Introduction to Compiler Construction

Scanning: Generating a Scanner

Outline

Regular Expressions

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Singleton Rule: if $r = a \in \Sigma$, then $L(r)$ consists only of the string a

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Singleton Rule: if $r = a \in \Sigma$, then $L(r)$ consists only of the string a

Concatenation Rule: if r and s are regular expressions, then $L(rs)$ consists of strings obtained by concatenating a string from $L(r)$ to a string from $L(s)$

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Singleton Rule: if $r = a \in \Sigma$, then $L(r)$ consists only of the string a

Concatenation Rule: if r and s are regular expressions, then $L(rs)$ consists of strings obtained by concatenating a string from $L(r)$ to a string from $L(s)$

Alternation Rule: if r and s are regular expressions, then $L(r|s)$ consists of strings from $L(r)$ or $L(s)$

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Singleton Rule: if $r = a \in \Sigma$, then $L(r)$ consists only of the string a

Concatenation Rule: if r and s are regular expressions, then $L(rs)$ consists of strings obtained by concatenating a string from $L(r)$ to a string from $L(s)$

Alternation Rule: if r and s are regular expressions, then $L(r|s)$ consists of strings from $L(r)$ or $L(s)$

Kleene Closure Rule: if r is a regular expression, then $L(r*)$ consists of strings obtained by concatenating zero or more instances of strings from $L(r)$

Regular Expressions

Regular expressions can be used to formally specify the syntax of tokens in a language

A regular expression r describes a language $L(r)$ of strings over an alphabet Σ

Epsilon Rule: if $r = \epsilon$, then $L(r)$ consists only of the empty string

Singleton Rule: if $r = a \in \Sigma$, then $L(r)$ consists only of the string a

Concatenation Rule: if r and s are regular expressions, then $L(rs)$ consists of strings obtained by concatenating a string from $L(r)$ to a string from $L(s)$

Alternation Rule: if r and s are regular expressions, then $L(r|s)$ consists of strings from $L(r)$ or $L(s)$

Kleene Closure Rule: if r is a regular expression, then $L(r*)$ consists of strings obtained by concatenating zero or more instances of strings from $L(r)$

Grouping Rule: both r and (r) describe the same language, ie, $L(r) = L((r))$

Regular Expressions

Regular Expressions

Example: given an alphabet $\Sigma = \{a, b\}$

- $aa|ab|ba|bb$ is the language of all two-symbol strings over the alphabet
- $a(a|b)*$ is the language of all non-empty strings of a's and b's starting with an a
- $(a|b)*ab$ is the language of all strings of a's and b's ending in ab

Regular Expressions

Example: given an alphabet $\Sigma = \{a, b\}$

- $aa|ab|ba|bb$ is the language of all two-symbol strings over the alphabet
- $a(a|b)*$ is the language of all non-empty strings of a's and b's starting with an a
- $(a|b)*ab$ is the language of all strings of a's and b's ending in ab

Example (identifiers in *j--*): `( "a"..."z" | "A"..."Z" | "_" | "$" ) ( "a"..."z" | "A"..."Z" | "_" | "0"..."9" | "$" )*`

Regular Expressions

Example: given an alphabet $\Sigma = \{a, b\}$

- $aa|ab|ba|bb$ is the language of all two-symbol strings over the alphabet
- $a(a|b)*$ is the language of all non-empty strings of a's and b's starting with an a
- $(a|b)*ab$ is the language of all strings of a's and b's ending in ab

Example (identifiers in *j*--): `( "a"..."z" | "A"..."Z" | "_" | "$" ) ( "a"..."z" | "A"..."Z" | "_" | "0"..."9" | "$" )*`

Example (integer literals in *j*--): `( "0"..."9" ) ( "0"..."9" )*`

Finite State Automata

Finite State Automata

For any language described by a regular expression, there is a state diagram called Finite State Automaton (FSA) that can recognize the same language

Finite State Automata

For any language described by a regular expression, there is a state diagram called Finite State Automaton (FSA) that can recognize the same language

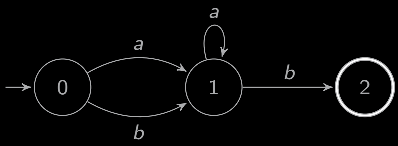
An FSA is a quintuple $A = (\Sigma, S, s_0, F, M)$, where

1. Σ is the input alphabet
2. S is a set of states
3. $s_0 \in S$ is a special start state
4. $F \subseteq S$ is a set of final states
5. M is a set of moves (aka transitions) of the form $m(r, a) = s$, where $r, s \in S$ and $a \in \Sigma$

Finite State Automata

Finite State Automata

Example (an FSA A that recognizes $L((a|b)a*b)$)



Formally, $A = (\Sigma, S, s_0, F, M)$, where $\Sigma = \{a, b\}$, $S = \{0, 1, 2\}$, $s_0 = 0$, $F = \{2\}$, and M is

r	a	$m(r, a)$
0	a	1
0	b	1
1	a	1
1	b	2

Non-deterministic Versus Deterministic Finite State Automata

Non-deterministic Versus Deterministic Finite State Automata

A non-deterministic finite state automaton (NFA) is one that allows

- An ϵ-move defined on the empty string ϵ, ie, $m(r, \epsilon) = s$
- More than one move from a state r on an input symbol a, ie, $m(r, a) = s$ and $m(r, a) = t$, where $s \neq t$

Non-deterministic Versus Deterministic Finite State Automata

A non-deterministic finite state automaton (NFA) is one that allows

- An ϵ-move defined on the empty string ϵ, ie, $m(r, \epsilon) = s$
- More than one move from a state r on an input symbol a, ie, $m(r, a) = s$ and $m(r, a) = t$, where $s \neq t$

An NFA is said to recognize an input string if, starting in the start state, there exists a set of moves based on the input that takes us into one of the final states

Non-deterministic Versus Deterministic Finite State Automata

Non-deterministic Versus Deterministic Finite State Automata

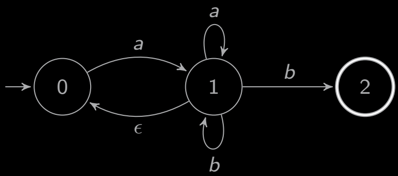
A deterministic finite state automaton (DFA) is one in which

- There are no ϵ-moves
- There is a unique move from any state r on an input symbol a, ie, if $m(r, a) = s$ and $m(r, a) = t$, then $s = t$
- There is a transition out of every state r on every input symbol a

Non-deterministic Versus Deterministic Finite State Automata

Non-deterministic Versus Deterministic Finite State Automata

Example: an NFA N that recognizes the language described by $a(a|b)*b$ over the alphabet $\{a, b\}$



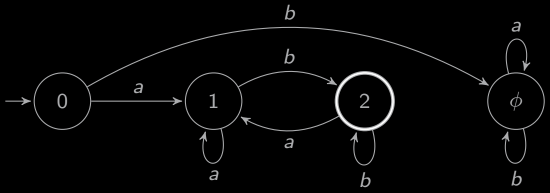
Formally, $N = (\Sigma, S, s_0, F, M)$ where $\Sigma = \{a, b\}$, $S = \{0, 1, 2\}$, $s_0 = 0$, $F = \{2\}$, and M is

r	a	$m(r, a)$
0	a	1
1	ϵ	0
1	a	1
1	b	1
1	b	2

Non-deterministic Versus Deterministic Finite State Automata

Non-deterministic Versus Deterministic Finite State Automata

Example: a DFA D that recognizes the language described by $a(a|b)*b$ over the alphabet $\{a, b\}$



Formally, $D = (\Sigma, S, s_0, F, M)$ where $\Sigma = \{a, b\}$, $S = \{0, 1, 2, \phi\}$, $s_0 = 0$, $F = \{2\}$, and M is

r	a	$m(r, a)$
0	a	1
0	b	ϕ
1	a	1
1	b	2
2	a	1
2	b	2
ϕ	a, b	ϕ

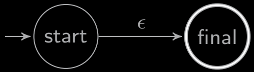
Regular Expressions to NFA

Regular Expressions to NFA

Given any regular expression r, we can construct an NFA N that recognizes the same language, ie, $L(N) = L(r)$

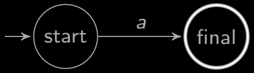
Regular Expressions to NFA · Thompson's Construction (Epsilon Rule)

NFA N_r for recognizing $L(r = \epsilon)$



Regular Expressions to NFA · Thompson's Construction (Singleton Rule)

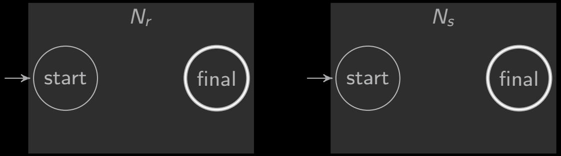
NFA N_r for recognizing $L(r = a)$



Regular Expressions to NFA · Thompson's Construction (Concatenation Rule)

Regular Expressions to NFA · Thompson's Construction (Concatenation Rule)

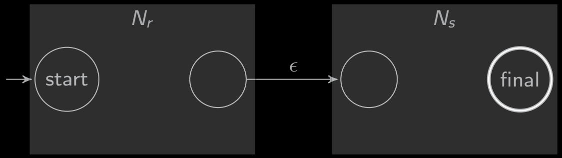
NFAs N_r and N_s for recognizing $L(r)$ and $L(s)$



Regular Expressions to NFA · Thompson's Construction (Concatenation Rule)

NFAs N_r and N_s for recognizing $L(r)$ and $L(s)$

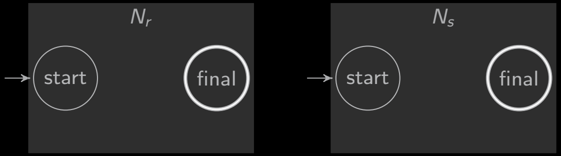
NFA N_{rs} for recognizing $L(rs)$



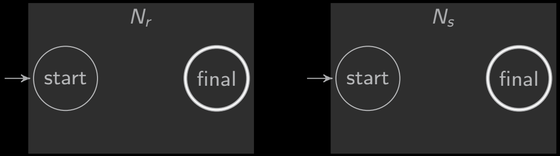
Regular Expressions to NFA · Thompson's Construction (Alternation Rule)

Regular Expressions to NFA · Thompson's Construction (Alternation Rule)

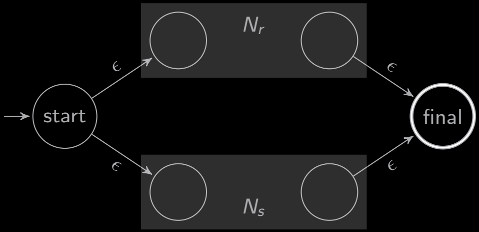
NFAs N_r and N_s for recognizing $L(r)$ and $L(s)$



NFAs N_r and N_s for recognizing $L(r)$ and $L(s)$



NFA $N_{r|s}$ for recognizing $L(r|s)$



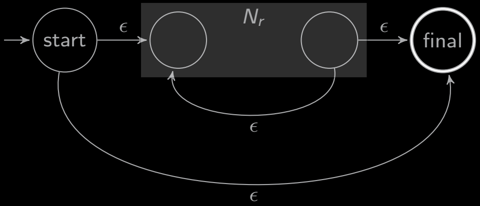
Regular Expressions to NFA · Thompson's Construction (Kleene Closure Rule)

NFAs N_r for recognizing $L(r)$

Regular Expressions to NFA · Thompson's Construction (Kleene Closure Rule)

NFAs N_r for recognizing $L(r)$

NFA N_{r*} for recognizing $L(r*)$



Regular Expressions to NFA · Thompson's Construction (Grouping Rule)

NFA N_r for recognizing $L(r)$ also recognizes $L((r))$

Regular Expressions to NFA

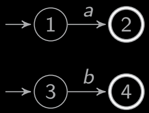
Regular Expressions to NFA

Example (NFA for $(a|b)a*b$)

Regular Expressions to NFA

Example (NFA for $(a|b)a*b$)

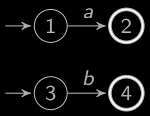
Using the singleton rule, we get the NFAs N_a and N_b for recognizing a and b as



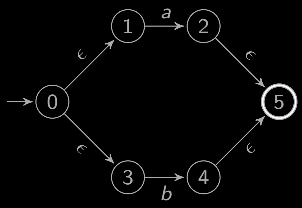
Regular Expressions to NFA

Example (NFA for $(a|b)a*b$)

Using the singleton rule, we get the NFAs N_a and N_b for recognizing a and b as



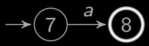
Using the alternation and grouping rules, we get the NFA $N_{(a|b)}$ for recognizing $(a|b)$ as



Regular Expressions to NFA

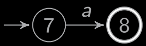
Regular Expressions to NFA

Using the singleton rule, we get the NFAs N_a for recognizing the second instance of a as

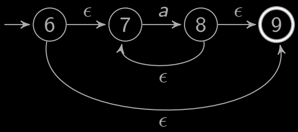


Regular Expressions to NFA

Using the singleton rule, we get the NFAs N_a for recognizing the second instance of a as



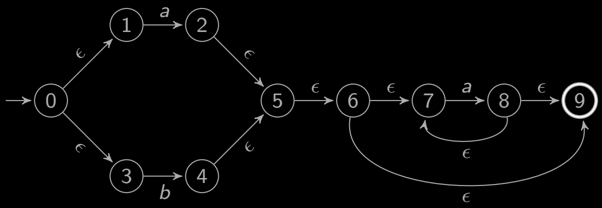
Using the Kleene closure rule, we get the NFA N_{a*} for recognizing $a*$ as



Regular Expressions to NFA

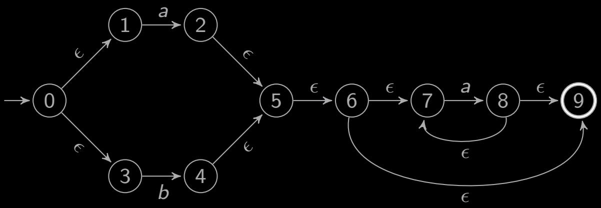
Regular Expressions to NFA

Using the concatenation rule, we get the NFA $N_{(a|b)a*}$ for recognizing $(a|b)a*$

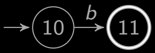


Regular Expressions to NFA

Using the concatenation rule, we get the NFA $N_{(a|b)a*}$ for recognizing $(a|b)a*$



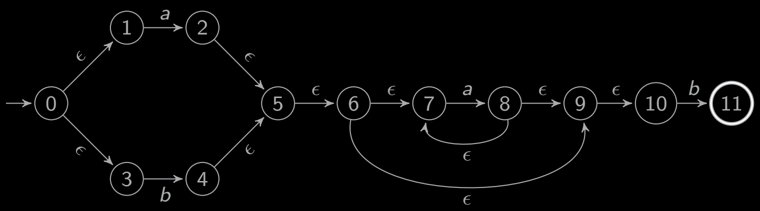
Using the singleton rule, we get the NFAs N_b for recognizing the second instance of b as



Regular Expressions to NFA

Regular Expressions to NFA

Finally, using the concatenation rule, we get the NFA $N_{(a|b)a*b}$ for recognizing $(a|b)a*b$ as



NFA to DFA

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

The DFA is always in a state that simulates all the possible states that the NFA could possibly be in having scanned the same portion of the input

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

The DFA is always in a state that simulates all the possible states that the NFA could possibly be in having scanned the same portion of the input

ϵ-closure(s) computes all states reachable from a given state s using ϵ-moves alone

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

The DFA is always in a state that simulates all the possible states that the NFA could possibly be in having scanned the same portion of the input

ϵ-closure(s) computes all states reachable from a given state s using ϵ-moves alone

$\epsilon\text{-closure}(s) = \{s\} \cup \{r \in S |$ there is a path of only ϵ-moves from s to $r\}$

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

The DFA is always in a state that simulates all the possible states that the NFA could possibly be in having scanned the same portion of the input

ϵ-closure(s) computes all states reachable from a given state s using ϵ-moves alone

$\epsilon\text{-closure}(s) = \{s\} \cup \{r \in S |$ there is a path of only ϵ-moves from s to $r\}$

ϵ-closure(S) computes all states reachable from any state $s \in S$ using ϵ-moves alone

NFA to DFA

For any NFA N, we can construct an equivalent DFA D such that $L(D) = L(N)$

The DFA is always in a state that simulates all the possible states that the NFA could possibly be in having scanned the same portion of the input

ϵ-closure(s) computes all states reachable from a given state s using ϵ-moves alone

ϵ-closure(s) $= \{s\} \cup \{r \in S |$ there is a path of only ϵ-moves from s to $r\}$

ϵ-closure(S) computes all states reachable from any state $s \in S$ using ϵ-moves alone

$$\epsilon\text{-closure}(S) = \bigcup_{s \in S} \epsilon\text{-closure}(s)$$

NFA to DFA · ϵ-closure(S)

Input: a set of states S
Output: ϵ-closure(S)

$P \leftarrow$ Stack(S)
$C \leftarrow$ Set(S)
while not P.isEmpty() **do**
    $r \leftarrow P$.pop()
    for $s \in m(r, \epsilon)$ **do**
        if $s \notin C$ **then**
            P.push(s)
            C.add(s)
        end if
    end for
end while
return C

NFA to DFA · ϵ-closure(s)

NFA to DFA · ϵ-closure(s)

Input: a state s
Output: ϵ-closure(s)

$S \leftarrow \mathrm{Set}(s)$
return ϵ-closure(S)

NFA to DFA · Subset Construction

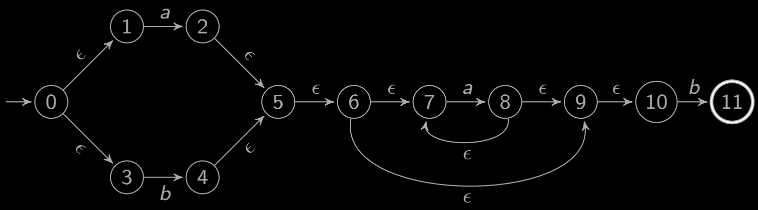
Input: an NFA $N = (\Sigma, S, s_0, M, F)$
Output: an equivalent DFA $D = (\Sigma, S_D, s_{D0}, M_D, F_D)$

$s_{D0} \leftarrow \epsilon\text{-closure}(s_0)$
$S_D \leftarrow \text{Set}(s_{D0})$
$M_D \leftarrow Moves()$
$stk \leftarrow \text{Stack}(s_{D0})$
$i \leftarrow 1$
while not $stk.\text{isEmpty}()$ **do**
$r \leftarrow stk.\text{pop}()$
for $a \in \Sigma$ **do**
$s_{Di} \leftarrow \epsilon\text{-closure}(\cup_{r' \in r} m(r', a))$
if $s_{Di} \neq \{\}$ **then**
if $s_{Di} \notin S_D$ **then**
$S_D.\text{add}(s_{Di})$
$stk.\text{push}(s_{Di})$
$i \leftarrow i + 1$
$M_D.\text{add}((r, a) \rightarrow s_{Di})$
else
$M_D.\text{add}((r, a) \rightarrow s_j)$, where $s_j \in S_D$ such that $s_j = s_{Di}$
end if
else
$S_D.\text{add}(\phi)$; $M_D.\text{add}((r, a) \rightarrow \phi)$; and $M_D.\text{add}((\phi, a) \rightarrow \phi)$
end if
end for
end while
$F_D \leftarrow \text{Set}()$
for $s_D \in S_D$ **do**
for $s \in s_D$ **do**
if $s \in F$ **then**
$F_D.\text{add}(s_D)$
end if
end for
end for
return $D = (\Sigma, S_D, s_{D0}, M_D, F_D)$

NFA to DFA

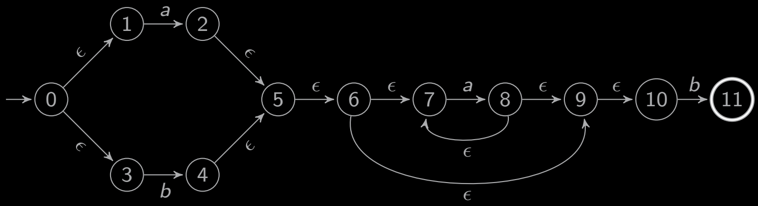
NFA to DFA

Example (DFA for $N_{(a|b)a*b}$)



NFA to DFA

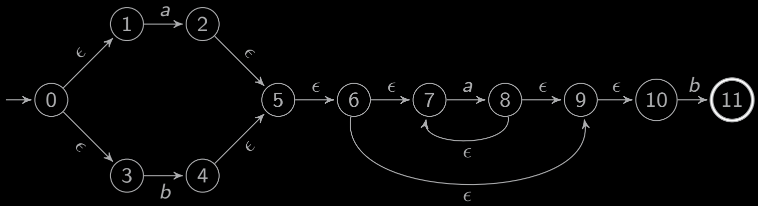
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$

NFA to DFA

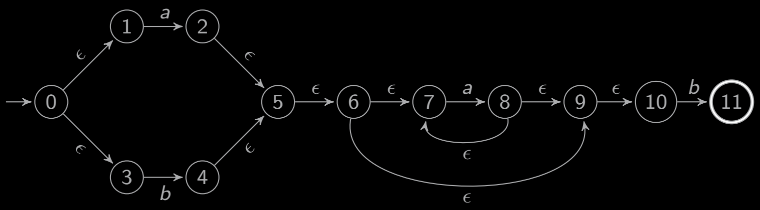
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)		

NFA to DFA

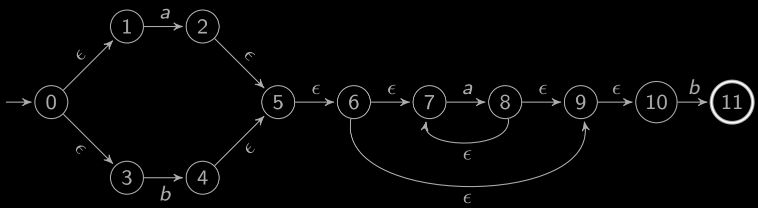
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	

NFA to DFA

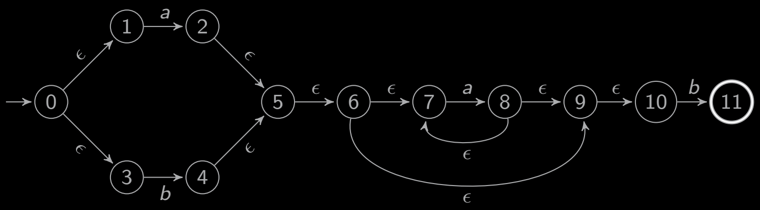
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$

NFA to DFA

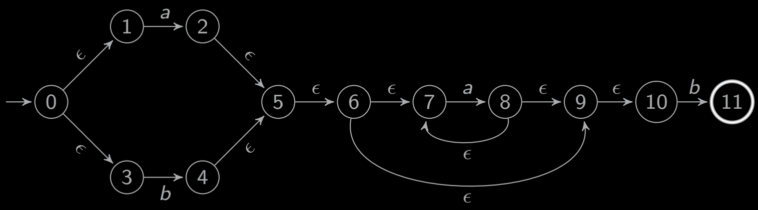
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0		

NFA to DFA

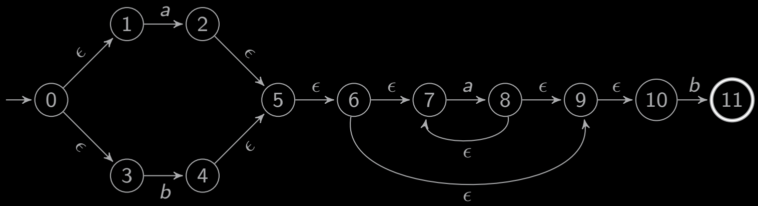
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	

NFA to DFA

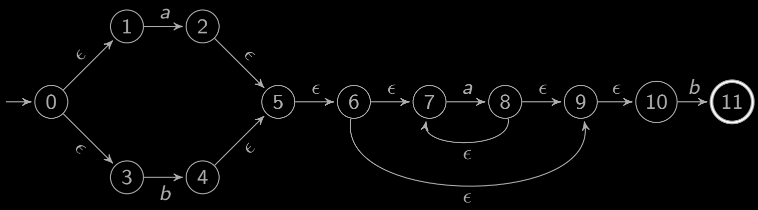
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$

NFA to DFA

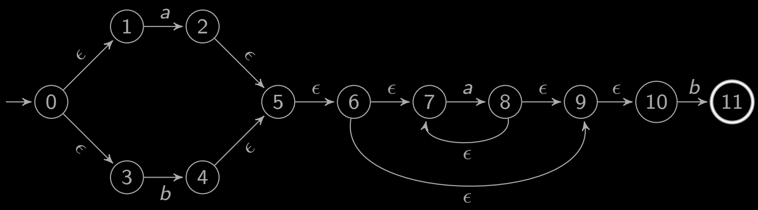
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1		

NFA to DFA

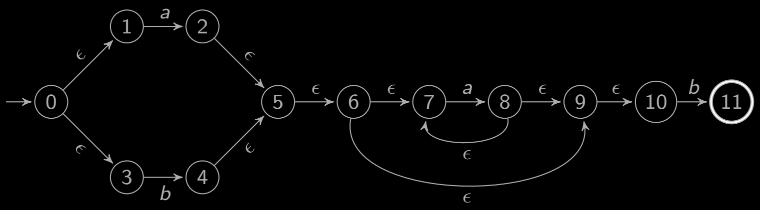
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	

NFA to DFA

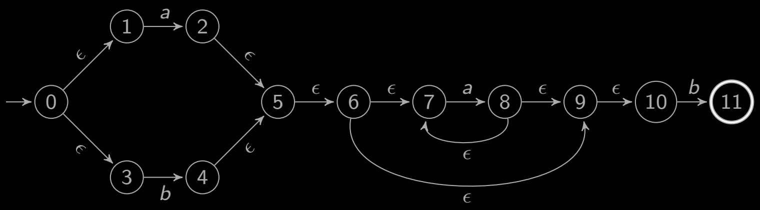
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$

NFA to DFA

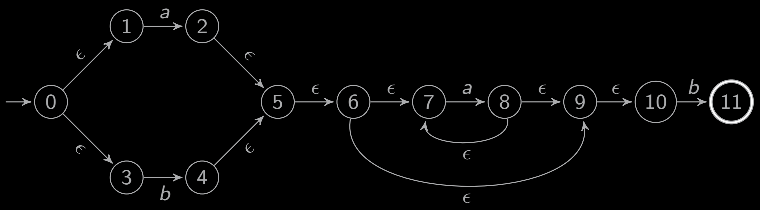
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1		

NFA to DFA

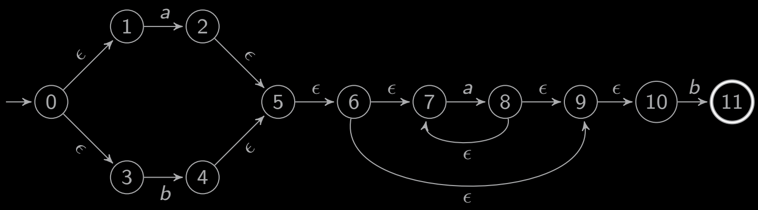
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	

NFA to DFA

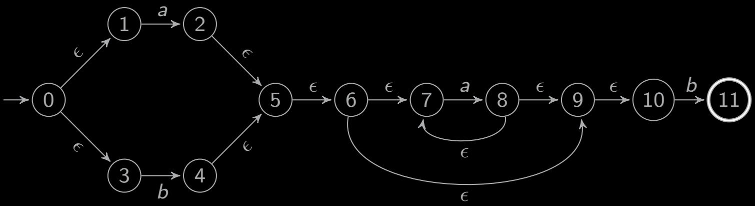
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r,a)$
$\{0,1,3\} = 0$ (start state)	a	$\{2,5,6,7,9,10\} = 1$
0	b	$\{4,5,6,7,9,10\} = 2$
1	a	$\{7,8,9,10\} = 3$
1	b	$\{11\} = 4$ (final state)

NFA to DFA

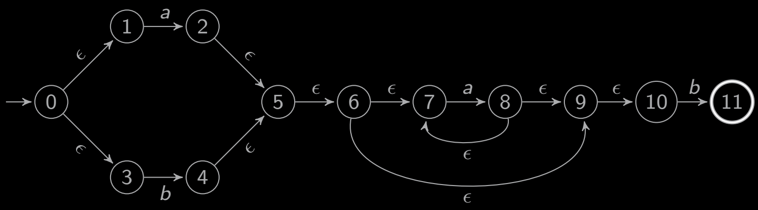
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2		

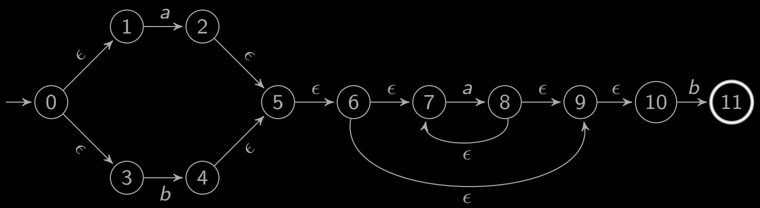
NFA to DFA

Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	

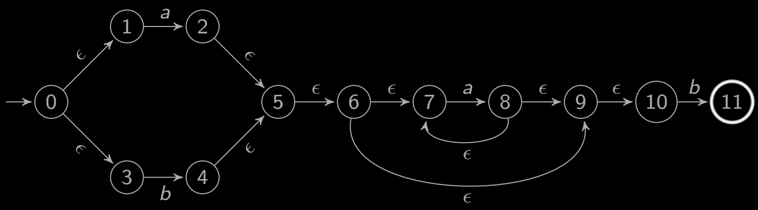
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3

NFA to DFA

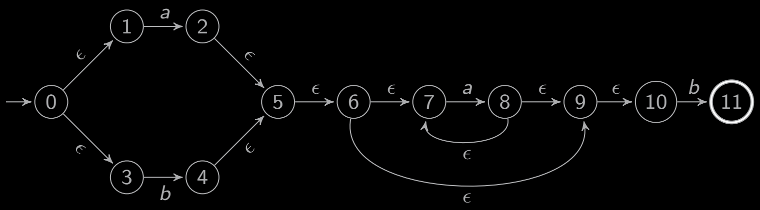
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2		

NFA to DFA

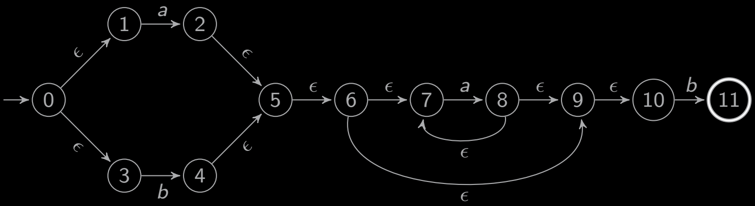
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	

NFA to DFA

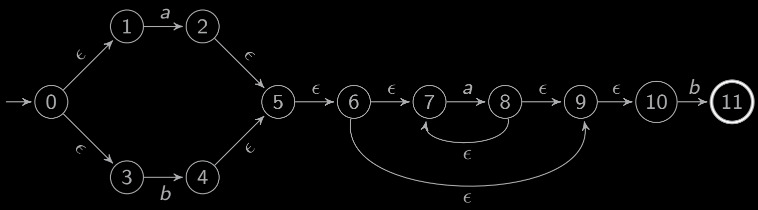
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4

NFA to DFA

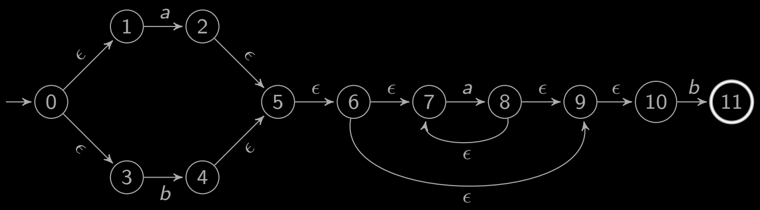
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3		

NFA to DFA

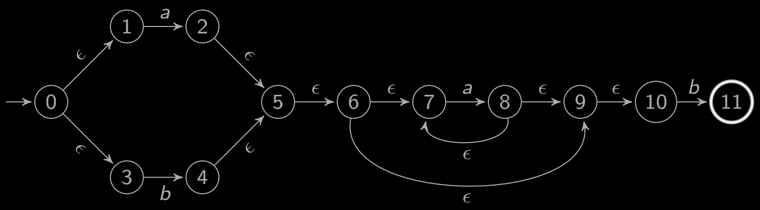
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	

NFA to DFA

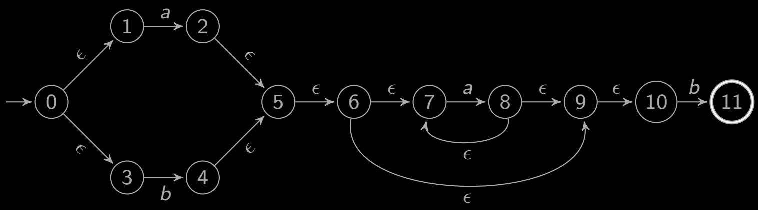
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3

NFA to DFA

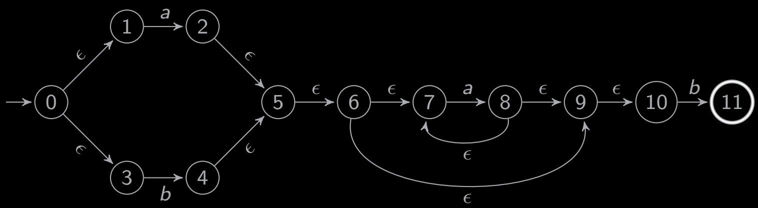
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3		

NFA to DFA

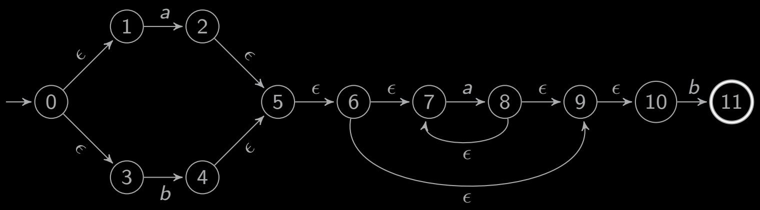
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	

NFA to DFA

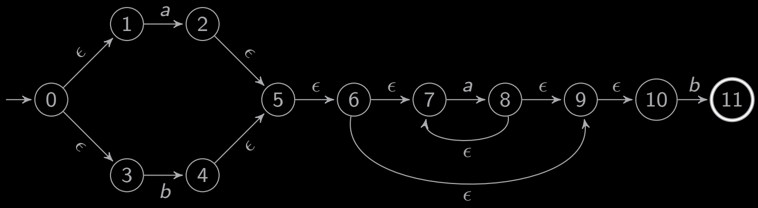
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4

NFA to DFA

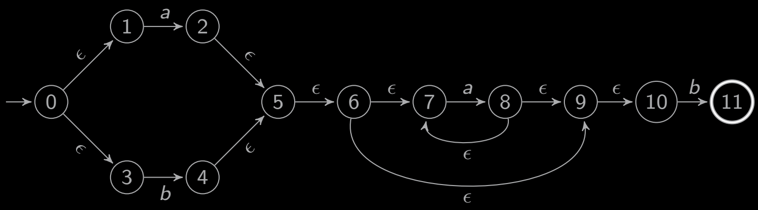
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4		

NFA to DFA

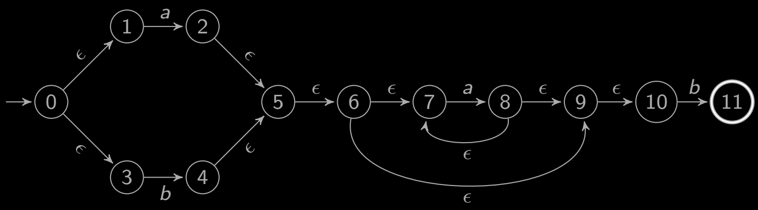
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4	a, b	

NFA to DFA

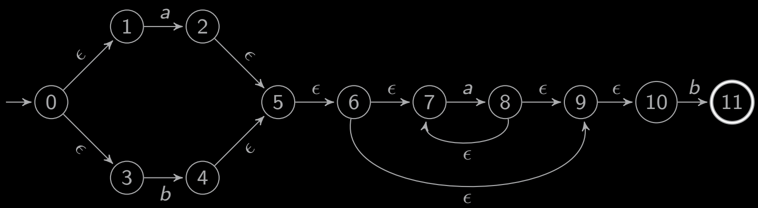
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4	a, b	ϕ

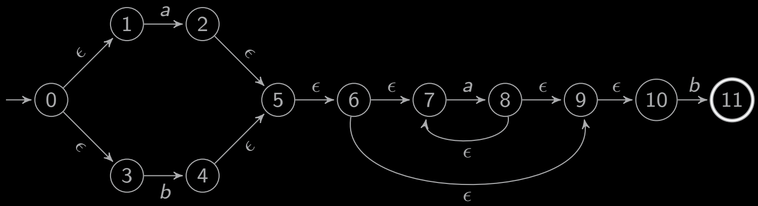
NFA to DFA

Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4	a, b	ϕ
ϕ		

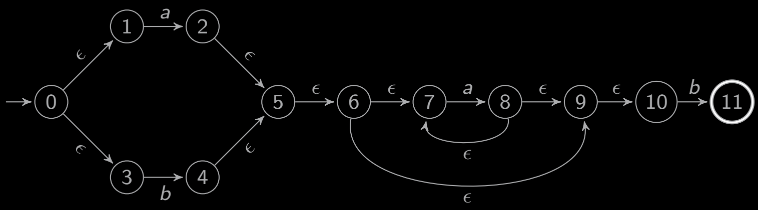
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4	a, b	ϕ
ϕ	a, b	

NFA to DFA

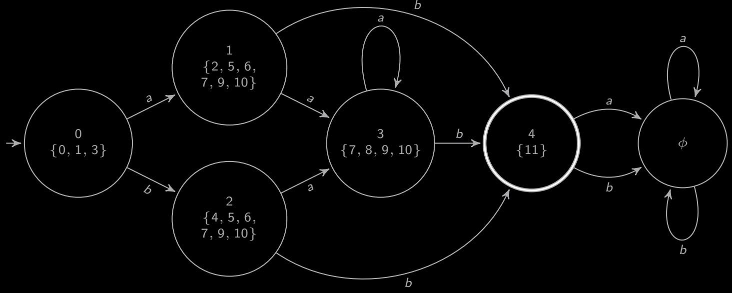
Example (DFA for $N_{(a|b)a*b}$)



r	a	$m(r, a)$
$\{0, 1, 3\} = 0$ (start state)	a	$\{2, 5, 6, 7, 9, 10\} = 1$
0	b	$\{4, 5, 6, 7, 9, 10\} = 2$
1	a	$\{7, 8, 9, 10\} = 3$
1	b	$\{11\} = 4$ (final state)
2	a	3
2	b	4
3	a	3
3	b	4
4	a, b	ϕ
ϕ	a, b	ϕ

NFA to DFA

NFA to DFA



Minimal DFA

Minimal DFA

To obtain a smaller but equivalent DFA, we partition the states such that the states in the new DFA are subsets of the states in the original DFA

Minimal DFA

To obtain a smaller but equivalent DFA, we partition the states such that the states in the new DFA are subsets of the states in the original DFA

The initial partition contains two subsets: the non-final states and the final states

Minimal DFA

To obtain a smaller but equivalent DFA, we partition the states such that the states in the new DFA are subsets of the states in the original DFA

The initial partition contains two subsets: the non-final states and the final states

We make sure that from each subset, on each input symbol, we transition into an identical subset; otherwise, we split the subset

Minimal DFA · Partitioning

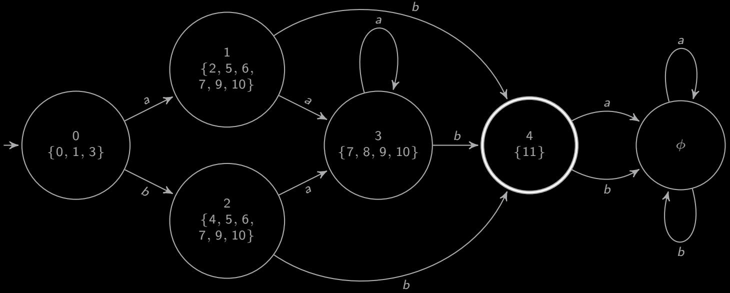
Input: a DFA $D = (\Sigma, S, s_0, M, F)$
Output: a partition of S

$\mathcal{P} \leftarrow \{S - F, F\}$
while splitting occurs **do**
  for $Q \in \mathcal{P}$ **do**
    if Q.size() > 1 **then**
      for $a \in \Sigma$ **do**
        $r \leftarrow$ a state chosen from Q
        $T \leftarrow$ the subset in the $\mathcal{P}$ containing $m(r, a)$
        $Q_1 \leftarrow \{s \in Q | m(s, a) \in T\}$
        $Q_2 \leftarrow \{s \in Q | m(s, a) \notin T\}$
        if $Q_2 \neq \{\}$ **then**
          replace Q in $\mathcal{P}$ by Q_1 and Q_2
          break
        end if
      end for
    end if
  end for
end while
return $\mathcal{P}$

Minimal DFA

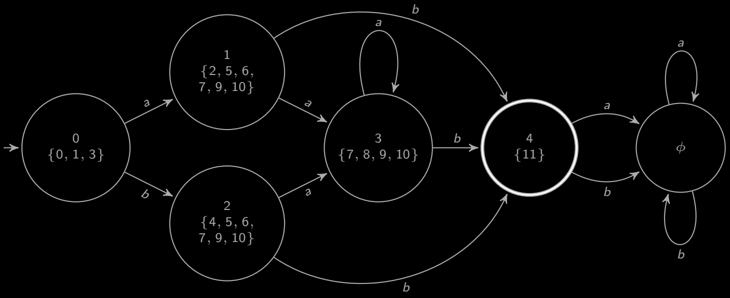
Minimal DFA

Example (minimal DFA for $D_{(a|b)a*b}$)



Minimal DFA

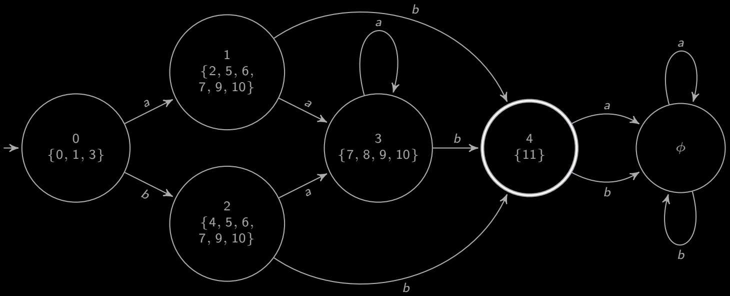
Example (minimal DFA for $D_{(a|b)a*b}$)



Initial partition $\mathcal{P} = \{\{0, 1, 2, 3, \phi\}, \{4\}\}$

Minimal DFA

Example (minimal DFA for $D_{(a|b)a*b}$)

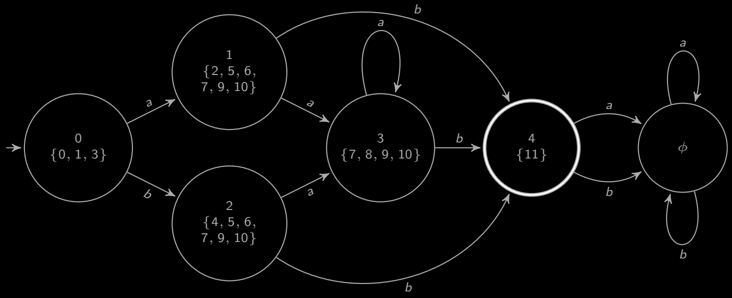


Initial partition $\mathcal{P} = \{\{0, 1, 2, 3, \phi\}, \{4\}\}$

The input symbol a does not split the subset $Q = \{0, 1, 2, 3, \phi\}$

Minimal DFA

Example (minimal DFA for $D_{(a|b)a*b}$)



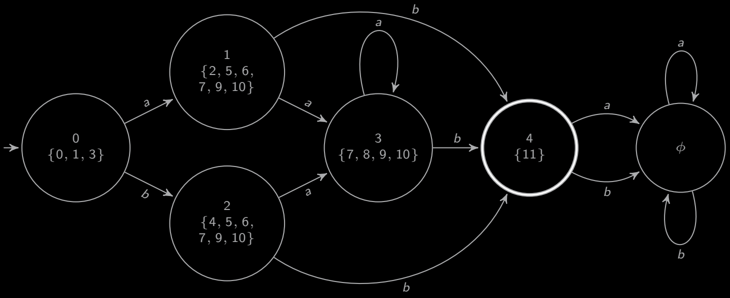
Initial partition $\mathcal{P} = \{\{0, 1, 2, 3, \phi\}, \{4\}\}$

The input symbol a does not split the subset $Q = \{0, 1, 2, 3, \phi\}$

The input symbol b splits the subset $Q = \{0, 1, 2, 3, \phi\}$ into $Q_1 = \{0, \phi\}$ and $Q_2 = \{1, 2, 3\}$

Minimal DFA

Example (minimal DFA for $D_{(a|b)a*b}$)



Initial partition $\mathcal{P} = \{\{0, 1, 2, 3, \phi\}, \{4\}\}$

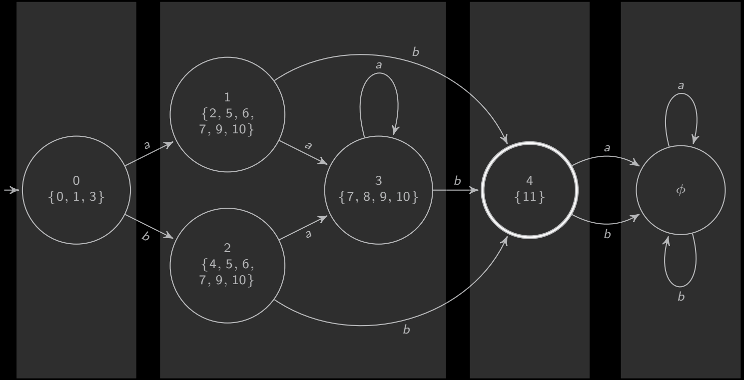
The input symbol a does not split the subset $Q = \{0, 1, 2, 3, \phi\}$

The input symbol b splits the subset $Q = \{0, 1, 2, 3, \phi\}$ into $Q_1 = \{0, \phi\}$ and $Q_2 = \{1, 2, 3\}$

The new and final partition is $\mathcal{P} = \{\{0\}, \{1, 2, 3\}, \{4\}, \{\phi\}\}$

Minimal DFA

Minimal DFA



Minimal DFA