LEX9: Implementing DFAs

Lexical Analysis

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

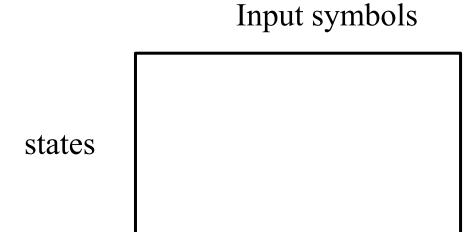
- Token ⇒ Pattern
- Pattern ⇒ Regular Expression
- Regular Expression \Rightarrow NFA
- NFA ⇒ DFA
- DFA ⇒ Table-driven implementation of DFA

- Token ⇒ Pattern
- Pattern ⇒ Regular Expression
- Regular Expression \Rightarrow NFA
- NFA \Rightarrow DFA
- → DFA ⇒ Table-driven implementation of DFA

- Token ⇒ Pattern
- Pattern ⇒ Regular Expression
- Regular Expression \Rightarrow NFA
- NFA ⇒ DFA
 Implement NFAs
- DFA ⇒ Table-driven implementation of DFA

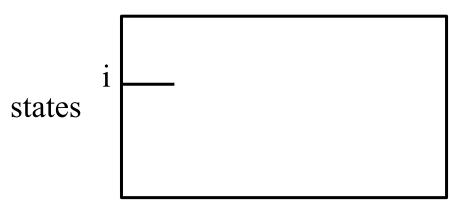
- Token ⇒ Pattern
- Pattern ⇒ Regular Expression
- Regular Expression \Rightarrow NFA
- NFA ⇒ DFA
 Implement NFAs
 Convert regove to the c
 - **Convert regexp to DFA**
- → DFA ⇒ Table-driven implementation of DFA

- 2D array storing the transition table
 - One dimension is states
 - Other dimension is input symbols
 - For every transition $S_i \rightarrow^a S_k$ define T[i,a]=k

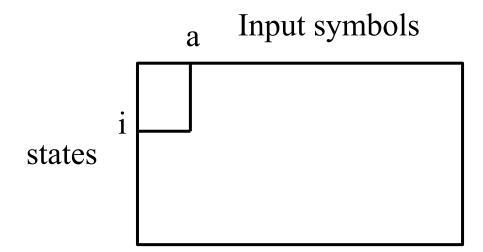


- 2D array storing the transition table
 - One dimension is states
 - Other dimension is input symbols
 - For every transition $S_i \rightarrow^a S_k$ define T[i,a]=k

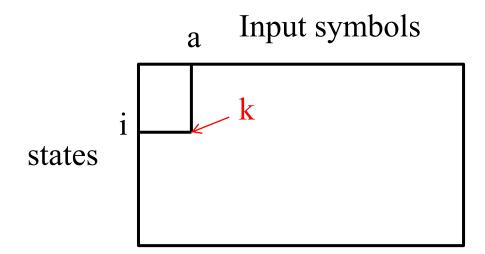
Input symbols

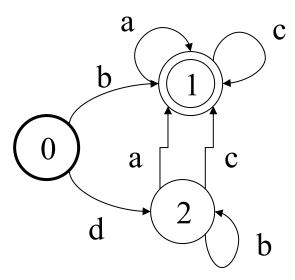


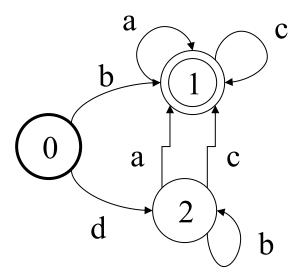
- 2D array storing the transition table
 - One dimension is states
 - Other dimension is input symbols
 - For every transition $S_i \rightarrow^a S_k$ define T[i,a]=k



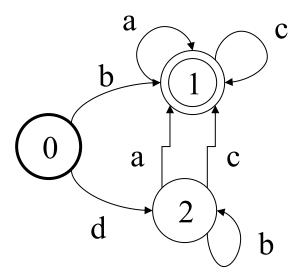
- 2D array storing the transition table
 - One dimension is states
 - Other dimension is input symbols
 - For every transition $S_i \rightarrow^a S_k$ define T[i,a]=k



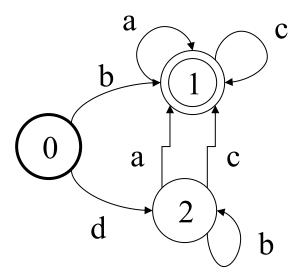




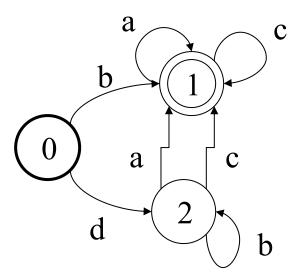
	a	b	c	d
0				
1				
2				



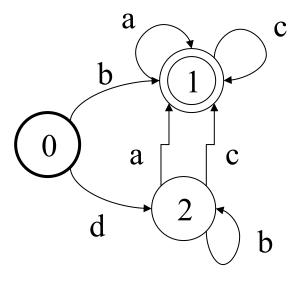
	a	b	c	d
0	-	1	ı	2
1				
2				



	a	b	c	d
0	-	1	ı	2
1	1	-	1	I
2				

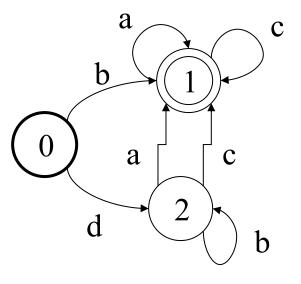


	a	b	c	d
0	ı	1	ı	2
1	1	-	1	I
2	1	2	1	-



	a	b	c	d
0	ı	1	ı	2
1	1	ı	1	ı
2	1	2	1	ı

```
i = 0
state = 0
while (input[i]) {
    state = nextState(state, input[i])
    i = i + 1
}
```



	a	b	c	d
0	ı	1	ı	2
1	1	ı	1	1
2	1	2	1	I

```
i = 0
state = 0
while (input[i]) {
    state = nextState(state, input[i])
    i = i + 1
```

```
nextState(state, x) {
    return A[state][x]
}
```

- 2D array storing the transition table
 - Too many states and duplicates

- 2D array storing the transition table
 - Too many states and duplicates
- Adjacency list
 - more space efficient but slower

- 2D array storing the transition table
 - Too many states and duplicates
- Adjacency list
 - more space efficient but slower
- Merge two ideas: array structures used for sparse tables like DFA transition tables

	a	b	c	d
0	-	1	-	2
1	1	-	1	1
2	1	2	1	-

	a	b	c	d
0	-	1	-	2
1	1	-	1	_
2	1	2	1	_

		-	1	-	2		
				1	-	1	ı
1	2	1	ı				
1	2	1	1	1	2	1	-
0	1	2	3	4	5	6	7

	a	b	c	d
0	-	1	-	2
1	1	-	1	•
2	1	2	1	

		-	1	_	2		
				1	ı	1	-
1	2	1	ı				
1	2	1	1	1	2	1	
0	1	2	3	4	5	6	7

next

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	ı

		-	1	-	2		
				1	_	1	-
1	2	1	_				
1	2	1	1	1	2	1	_
0	1	2	3	4	5	6	7

next

0	2
1	4
2	0

	a	b	c	d
0	-	1	-	2
1	1	ı	1	ı
2	1	2	1	ı

		-	1	_	2			
				1	_	1	_	
1	2	1	_					
1	2	1	1	1	2	1	_	n
0	1	2	3	4	5	6	7	

next

0	2
1	4
2	0

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	ı

		-	1	-	2			
				1	_	1	-	
1	2	1	-					
1	2	1	1	1	2	1	-	next
0	1	2	3	4	5	6	7	

base

0	2
1	4
2	0

nextState(2,a)=

	a	b	c	d
0	-	1	-	2
1	1	-	1	_
2	1	2	1	-

		_	1	-	2			
				1	_	1	-	
1	2	1	-					
1	2	1	1	1	2	1	1	ne
0	1	2	3	4	5	6	7	

next

$$nextState(2,a) = next[0+0]$$

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	-

		-	1	-	2			
				1	_	1	_	
1	2	1	-					
1	2	1	1	1	2	1	_	n
0	1	2	3	4	5	6	7	

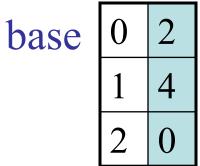
next

base

nextState(2,a) = next[0+0]

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	ı

		_	1	-	2			
				1	_	1	_	
1	2	1	-					
1	2	1	1	1	2	1	-	next
0	1	2	3	4	5	6	7	

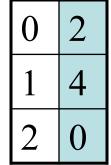


nextState(2,a)= next
$$[0+0]$$

nextState(1,c)=

	a	b	c	d
0	ı	1	-	2
1	1	-	1	-
2	1	2	1	-

		-	1	-	2			
				1	_	1	ı	
1	2	1	-					
1	2	1	1	1	2	1	-	next
0	1	2	3	4	5	6	7	

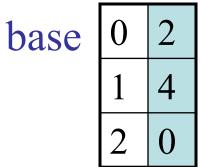


$$nextState(2,a) = next[0+0]$$

$$nextState(1,c) = next[4+2]$$

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	ı

		-	1	_	2			
				1	_	1	-	
1	2	1	-					
1	2	1	1	1	2	1	-	nex
0	1	2	3	4	5	6	7	



$$nextState(2,a) = next[0+0]$$

$$nextState(1,c) = next[4+2]$$

	a	b	c	d
0	-	1	-	2
1	1	•	1	ı
2	1	2	1	-

		_	1	-	2			
				1	_	1	_	
1	2	1	-					
1	2	1	1	1	2	1	_	next
0	1	2	3	4	5	6	7	

0	2
1	4
2	0

	a	b	c	d
0	-	1	-	2
1	1	•	1	ı
2	1	2	1	-

		-	1	-	2			
				1	_	1	-	
1	2	1	-					
1	2	1	1	1	2	1	_	
0	1	2	3	4	5	6	7	

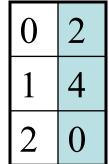
next

0	2
1	4
2	0

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	-

		-	1	-	2			
				1	_	1	ı	
1	2	1	-					
1	2	1	1	1	2	1	-	ne
0	1	2	3	4	5	6	7	

next



$$nextState(2,a) = next[0+0]$$

$$nextState(1,c) = next[4+2]$$

$$nextState(0,c) = next[2+2]$$

	a	b	c	d
0	ı	1	-	2
1	1	-	1	ı
2	1	2	1	ı

		-	1	_	2			
				1	-	1	ı	
1	2	1	_					
1	2	1	1	1	2	1	ı	next
0	1	2	3	4	5	6	7	
2	2	2	0	1	0	1	-	chec
~				F.0	0.7			-

base

0	2
1	4
2	0

$$nextState(2,a) = next[0+0]$$

$$nextState(1,c) = next[4+2]$$

$$nextState(0,c) = next[2+2]$$

check

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	•

		_	1	-	2		
				1	-	1	-
1	2	1	-				
1	2	1	1	1	2	1	-
0	1	2	3	4	5	6	7
2	2	2	0	1	0	1	_

base 0 2 1 4 2 0

$$nextState(s, x)$$
:

$$L := base[s] + x$$

return next[L] if check[L] ==
$$s$$

next

check

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	-

	a	b	c	d
0	ı	1	-	2
1	1	-	1	•
2	1	2	1	•

	-	1	-	2		
			1	-	1	-
-	2	-	-			
-	2	1	1	2	1	_
0	1	2	3	4	5	6

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	•

	-	1	-	2		
			1	-	1	-
-	2	-	-			
_	2	1	1	2	1	-
0	1	2	3	4	5	6

next

base

0	1
1	3
2	0

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	

	-	1	-	2		
			1	_	1	-
-	2	_	-			
-	2	1	1	2	1	-
0	1	2	3	4	5	6

$$nextState(s, x)$$
:
 $L := base[s] + x$
 $return next[L]$

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	•

	-	1	-	2		
			1	-	1	-
-	2	-	_			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	_

base

nextState(s, x):

$$L := base[s] + x$$

return next[L]

	a	b	c	d
0	ı	1	ı	2
1	1	-	1	-
2	1	2	1	-

	_	1	_	2		
			1	-	1	-
-	2	_	-			
-	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	_
	~					

next

base

nextState(s, x):

$$L := base[s] + x$$

return next[L] if check[L]
$$== s$$

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	1

	_	1	_	2		
			1	-	1	-
-	2	_	-			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	-
	~					

base

nextState(s, x):

$$L := base[s] + x$$

return next[L] if check[L] == s

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	1

	-	1	-	2		
			1		1	_
-	2	-	_			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_
	!	<u> </u>	ļ	l		

next

base

$$nextState(s, x)$$
:

$$L := base[s] + x$$

return next[L] if check[L] == s

default

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	-

	_	1	-	2		
			1	-	1	-
-	2	-	_			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_

next

base

nextState(s, x): L := base[s] + x

return next[L] if check[L] == s**else return** *nextState*(default[s], x)

default

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	-

	-	1	-	2		
			1	_	1	-
ı	2	_	-			
-	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_

base 0 1 -1 3 -2 0 1

nextState(s, x):

L := base[s] + x

return next[L] if check[L] == s
else return nextState(default[s], x)

default

next

check

nextState(2,b)=

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	_

	-	1	-	2		
			1	ı	1	ı
ı	2	-	-			
I	2	1	1	2	1	_
0	\bigcirc	2	3	4	5	6
_	2	0	1	0	1	_

base 0 1 -1 3 -2 0 1

nextState(s, x):

L := base[s] + x

return next[L] if check[L] == s
else return nextState(default[s], x)

default

next

check

nextState(2,b)=

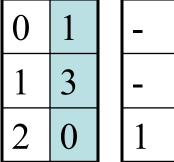
	a	b	c	d
0	ı	1	-	2
1	1	_	1	-
2	1	2	1	_

	-	1	-	2		
			1	-	1	ı
_	2	-	-			
_	2	1	1	2	1	_
0	1	2	3	4	5	6
_	(2)	0	1	0	1	_

next

check

base



nextState(s, x):

$$nextState(2,b)=$$

L := base[s] + x $default \qquad return next[L]$

return next[L] if check[L] == s
else return nextState(default[s], x)

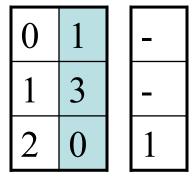
6/4/2016

	a	b	c	d
0	-	1	_	2
1	1	-	1	_
2	1	2	1	_

		1	-	2		
			1	_	1	-
-	2	_	-			
-	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	_
					noxt	C404

next

base



nextState(s, x):

$$nextState(2,b)=next[1]$$

default

L := base[s] + xreturn next[L] if check[L] == s

else return nextState(default[s], x)

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	1

	-	1	-	2		
			1		1	-
-	2	_	-			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_

base 0 1 2 0 2 0

default

nextState(s, x) : nextState(2,a) =

L := base[s] + x

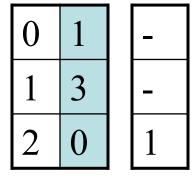
return next[L] if check[L] == s
else return nextState(default[s], x)

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	-

	_	1	-	2		
			1	_	1	-
-	2	-	1			
-	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	_

next

base



default

nextState(s, x) : nextState(2,a) =

L := base[s] + x

return next[L] if check[L] == s
else return nextState(default[s], x)

	a	b	c	d
0	ı	1	-	2
1	1	_	1	-
2	1	2	1	_

	_	1	-	2		
			1	-	1	-
-	2	-	1			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
_	2	0	1	0	1	_
_	2	0	1	0	1	_

base

L := base[s] + x

nextState(s, x):

return next[L] if check[L] == s**else return** *nextState*(default[s], x)

default

next

nextState(2,a)=

	a	b	c	d
0	-	1	-	2
1	1	-	1	-
2	1	2	1	-

	-	1	-	2		
			1	-	1	-
-	2	-	-			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	-

next

check

base

nextState(2,a)= nextState(s, x): nextState(1,a) L := base[s] + x

return next[L] if check[L] == s**else return** nextState(default[s], x)

default

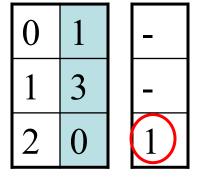
6/4/2016

	a	b	c	d
0	-	1	-	2
1	1	-	1	ı
2	1	2	1	

	-	1	-	2		
			1	-	1	-
-	2	_	-			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_
	-					

next

base



default

$$nextState(s, x) : nextState(2,a) =$$

 $L := base[s] + x$ $nextState(1,a)$

return next[L] if check[L] == s
else return nextState(default[s], x)

	a	b	c	d
0	ı	1	-	2
1	1	-	1	-
2	1	2	1	-

	-	1	-	2		
			1	-	1	-
_	2	_	_			
_	2	1	1	2	1	-
0	1	2	3	4	5	6
-	2	0	1	0	1	_

next

nextState(2,a)=

```
nextState(s, x):
                   nextState(1,a) = next[3]
 L := base[s] + x
 return next[L] if check[L] == s
 else return nextState(default[s], x)
```