CMPT 379
Compilers

Anoop Sarkar

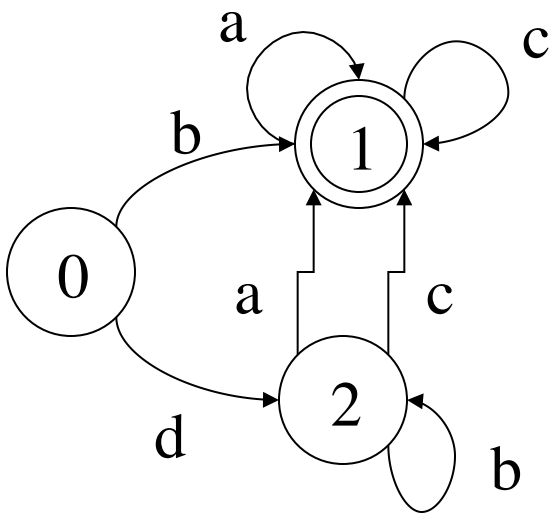
http://www.cs.sfu.ca/~anoop

Efficient data-structures for DFAs

Implementing DFAs

- 2D array storing the transition table
- Adjacency list, more space efficient but slower
- Merge two ideas: array structures used for sparse tables like DFA transition tables
  - base & next arrays: Tarjan and Yao, 1979
  - Dragon book (default+base & next+check)

Implementing DFAs



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

Implementing DFAs

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

base

0	2
1	4
2	0

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

next

2	2	2	0	1	0	1	-

check

nextstate(*s*, *x*) :

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

return next[L] **if** check[L] **eq** *s*

Implementing DFAs

	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

next

-	2	0	1	0	1	-

check

base

0	1
1	3
2	0

-
-
1

default

nextstate(s, x) :

L := base[s] + x

return next[L] **if** check[L] **eq** s

else return *nextstate*(default[s], x)