

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



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



i. How can we represent an NFA in computer memory?
 An n × n Boolean matrix for λs, and a state × c ∈ Σ "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?



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

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 Tis Start is Accept State a b c d e f

 $A = \{9, 10\}$ i = 0Stack 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 Nlet i be the starting state of N



NFA to DFA Algorithm FollowLambda

Transition Table Tis Start is Accept State a b c d e f

 $A = \{9, 10\}$ i = 0Stack 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 $B \leftarrow$ FollowLamda($\{i\}$)



NFA to DFA Algorithm FollowLambda

Transition Table Tis Start is Accept State a b c d e f

 $A = \{9, 10\}$ i = 0Stack L <empty>

```
procedure FollowLambda(S a \subset of NFA N states)
returns the set of NFA states encountered by
recursively following only \lambda transitions
from states in S
  Let M be an empty stack
  foreach ( state t \in S ) push t onto M
  while (|M| > 0) do (
    t \leftarrow \text{pop } M
    foreach ( \lambda transition from t to state q ) do (
                                                                                              3
      if (a \notin S) then (
         add a to S
                                                                                                        10
         push q onto M
                                                                                     а
                                                                                     b
  return S
```

NFA to DFA Algorithm	Transition Table T											
FollowLambda	is Start	is Accept	State	а	b	С	d	е	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][corl] be an empty transition table defining *D*. $T[row][\cdot]$ is uniquely identified by a set of states from *N*, each $T[\cdot][corl]$ 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 \leftarrow FollowLamda (\{i\})
initialize row T[B][\cdot]
mark T[B][\cdot] as the starting state of D
if (A \cap B \neq \emptyset) then (
mark T[B][\cdot] as an accepting state of D
)
push B onto L
```



NEA to DEA Algorithm	Transition Table T								
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f
	Y	N	{0,4}						
$A = \{9, 10\}$									
i = 0									
Stack L <empty></empty>									
$S = \{0,4\}$									
c = a									
<pre>repeat ($S \leftarrow \text{pop } L$ foreach ($c \in \Sigma$) do ($R \leftarrow \text{FollowLambda(FollowChar(S, c))}$ $T[S][c] \leftarrow R$ if ($R > 0$ AND $T[R][\cdot]$ does not exi initialize row $T[R][\cdot]$ if ($A \cap R \neq \emptyset$) then (mark $T[R][\cdot]$ as an accepting st) push R onto L))) while ($L > 0$)</pre>	st) then (5 2 2 a	λ λ e	a 3		7	d v

NFA to DFA Algorithm	Transition Table T											
FollowChar	is Start	is Accept	State	а	b	С	d	е	f			
	Y	N	{0,4}									
$A=\{9,10\}$												
i = 0												
Stack L <empty></empty>												
$S = \{0,4\}$												
c = a												

```
\emptyset \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 \in S ) do (

foreach ( c transition from t to state q ) do (

add q to F

)

return F
```



NFA to DFA Algorithm		Trar	sition T	able	Т				
FollowLambda	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}						
$A = \{9, 10\}$									
i = 0									
Stack $L < empty >$									
$S = \{0, 4\}$									
c = a									
$\emptyset \leftarrow FollowLambda(\emptyset)$									
procedure FollowLambda ($S \ a \subseteq of \ NFA$ returns the set of NFA states encountrecursively following only λ transition from states in S Let M be an empty stack foreach (state $t \in S$) push t onto M while ($ M > 0$) do ($t \leftarrow pop \ M$ foreach (λ transition from t to sif ($q \notin S$) then (add q to S push q onto M)	N states) ered by ons			5 2 4	h h e	6 a 3		7	d 8 d
)		\backslash		b		_			
return S									

NFA to DFA Algorithm		Tran	sition T	able	Т				
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f
	Y	N	{0,4}	Ø					
$A = \{9, 10\}$									
i = 0									
Stack L <empty></empty>									
$S = \{0,4\}$									
c = a									
$R = \emptyset$									
$\begin{array}{l} \textbf{repeat} (\\ S \leftarrow \text{ pop } L\\ \textbf{foreach} (c \in \Sigma) \textbf{ do } (\\ R \leftarrow \text{FollowLambda}(\text{FollowChar}(S,c))\\ T[S][c] \leftarrow R\\ \textbf{if} (R > 0 \textbf{ AND } T[R][\cdot] \text{ does not exis}\\ \text{ initialize row } T[R][\cdot]\\ \textbf{if} (A \cap R \neq \emptyset) \textbf{ then } (\\ mark \; T[R][\cdot] \text{ as an accepting sta})\\ \text{ push } R \text{ onto } L\\)\\ \end{pmatrix}\\ \textbf{while } (L > 0) \end{array}$	t) then (ite of D			5 2 2 a b	h h e	6 a		7	d f a

NFA to DFA Algorithm	Transition Table T											
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f			
	Y	N	{0,4}	Ø								
$A = \{9, 10\}$												
i = 0												
Stack L <empty></empty>												

```
S = \{0, 4\}

c = b

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

 ${\tt procedure}$ 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 \in S ) do (

foreach ( c transition from t to state q ) do (

add q to F

)

return F
```



NFA to DFA Algorithm		Tran	sition Ta	able	Т					
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f	
	Y	Ν	{0,4}	Ø						
$A = \{9, 10\}$										
i = 0										
Stack <i>L</i> <empty></empty>										
$S = \{0, 4\}$										
c = b										
$\{10\} \leftarrow \textit{FollowLambda}(\{10\})$										
procedure FollowLambda(S a \subseteq of NFA N returns the set of NFA states encounter recursively following only λ transition from states in S	'states) red by ns			5	λ.	6	λ		d f	;);
Let M be an empty stack foreach (state $t \in S$) push t onto M while ($ M > 0$) do (° d	$\sqrt{2}$	2	a	c ×	7)-		-
$t \leftarrow \text{pop } M$ foreach (λ transition from t to st	ate <i>a</i>) do	(4 e/		P.	$\overrightarrow{3}$				
if ($q \notin S$) then (``````````````````````````````````````			e					
add q to s push q onto M								10		

a b

, return S

NEA to DEA Algorithm		Tra	nsition	Tabl	e T					
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f	
	Y	N	{0,4}	Ø	{10}					
$A = \{9, 10\}$	Ν	Y	{10}							
i = 0										
Stack $L < \{10\} >$										
$S = \{0,4\}$										
c = b										
$R = \{10\}$										
$\begin{array}{l} \textbf{repeat} (\\ S \leftarrow \text{ pop } L\\ \textbf{foreach } (c \in \Sigma) \textbf{ do } (\\ R \leftarrow \text{FollowLambda}(\text{FollowChar}(S,c))\\ T[S][c] \leftarrow R\\ \textbf{if} (R > 0 \textbf{ AND } T[R][\cdot] \text{ does not exi}.\\ initialize row T[R][\cdot]\\ \textbf{if} (A \cap R \neq \emptyset) \textbf{ then } (\\ mark \ T[R][\cdot] \text{ as an accepting st} \\)\\ \text{push } R \text{ onto } L\\)\\ \end{pmatrix}\\ \textbf{ while } (L > 0) \end{array}$	st) then (ate of D		c d e u	5 2 2 a b	h h e	A C A	7	d f	a	

NEA to DEA Algorithm		Tra	nsition	Tabl	e T				
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}				
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0									
Stack $L < \{10\} >$									
$S = \{0, 4\}$									
c = c									
$\{1,4\} \leftarrow FollowChar(S,c)$									
procedure FollowChar(S a \subseteq of NFA N returns the set of NFA states obtaine	states, $c\in\Sigma$ d from follow) ving				λ			8

returns the set of NFA states obtained from following all c transitions from states in ${\cal S}$

```
Let F be an empty set

foreach ( state t \in S ) do (

foreach ( c transition from t to state q ) do (

add q to F

)

return F
```



NFA to DFA Algorithm	Transition Table T									
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f	
	Y	N	{0,4}	Ø	{10}					
$A = \{9, 10\}$	Ν	Y	{10}							
i = 0										
Stack $L < \{10\} >$										
$S = \{0, 4\}$										
c = c										
$\{1,4\} \leftarrow FollowLambda(\{1,4\}$										
<pre>procedure FollowLambda(S a \subseteq of NFA N returns the set of NFA states encounter recursively following only λ transition from states in S Let M be an empty stack foreach (state $t \in S$) push t onto M while ($M > 0$) do ($t \leftarrow pop M$ foreach (λ transition from t to st if ($q \notin S$) then (add q to S push q onto M)))</pre>	states) red by is ate q) do			5 2 2 a b	h a b 3 e	2 2 2	7	d f	a	

NFA to DFA Algorithm	Transition Table T									
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f	
-	Y	Ν	{0,4}	Ø	{10}	{1,4}				
$A = \{9, 10\}$	Ν	Y	{10}							
i = 0	Ν	N	{1,4}							
Stack $L < \{1, 4\}, \{10\} >$										
$S=\{0,4\}$										
c = c										
$R = \{1, 4\}$										
<pre>repeat ($S \leftarrow \text{pop } L$ foreach ($c \in \Sigma$) do ($R \leftarrow \text{FollowLambda}(\text{FollowChar}(S, c))$ $T[S][c] \leftarrow R$ if ($R > 0$ AND $T[R][\cdot]$ does not exist initialize row $T[R][\cdot]$ if ($A \cap R \neq \emptyset$) then (mark $T[R][\cdot]$ as an accepting sta) push R onto L))) while ($L > 0$)</pre>	t) then (5 2 2 a	h h b 3 e			a	e b	d e

NEA to DEA Algorithm	Transition Table T								
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f
=	Y	N	{0,4}	Ø	{10}	{1,4}			
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	Ν	{1,4}						
Stack $L < \{1, 4\}, \{10\} >$									
$S = \{0, 4\}$									
c = d									
$\{5\} \leftarrow FollowChar(S,c)$									
procedure FollowChar(S a \subseteq of NFA N streturns the set of NFA states obtained all c transitions from states in S	ates, $c\in\Sigma$ from follow) ving		5	À. C	λ	d	8	e b
Let F be an empty set foreach (state $t \in S$) do (foreach (c transition from t to sta add q to F	te q) ${f do}$	(c d		h a	c 7	5	a	

```
return F
```



NFA to DFA Algorithm		т	ransitio	n Tal	ble T					
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f	
	Y	N	{0,4}	Ø	{10}	{1,4}				
$A = \{9, 10\}$	Ν	Y	{10}							
i = 0	Ν	Ν	{1,4}							
Stack $L < \{1, 4\}, \{10\} >$										
$S = \{0, 4\}$										
c = d										
$\{5,6,8\} \leftarrow \textit{FollowLambda}(\{5\})$										
procedure FollowLambda($S \ a \subseteq$ of NFA N returns the set of NFA states encounte recursively following only λ transition from states in S	′states) red by ns			5	λ 6	λ	d f	8	e b	d 9
Let M be an empty stack foreach (state $t \in S$) push t onto M while ($ M > 0$) do ($t \leftarrow$ pop M			c d	2	λ a b	с 7	<u>} </u>	a		
foreach (λ transition from t to st if ($a \notin S$) then (ate q) ${f do}$	(2			-(3)	A				
add q to S					е)			
push q onto M)				а			//			
)		\backslash								

return S

b

NFA to DFA Algorithm	Transition Table T								
discover new state sets	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}		
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	N	{1,4}						
Stack $L < \{5, 6, 8\}, \{1, 4\}, \{10\} >$	Ν	N	{5,6,8}						
$S = \{0, 4\}$									
d = d									
$R = \{5, 6, 8\}$									
$\begin{array}{l} \textbf{repeat} (\\ S \leftarrow \text{ pop } L\\ \textbf{foreach } (\ c \in \Sigma \) \ \textbf{do} \ (\\ R \leftarrow \text{ FollowLambda} (\text{FollowChar} (S, c))\\ T[S][c] \leftarrow R\\ \textbf{if} \ (\ R > 0 \ \textbf{AND} \ T[R][\cdot] \ \text{does not exi}\\ \text{initialize row} \ T[R][\cdot]\\ \textbf{if} \ (\ A \cap R \neq \emptyset \) \ \textbf{then} \ (\\ mark \ T[R][\cdot] \ \text{as an accepting st}\\)\\ \text{push } R \ \text{onto } L\\)\\ \end{pmatrix}\\ \textbf{while} \ (\ L > 0 \) \end{array}$) .st) then (tate of <i>D</i>			5 2 a b	$\frac{\lambda}{b}$ $\frac{a}{3}$ $\frac{a}{b}$ $\frac{a}{3}$ $\frac{b}{a}$		d f a	e b	

NFA to DFA Algorithm	Transition Table T								
characters e and f yield \emptyset	is Start	is Accept	State	а	b	С	d	е	f
	Y	N	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	N	{1,4}						
Stack $L < \{5, 6, 8\}, \{1, 4\}, \{10\} >$	Ν	N	{5,6,8}						
$S = \{0, 4\}$									
c = f									
$R = \emptyset$									
<pre>repeat ($S \leftarrow \text{ pop } L$ foreach ($c \in \Sigma$) do ($R \leftarrow \text{FollowLambda}(\text{FollowChar}(S, c))$ $T[S][c] \leftarrow R$ if ($R > 0$ AND $T[R][\cdot]$ does not exi initialize row $T[R][\cdot]$ if ($A \cap R \neq 0$) then (mark $T[R][\cdot]$ as an accepting st) push R onto L))) while ($L > 0$)</pre>) ast) then (tate of <i>D</i>			5 2 a b	λ 6 λ e	λ (7) (10	d a a	e b	

NFA to DFA Algorithm	Transition Table T									
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f	
	Y	N	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø	
$A = \{9, 10\}$	N	Y	{10}							
i = 0	N	Ν	{1,4}							
Stack $L < \{9\}, \{1, 4\}, \{10\} >$	Ν	N	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}	
$S = \{5, 6, 8\}$	Ν	Y	{9}							





NFA to DFA Algorithm	hm Transition Table T								
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f
	Y	N	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	Ν	{1,4}						
Stack $L < \{8\}, \{1, 4\}, \{10\} >$	Ν	Ν	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}
$S = \{9\}$	Ν	Y	{9}	Ø	Ø	Ø	{9 }	{8}	Ø
~ (^)	Ν	Ν	{8}						
repeat ($S \leftarrow ext{ pop } L$						λ	8	e d	





NEA to DEA Algorithm	Transition Table T								
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	Ν	{1,4}						
Stack $L < \{1, 4\}, \{10\} >$	Ν	N	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}
$S = \{8\}$	Ν	Y	{9}	Ø	Ø	Ø	{9}	{8}	Ø
	Ν	N	{8}	Ø	{9}	Ø	Ø	Ø	Ø
<pre>repeat ($S \leftarrow \text{ pop } L$ foreach ($c \in \Sigma$) do ($R \leftarrow \text{FollowLambda}(\text{FollowChar}(S, c))$; $T[S][c] \leftarrow R$ if ($R > 0$ AND $T[R][\cdot]$ does not exi initialize row $T[R][\cdot]$ if ($A \cap R \neq 0$) then (mark $T[R][\cdot]$ as an accepting st) push R onto L))) while ($L > 0$)</pre>) Late of D			5 2 a b	$\frac{\lambda}{6}$	λ (7) (10	d a	d 9	١

NFA to DFA Algorithm	Transition Table T								
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	Ν	{1,4}	Ø	Ø	{4}			
Stack $L < \{2, 6\}, \{4\}, \{10\} >$	Ν	N	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}
$S = \{1, 4\}$	Ν	Y	{9}	Ø	Ø	Ø	{9}	{8}	Ø
c = c	Ν	N	{8}	Ø	{9}	Ø	Ø	Ø	Ø
$R = \{4\}$	Ν	Ν	{4}						
$\begin{array}{l} \textbf{repeat} (\\ S \leftarrow \text{ pop } L\\ \textbf{foreach} (c \in \Sigma) \textbf{ do } (\\ R \leftarrow \text{FollowLambda}(\text{FollowChar}(S,c))\\ T[S][c] \leftarrow R\\ \textbf{if} (R > 0 \textbf{ AND } T[R][\cdot] \text{ does not exi}\\ \text{initialize row } T[R][\cdot]\\ \textbf{if} (A \cap R \neq 0) \textbf{ then } (\\ mark \; T[R][\cdot] \text{ as an accepting st})\\ \text{push } R \text{ onto } L\\)\\ \end{pmatrix}\\ \textbf{while } (L > 0) \end{array}$	st) then (5 2 a b			d a	e d 9	١

NFA to DFA Algorithm	Transition Table T								
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	N	{1,4}	Ø	Ø	{4}	{5,6,8}		
Stack $L < \{2, 6\}, \{4\}, \{10\} >$	Ν	N	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}
$S = \{1, 4\}$	Ν	Y	{9}	Ø	Ø	Ø	{9}	{8}	Ø
c = d	Ν	N	{8}	Ø	{9}	Ø	Ø	Ø	Ø
$R = \{5, 6, 8\}$	Ν	Ν	{4}						
$\begin{array}{l} \textbf{repeat} (\\ S \leftarrow \text{ pop } L\\ \textbf{foreach } (c \in \Sigma) \textbf{ do } (\\ R \leftarrow \text{FollowLambda}(\text{FollowChar}(S,c))\\ T[S][c] \leftarrow R\\ \textbf{if} (R > 0 \textbf{ AND } T[R][\cdot] \text{ does not exi}\\ \text{ initialize row } T[R][\cdot]\\ \textbf{if} (A \cap R \neq 0) \textbf{ then } (\\ mark \; T[R][\cdot] \text{ as an accepting st})\\ \text{ push } R \text{ onto } L\\)\\ \end{pmatrix}\\ \textbf{ while } (L > 0) \end{array}$) Late of D			5 2 a b			d a a	e d 9)

NEA to DEA Algorithm	Transition Table T								
pop L and do it all again	is Start	is Accept	State	а	b	С	d	е	f
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø
$A = \{9, 10\}$	Ν	Y	{10}						
i = 0	Ν	Ν	{1,4}	Ø	Ø	{4}	{5,6,8}	{2,6}	
Stack $L < \{2, 6\}, \{4\}, \{10\} >$	Ν	Ν	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}
$S = \{1, 4\}$	Ν	Y	{9}	Ø	Ø	Ø	{9}	{8}	Ø
$c = \mathbf{P}$	Ν	Ν	{8}	Ø	{9}	Ø	Ø	Ø	Ø
$R = \begin{cases} 2 & 6 \end{cases}$	Ν	Ν	{4}						
$K = \{2, 0\}$	Ν	Ν	{2,6}				_		
repeat (λ		e d	
foreach ($c\in\Sigma$) do (\sim			d		
$R \leftarrow$ FollowLambda (FollowChar (S, c))			((5)-	$\frac{\lambda}{6}$		f		
$T[S][c] \leftarrow R$ if $(R > 0$ AND $T[R][\cdot]$ does not exit	st) then (d	\sim	λ		a		
initialize row $T[R][\cdot]$, , , , , , , , , , , , , , , , , , , 		° /	_/	/	$\sqrt{7}$			
if $(A \cap R \neq \emptyset)$ then ($\left(2\right)$	b a	c/			
mark $I[K][\cdot]$ as an accepting st	ate of D		e e	<u> </u>	× 3				
push R onto L		A A A A A A A A A A A A A A A A A A A			<u> </u>		\ \		
)					0	(10)		
) while ($ L >0$)		\rightarrow		a		~~			
		\backslash		h					

NFA to DFA Algorithm	Transition Table T										
	is Start	is Accept	State	а	b	С	d	е	f		
	Y	Ν	{0,4}	Ø	{10}	{1,4}	{5,6,8}	Ø	Ø		
$A = \{9, 10\}$	Ν	Y	{10}	{0,4}	Ø	Ø	Ø	{1}	Ø		
i = 0	Ν	Ν	{1,4}	Ø	Ø	{4}	{5,6,8}	{2,6}	Ø		
Stack $L < empty >$	Ν	Ν	{5,6,8}	Ø	{9}	Ø	Ø	Ø	{9}		
	Ν	Y	{9}	Ø	Ø	Ø	{9}	{8}	Ø		
	Ν	Ν	{8}	Ø	{9}	Ø	Ø	Ø	Ø		
	Ν	Ν	{4}	Ø	Ø	{4}	{5,6,8}	Ø	Ø		
	Ν	Ν	{2,6}	{10}	{3,10}	Ø	Ø	Ø	{9}		
	Ν	Y	{3,10}	{0,4}	Ø	{7}	Ø	{1}	Ø		
	Ν	Ν	{7}	{9}	Ø	Ø	{8}	Ø	Ø		
	Ν	Ν	{1}	Ø	Ø	Ø	Ø	{2,6}	Ø		

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

