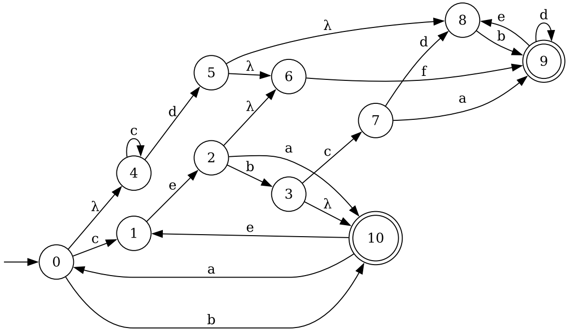
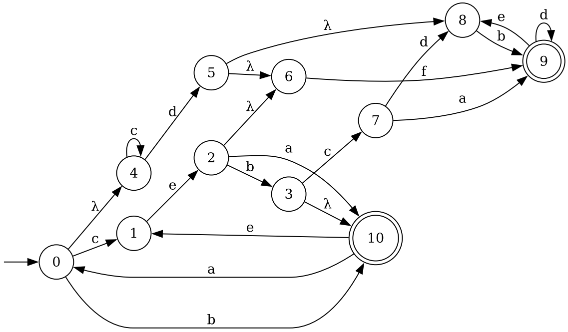
Why Don't We Use NFAs during Scanning?



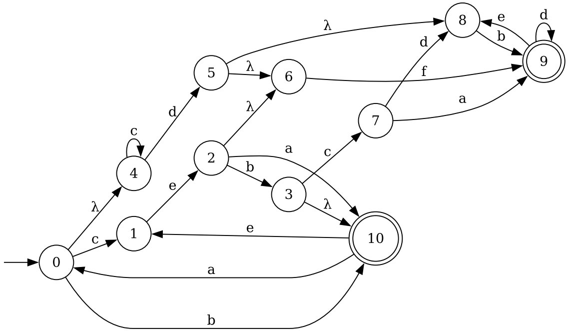
i. How can we represent an NFA in computer memory?

Why Don't We Use NFAs during Scanning?



i. How can we represent an NFA in computer memory?
   An $n \times n$ Boolean matrix for λs, and a $state \times c \in \Sigma$ "transition table" whose cells contain what?

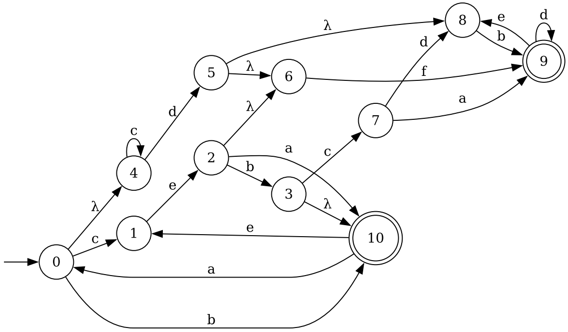
Why Don't We Use NFAs during Scanning?



i. How can we represent an NFA in computer memory?
An $n \times n$ Boolean matrix for λs, and a $state \times c \in \Sigma$ "transition table" whose cells contain what?

ii. While matching a character sequence to an NFA, what type of data structure must be used to remember where in the NFA we are?

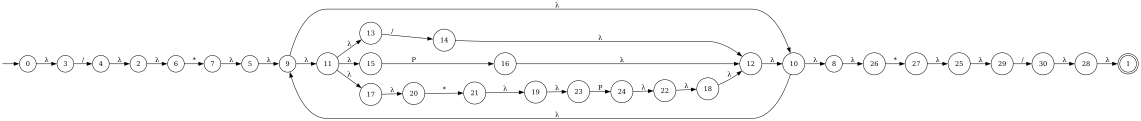
Why Don't We Use NFAs during Scanning?



i. How can we represent an NFA in computer memory?
   An $n \times n$ Boolean matrix for λs, and a $state \times c \in \Sigma$ "transition table" whose cells contain what?
ii. While matching a character sequence to an NFA, what type of data structure must be used to remember where in the NFA we are?
iii. Can we represent a DFA more efficiently?
iv. What data structure is required to remember DFA matching state?

Why Don't We Use NFAs during Scanning?

An example of the simple `/* C/C++ comment */` RE converted to an NFA using automated tools (in fact, all of which you will build in this course!). . .



c++comment-automated.pdf

NFA to DFA Algorithm
initialization

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f

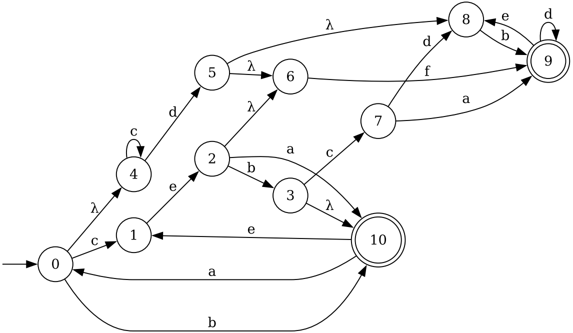
$A = \{9, 10\}$

$i = 0$

Stack L <empty>

procedure NFAtoDFA(N an NFA)
Let $T[row][col]$ be an empty transition table defining D. $T[row][\cdot]$ is uniquely identified by a set of states from N, each $T[\cdot][col]$ uniquely identifies a character $c \in \Sigma$.

let L be an empty stack
let A be the set of accepting states for N
let i be the starting state of N



NFA to DFA Algorithm
FollowLambda

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f

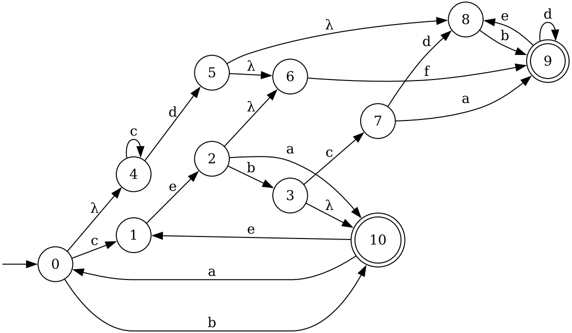
$A = \{9, 10\}$

$i = 0$

Stack L <empty>

```
procedure NFAtoDFA( N an NFA )
Let T[row][col] be an empty transition table defining
D. T[row][·] is uniquely identified by a set of
states from N, each T[·][col] uniquely identifies
a character c ∈ Σ.

  let L be an empty stack
  let A be the set of accepting states for N
  let i be the starting state of N
  B ← FollowLamda( {i} )
```



NFA to DFA Algorithm
FollowLambda

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f

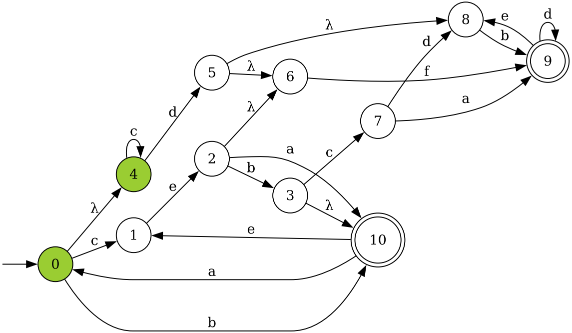
$A = \{9, 10\}$

$i = 0$

Stack L <empty>

```
procedure FollowLambda( S a ⊆ of NFA N states )
returns the set of NFA states encountered by
recursively following only λ transitions
from states in S

  Let M be an empty stack
  foreach ( state t ∈ S ) push t onto M
  while ( |M| > 0 ) do (
    t ← pop M
    foreach ( λ transition from t to state q ) do (
      if ( q ∉ S ) then (
        add q to S
        push q onto M
      )
    )
  )
  return S
```



NFA to DFA Algorithm
FollowLambda

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}						

$A = \{9, 10\}$

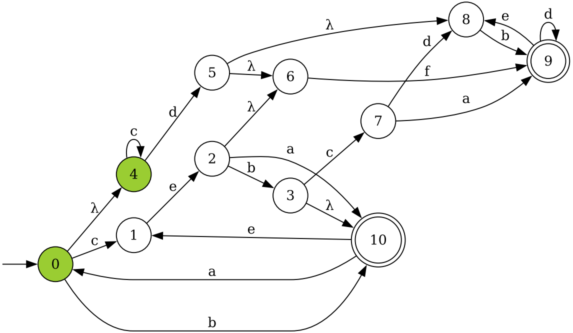
$i = 0$

Stack L <$\{0,4\}$>

$B = \{0,4\}$

procedure NFAtoDFA(N an NFA)
Let $T[row][col]$ be an empty transition table defining D. $T[row][\cdot]$ is uniquely identified by a set of states from N, each $T[\cdot][col]$ uniquely identifies a character $c \in \Sigma$.

```
  let L be an empty stack
  let A be the set of accepting states for N
  let i be the starting state of N
  B ← FollowLamda( {i} )
  initialize row T[B][·]
  mark T[B][·] as the starting state of D
  if ( A ⋂ B ≠ ∅ ) then (
    mark T[B][·] as an accepting state of D
  )
  push B onto L
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}						

$A = \{9, 10\}$

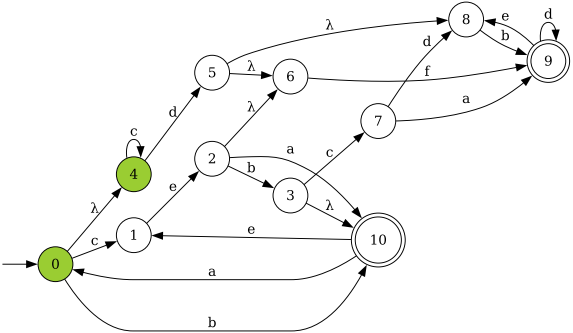
$i = 0$

Stack L <empty>

$S = \{0, 4\}$

$c = $ a

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm
FollowChar

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}						

$A = \{9, 10\}$

$i = 0$

Stack L <empty>

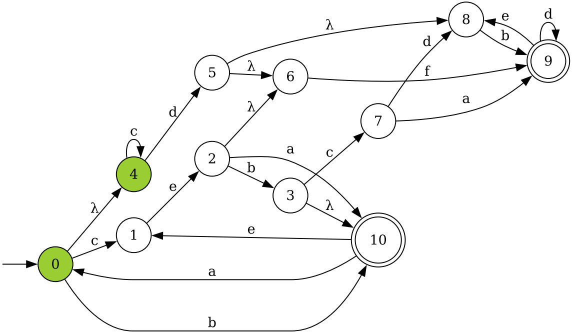
$S = \{0, 4\}$

$c =$ a

$\emptyset \leftarrow FollowChar(S, c)$

```
procedure FollowChar( S a ⊆ of NFA N states, c ∈ Σ )
returns the set of NFA states obtained from following
all c transitions from states in S

  Let F be an empty set
  foreach ( state t ∈ S ) do (
    foreach ( c transition from t to state q ) do (
      add q to F
    )
  )
  return F
```



NFA to DFA Algorithm
FollowLambda

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}						

$A = \{9, 10\}$

$i = 0$

Stack L <empty>

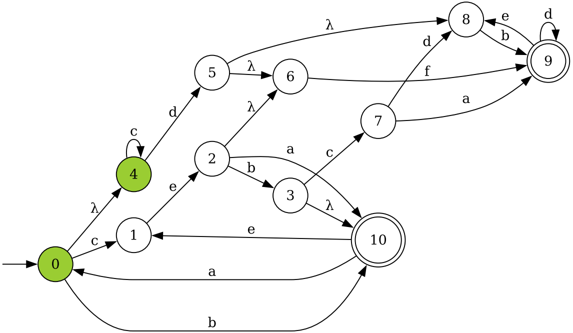
$S = \{0, 4\}$

$c =$ a

$\emptyset \leftarrow FollowLambda(\emptyset)$

```
procedure FollowLambda( S a ⊆ of NFA N states )
returns the set of NFA states encountered by
recursively following only λ transitions
from states in S

  Let M be an empty stack
  foreach ( state t ∈ S ) push t onto M
  while ( |M| > 0 ) do (
    t ← pop M
    foreach ( λ transition from t to state q ) do (
      if ( q ∉ S ) then (
        add q to S
        push q onto M
      )
    )
  )
  return S
```



NFA to DFA Algorithm

discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	$\emptyset$					

$A = \{9, 10\}$

$i = 0$

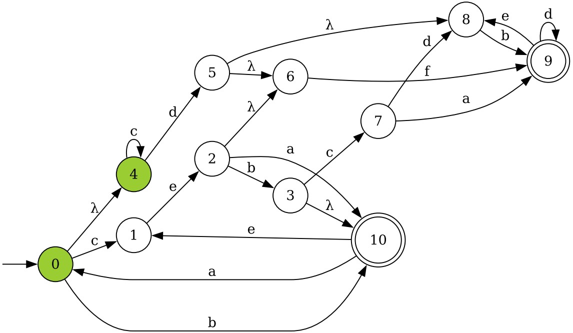
Stack L <empty>

$S = \{0, 4\}$

$c =$ a

$R = \emptyset$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	$\emptyset$					

$A = \{9, 10\}$

$i = 0$

Stack L <empty>

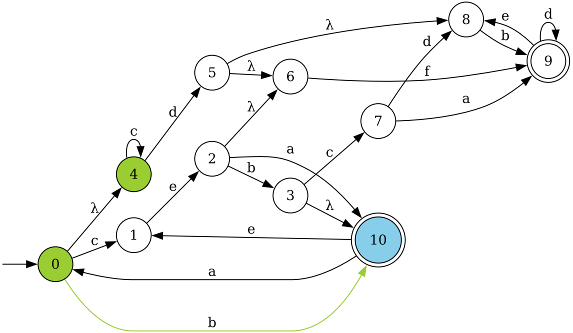
$S = \{0, 4\}$

$c =$ b

$\{10\} \leftarrow FollowChar(S, c)$

procedure FollowChar(S a $\subseteq$ of NFA N states, $c \in \Sigma$)
returns the set of NFA states obtained from following
all c transitions from states in S

```
  Let F be an empty set
  foreach ( state t ∈ S ) do (
    foreach ( c transition from t to state q ) do (
      add q to F
    )
  )
  return F
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	$\emptyset$					

$A = \{9, 10\}$

$i = 0$

Stack L <empty>

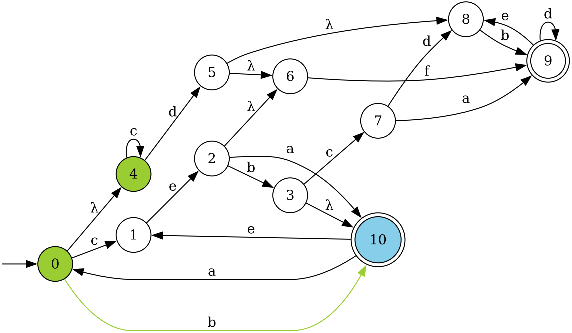
$S = \{0, 4\}$

$c =$ b

$\{10\} \leftarrow FollowLambda(\{10\})$

```
procedure FollowLambda( S a ⊆ of NFA N states )
returns the set of NFA states encountered by
recursively following only λ transitions
from states in S

  Let M be an empty stack
  foreach ( state t ∈ S ) push t onto M
  while ( |M| > 0 ) do (
    t ← pop M
    foreach ( λ transition from t to state q ) do (
      if ( q ∉ S ) then (
        add q to S
        push q onto M
      )
    )
  )
  return S
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}				
N	Y	{10}						

$A = \{9, 10\}$

$i = 0$

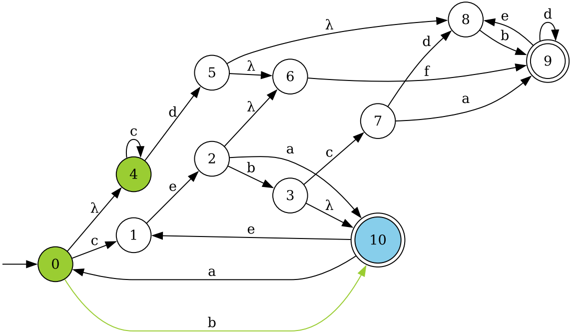
Stack L <$\{10\}$>

$S = \{0, 4\}$

$c =$ b

$R = \{10\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}				
N	Y	{10}						

$A = \{9, 10\}$

$i = 0$

Stack L <{10}>

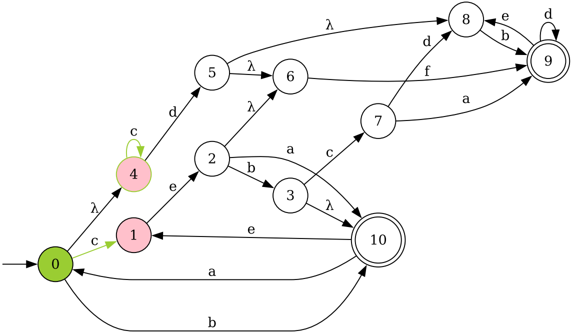
$S = \{0, 4\}$

$c =$ c

$\{1, 4\} \leftarrow FollowChar(S, c)$

```
procedure FollowChar( S a ⊆ of NFA N states, c ∈ Σ )
returns the set of NFA states obtained from following
all c transitions from states in S

  Let F be an empty set
  foreach ( state t ∈ S ) do (
    foreach ( c transition from t to state q ) do (
      add q to F
    )
  )
  return F
```



NFA to DFA Algorithm
discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}				
N	Y	{10}						

$A = \{9, 10\}$

$i = 0$

Stack L <{10}>

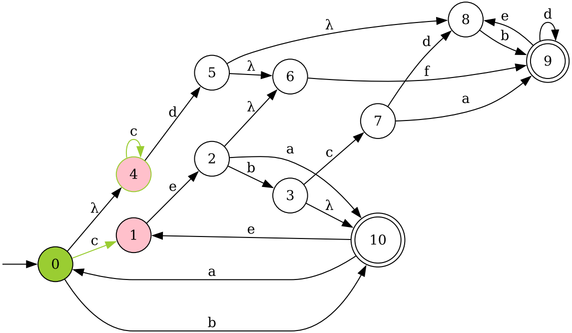
$S = \{0, 4\}$

$c = $ c

$\{1, 4\} \leftarrow FollowLambda(\{1, 4\})$

```
procedure FollowLambda( S a ⊆ of NFA N states )
returns the set of NFA states encountered by
recursively following only λ transitions
from states in S

  Let M be an empty stack
  foreach ( state t ∈ S ) push t onto M
  while ( |M| > 0 ) do (
    t ← pop M
    foreach ( λ transition from t to state q ) do (
      if ( q ∉ S ) then (
        add q to S
        push q onto M
      )
    )
  )
  return S
```



NFA to DFA Algorithm

discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}			
N	Y	{10}						
N	N	{1,4}						

$A = \{9, 10\}$

$i = 0$

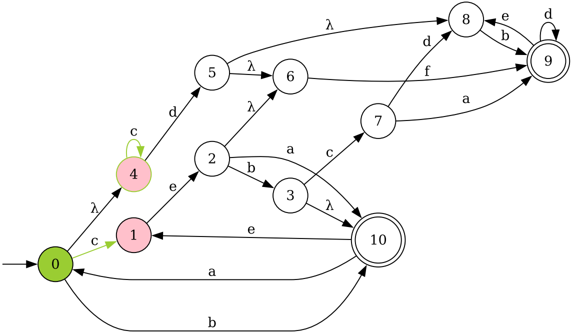
Stack L <{1,4},{10}>

$S = \{0,4\}$

$c =$ c

$R = \{1,4\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm

discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}			
N	Y	{10}						
N	N	{1,4}						

$A = \{9, 10\}$

$i = 0$

Stack L <{1,4},{10}>

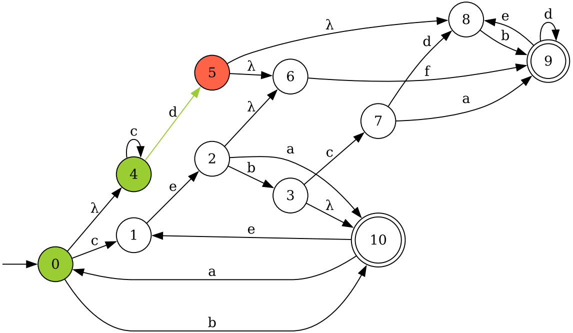
$S = \{0,4\}$

$c =$ d

$\{5\} \leftarrow FollowChar(S, c)$

```
procedure FollowChar( S a ⊆ of NFA N states, c ∈ Σ )
returns the set of NFA states obtained from following
all c transitions from states in S

  Let F be an empty set
  foreach ( state t ∈ S ) do (
    foreach ( c transition from t to state q ) do (
      add q to F
    )
  )
  return F
```



NFA to DFA Algorithm

discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}			
N	Y	{10}						
N	N	{1,4}						

$A = \{9, 10\}$

$i = 0$

Stack L <{1,4},{10}>

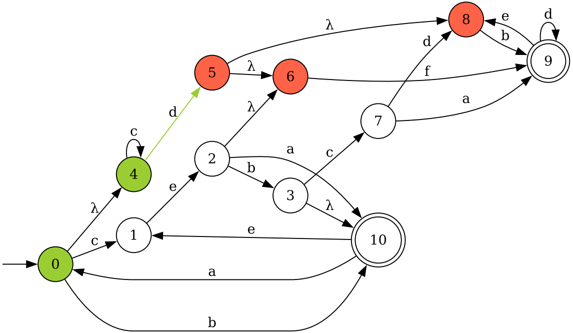
$S = \{0,4\}$

$c =$ d

$\{5,6,8\} \leftarrow FollowLambda(\{5\})$

```
procedure FollowLambda( S a ⊆ of NFA N states )
returns the set of NFA states encountered by
recursively following only λ transitions
from states in S

  Let M be an empty stack
  foreach ( state t ∈ S ) push t onto M
  while ( |M| > 0 ) do (
    t ← pop M
    foreach ( λ transition from t to state q ) do (
      if ( q ∉ S ) then (
        add q to S
        push q onto M
      )
    )
  )
  return S
```



NFA to DFA Algorithm

discover new state sets

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	$\emptyset$	{10}	{1,4}	{5,6,8}		
N	Y	{10}						
N	N	{1,4}						
N	N	{5,6,8}						

$A = \{9, 10\}$

$i = 0$

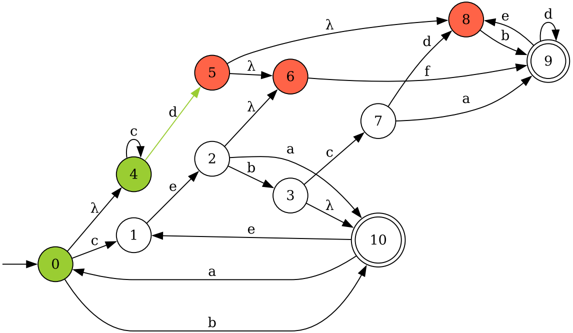
Stack L <$\{5,6,8\}, \{1,4\}, \{10\}$>

$S = \{0,4\}$

$d =$ d

$R = \{5,6,8\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ∩ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm

characters e and f yield ∅

$A = \{9, 10\}$

$i = 0$

Stack L <$\{5,6,8\},\{1,4\},\{10\}$>

$S = \{0,4\}$

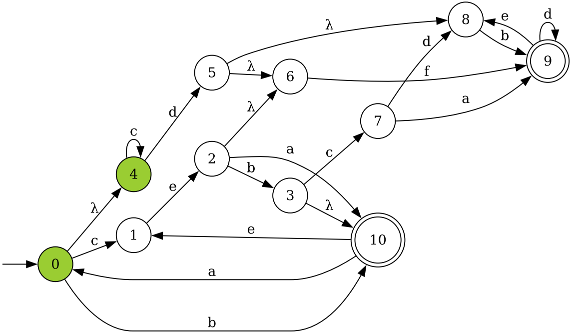
$c =$ f

$R = \emptyset$

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}						
N	N	{5,6,8}						

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm

pop L and do it all again

$A = \{9, 10\}$

$i = 0$

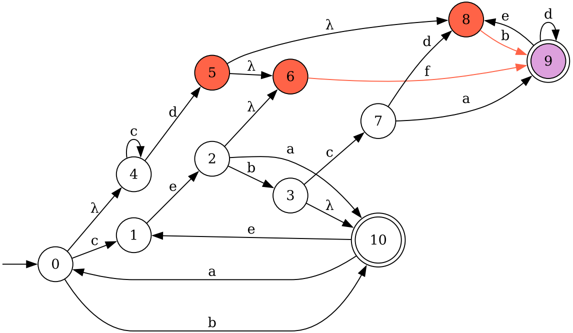
Stack L <$\{9\}, \{1,4\}, \{10\}$>

$S = \{5, 6, 8\}$

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}						
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}						

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm

pop L and do it all again

$A = \{9, 10\}$

$i = 0$

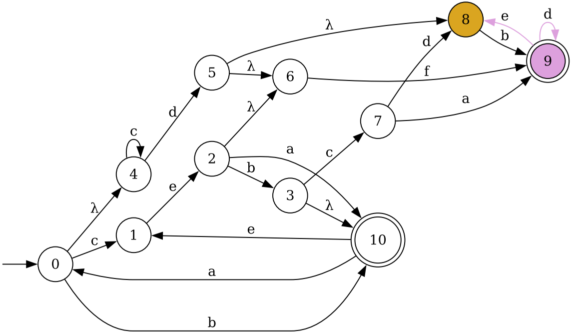
Stack L <$\{8\}, \{1,4\}, \{10\}$>

$S = \{9\}$

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}						
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}						

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm
pop L and do it all again

$A = \{9, 10\}$

$i = 0$

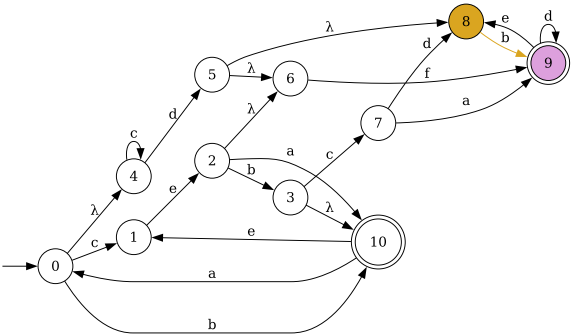
Stack L <$\{1,4\}, \{10\}$>

$S = \{8\}$

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}						
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}	∅	{9}	∅	∅	∅	∅

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```



NFA to DFA Algorithm

pop L and do it all again

$A = \{9,10\}$

$i = 0$

Stack L <$\{2,6\},\{4\},\{10\}$>

$S = \{1,4\}$

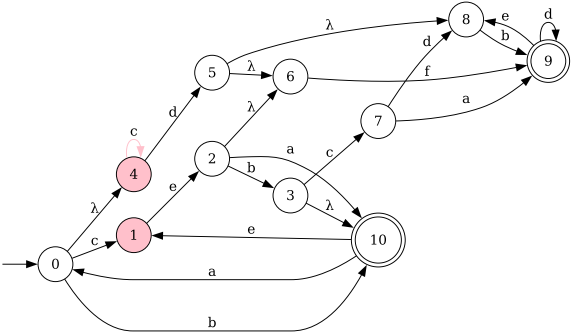
$c =$ c

$R = \{4\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}	∅	∅	{4}			
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}	∅	{9}	∅	∅	∅	∅
N	N	{4}						



NFA to DFA Algorithm

pop L and do it all again

$A = \{9, 10\}$

$i = 0$

Stack L <$\{2,6\},\{4\},\{10\}$>

$S = \{1,4\}$

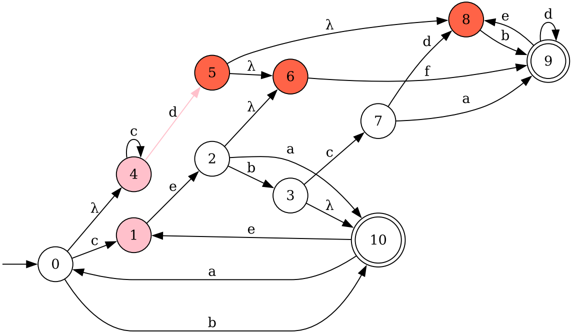
$c =$ d

$R = \{5,6,8\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}	∅	∅	{4}	{5,6,8}		
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}	∅	{9}	∅	∅	∅	∅
N	N	{4}						



NFA to DFA Algorithm
pop L and do it all again

$A = \{9, 10\}$

$i = 0$

Stack L <$\{2,6\}, \{4\}, \{10\}$>

$S = \{1,4\}$

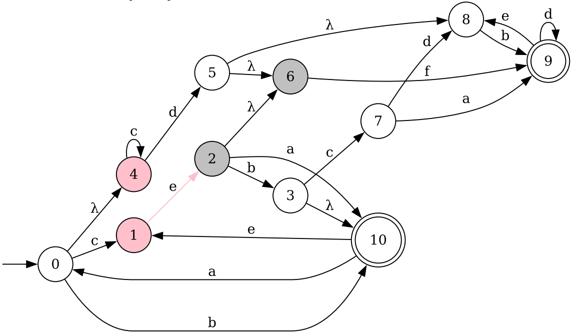
$c =$ e

$R = \{2,6\}$

```
repeat (
  S ← pop L
  foreach ( c ∈ Σ ) do (
    R ← FollowLambda(FollowChar(S,c))
    T[S][c] ← R
    if ( |R| > 0 AND T[R][·] does not exist ) then (
      initialize row T[R][·]
      if ( A ⋂ R ≠ ∅ ) then (
        mark T[R][·] as an accepting state of D
      )
      push R onto L
    )
  )
) while ( |L| > 0 )
```

Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}						
N	N	{1,4}	∅	∅	{4}	{5,6,8}	{2,6}	
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}	∅	{9}	∅	∅	∅	∅
N	N	{4}						
N	N	{2,6}						



NFA to DFA Algorithm

$A = \{9, 10\}$

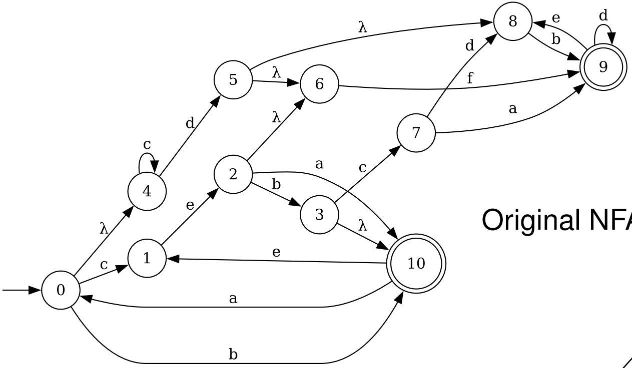
$i = 0$

Stack L <empty>

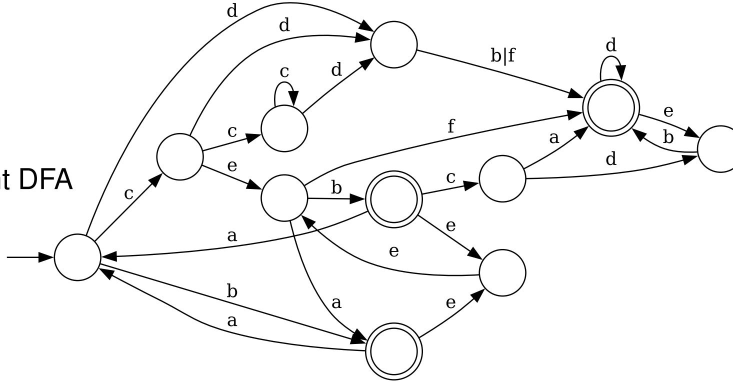
Transition Table T

is Start	is Accept	State	a	b	c	d	e	f
Y	N	{0,4}	∅	{10}	{1,4}	{5,6,8}	∅	∅
N	Y	{10}	{0,4}	∅	∅	∅	{1}	∅
N	N	{1,4}	∅	∅	{4}	{5,6,8}	{2,6}	∅
N	N	{5,6,8}	∅	{9}	∅	∅	∅	{9}
N	Y	{9}	∅	∅	∅	{9}	{8}	∅
N	N	{8}	∅	{9}	∅	∅	∅	∅
N	N	{4}	∅	∅	{4}	{5,6,8}	∅	∅
N	N	{2,6}	{10}	{3,10}	∅	∅	∅	{9}
N	Y	{3,10}	{0,4}	∅	{7}	∅	{1}	∅
N	N	{7}	{9}	∅	∅	{8}	∅	∅
N	N	{1}	∅	∅	∅	∅	{2,6}	∅

When L is empty, the table T holds a DFA derived from the original NFA.



Original NFA



Equivilant DFA