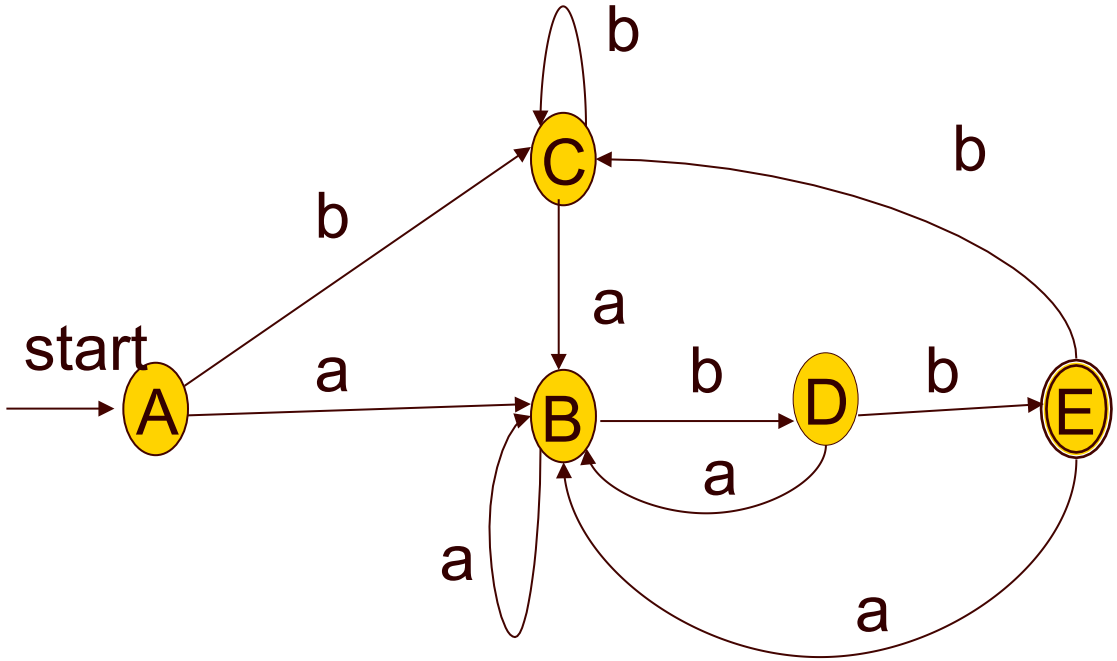
STATE	SYMBOL a	b
A	B	C

Let's call this C = {1,2,4,5,6,7}, and this is another state of the equivalent DFA Dtran[A,b] = C
Unmarked set = {B,C}

Continuing the example…

- The process is now repeated for all elements of the unmarked set
  - Unmarked set = {B, C}
  - These sets may produced new sets that are added to the unmarked set
- Eventually, all sets that are states of the DFA are marked
  - Unmarked set = { }
- For the example, here are 5 different sets of states that are constructed
  - A (start state), B, C, D, E (accepting state)

DFA and DTran



STATE	SYMBOL	
	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C