



Regular Expression to DFA (Direct Method)

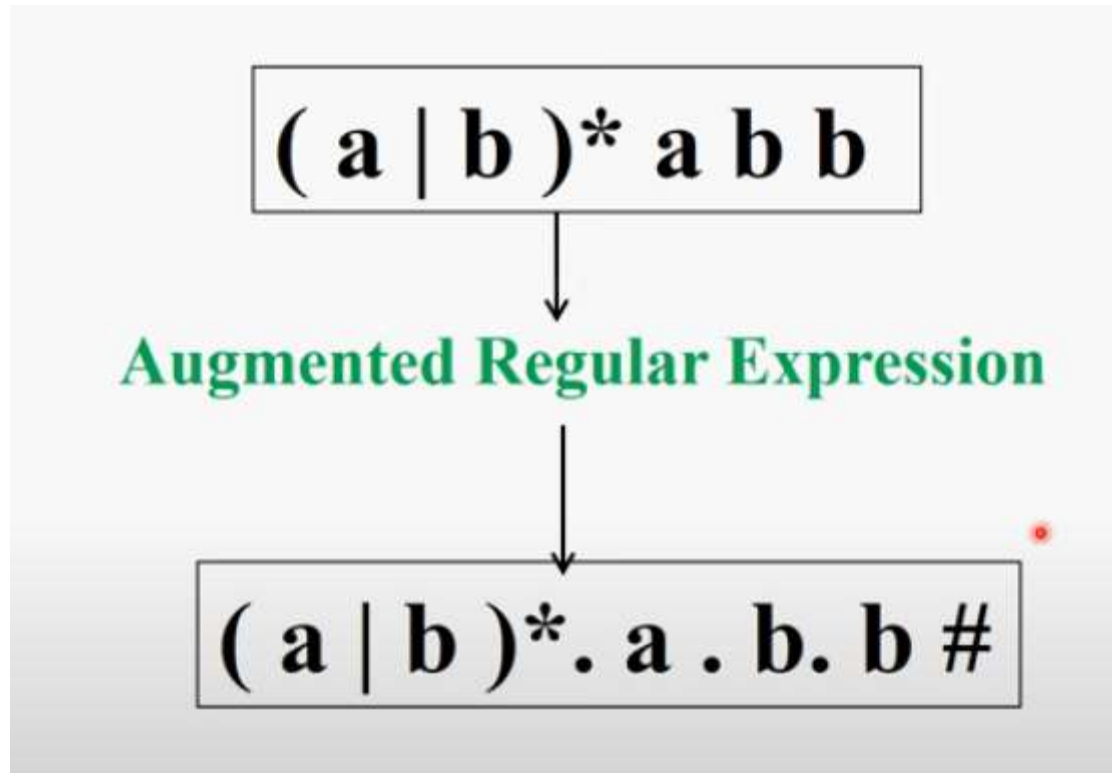
- Regular Expression to Augmented Regular Expression
- Augmented RE to syntax tree
- Nullable, First pos, Last pos, Followpos
 - Nullable \rightarrow^* and Empty
 - Followpos \rightarrow^* (closure) and Concatenation (.)
- Construct DFA



Regular Expression to DFA (Direct Method)

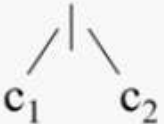
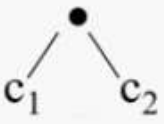



Example 1: $(a|b)^*abb$





Rules for Computing the Function

Node n	$nullable(n)$	$firstpos(n)$	$lastpos(n)$
A leaf labeled by ϵ	true	\emptyset	\emptyset
A leaf with position i	false	$\{i\}$	$\{i\}$
	$nullable(c_1)$ or $nullable(c_2)$	$firstpos(c_1) \cup firstpos(c_2)$	$lastpos(c_1) \cup lastpos(c_2)$
	$nullable(c_1)$ and $nullable(c_2)$	if ($nullable(c_1)$) $firstpos(c_1) \cup firstpos(c_2)$ else $firstpos(c_1)$	if ($nullable(c_2)$) $lastpos(c_1) \cup lastpos(c_2)$ else $lastpos(c_2)$
	<i>true</i>	$firstpos(c_1)$	$lastpos(c_1)$

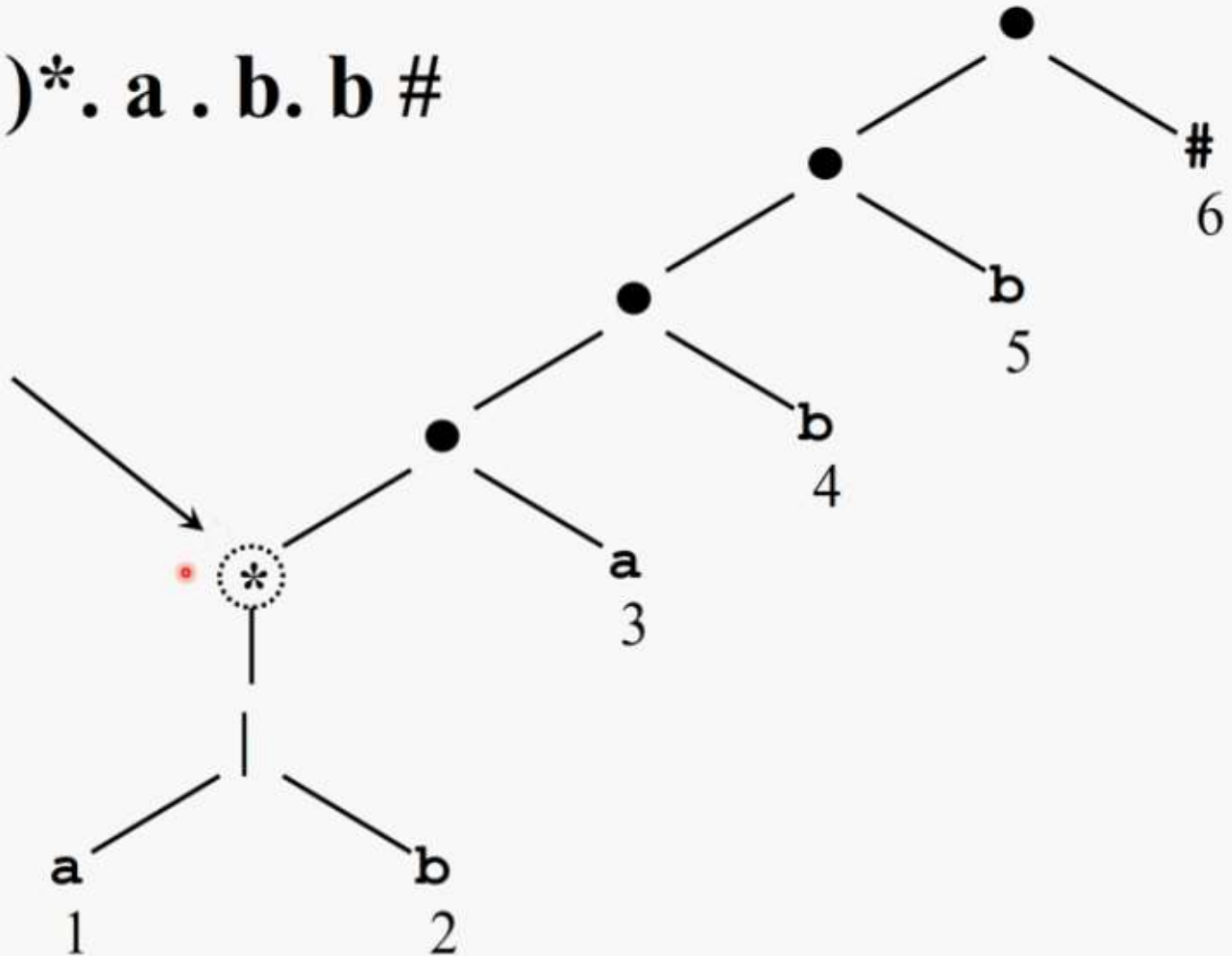


Syntax Tree



$(a | b)^* . a . b . b \#$

nullable



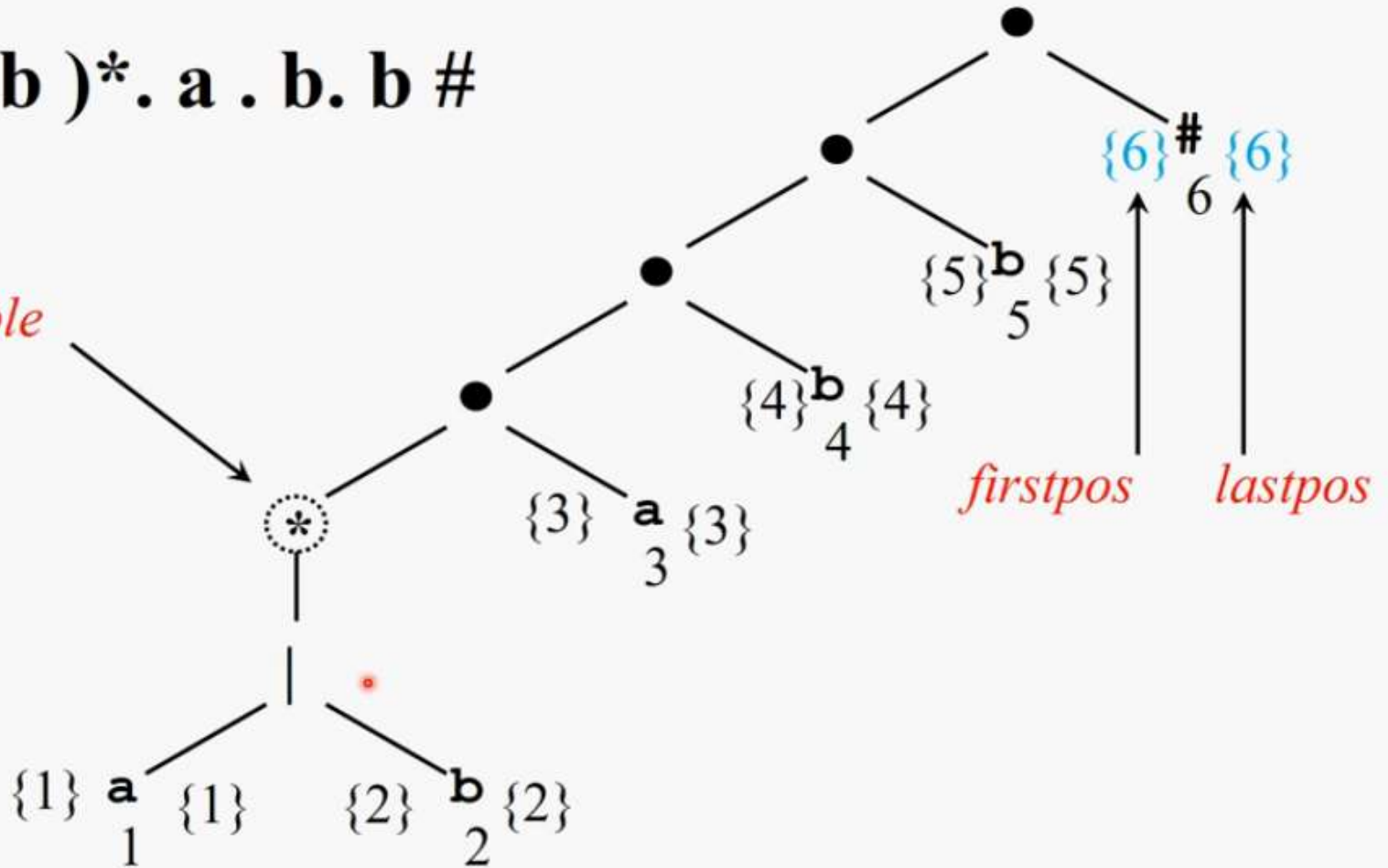


Firstpos and Lastpos of operands



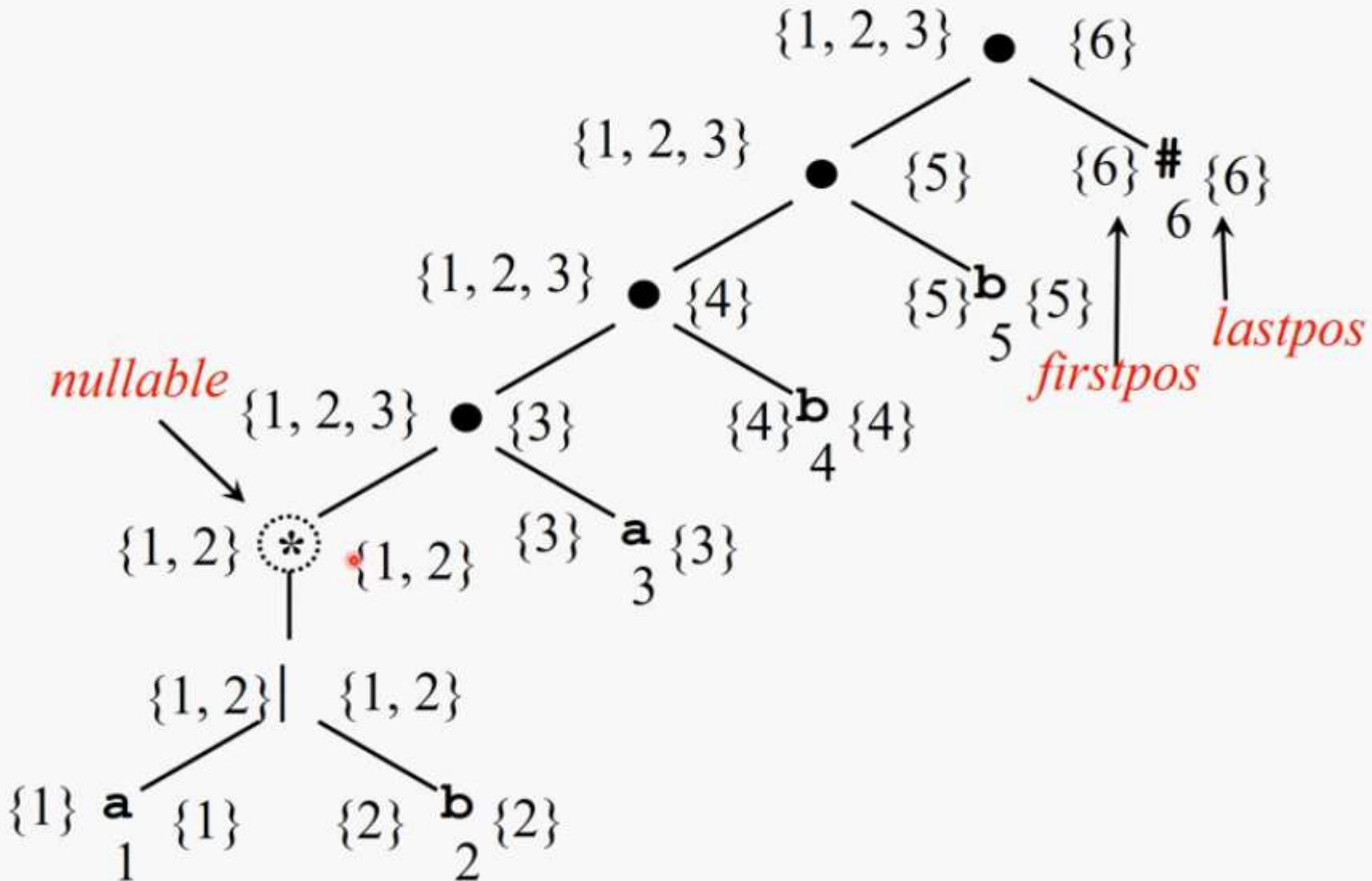
$(a | b)^* . a . b . b \#$

nullable





Firstpos and Lastpos of Operators



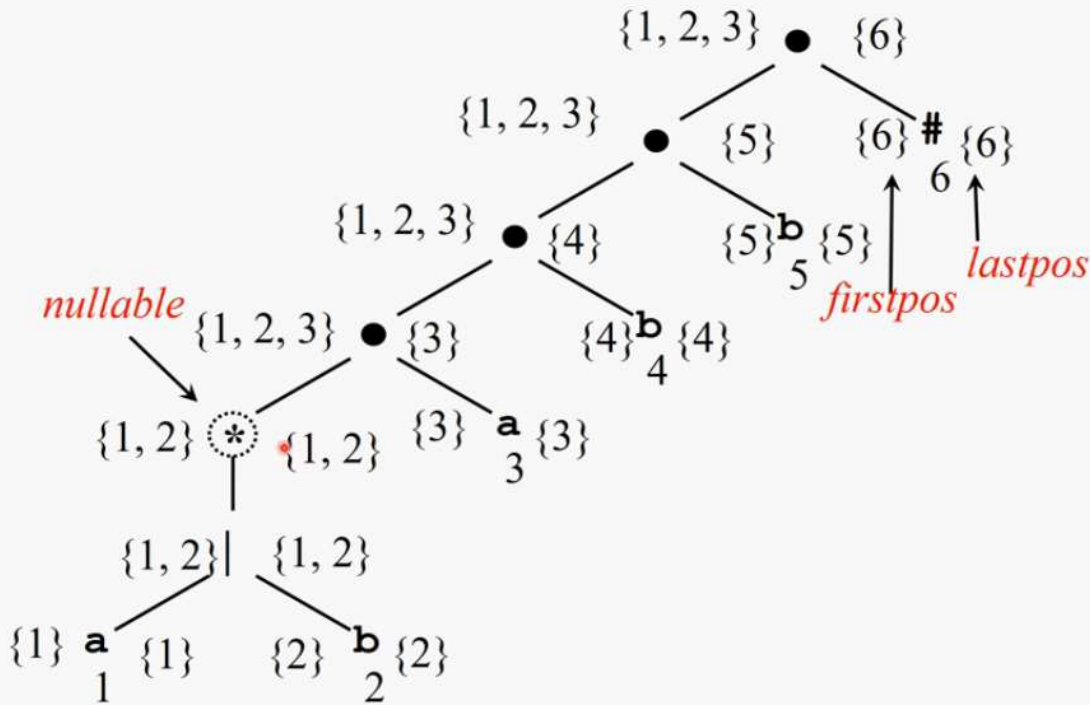


Followpos

$(a | b)^* . a . b . b \#$



1 2 3 4 5 6

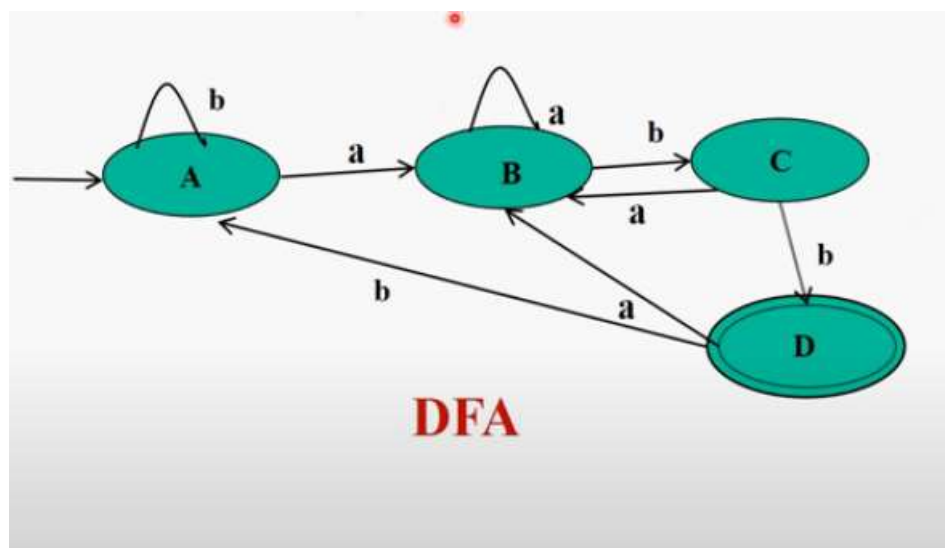
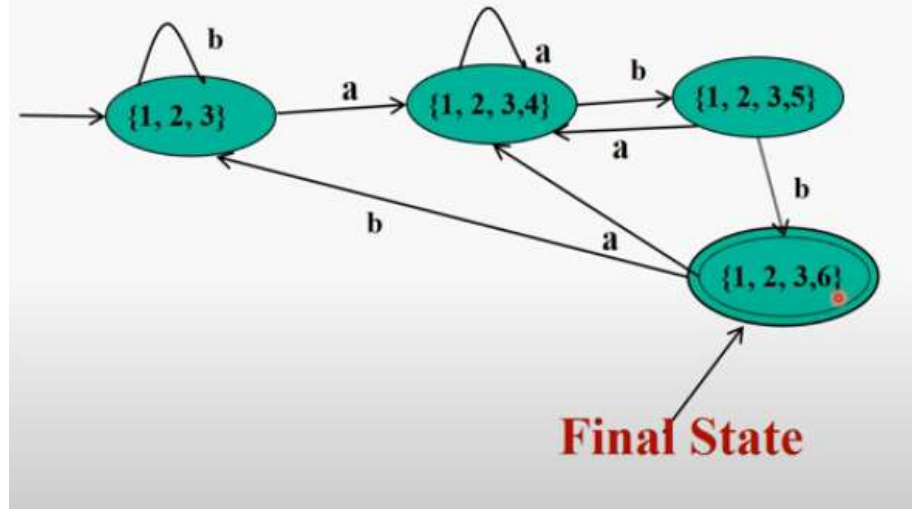


		Followpos
a	1	1,2,3
b	2	1,2,3
a	3	4
b	4	5
b	5	6
#	6	-



DFA Construction

	Node	<i>followpos</i>
a	1	1,2,3
b	2	1,2,3
a	3	4
b	4	5
b	5	6
#	6	-

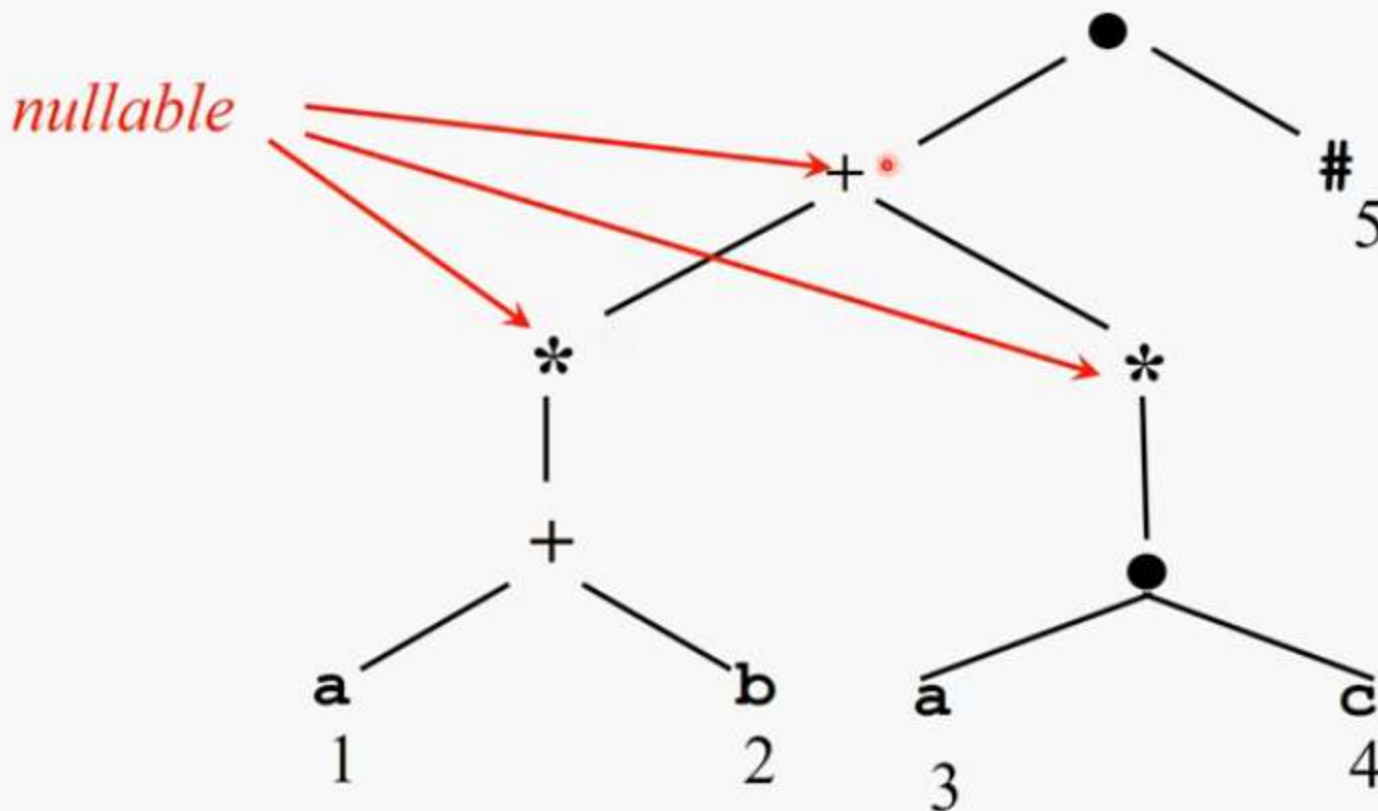




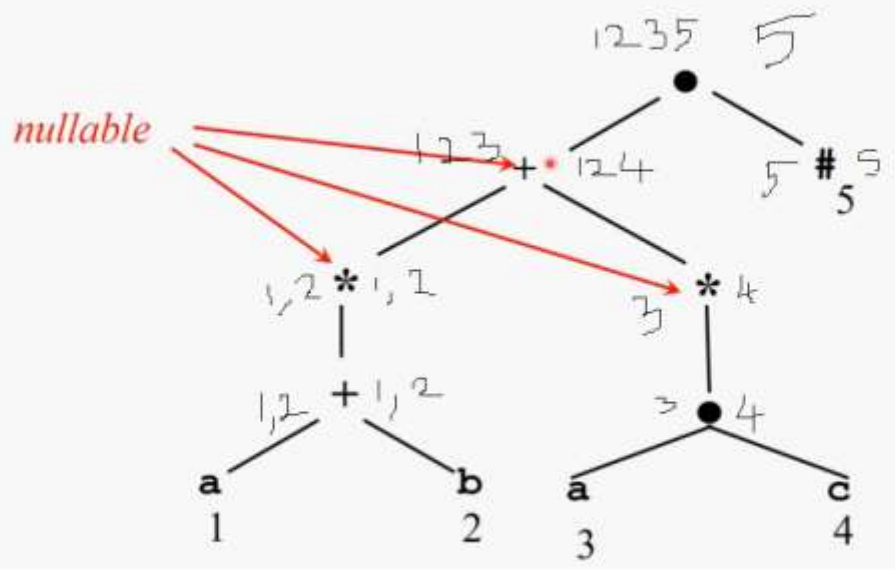
Assignment - Example 2



$((a + b)^* + (a \cdot c)^*) \cdot \#$

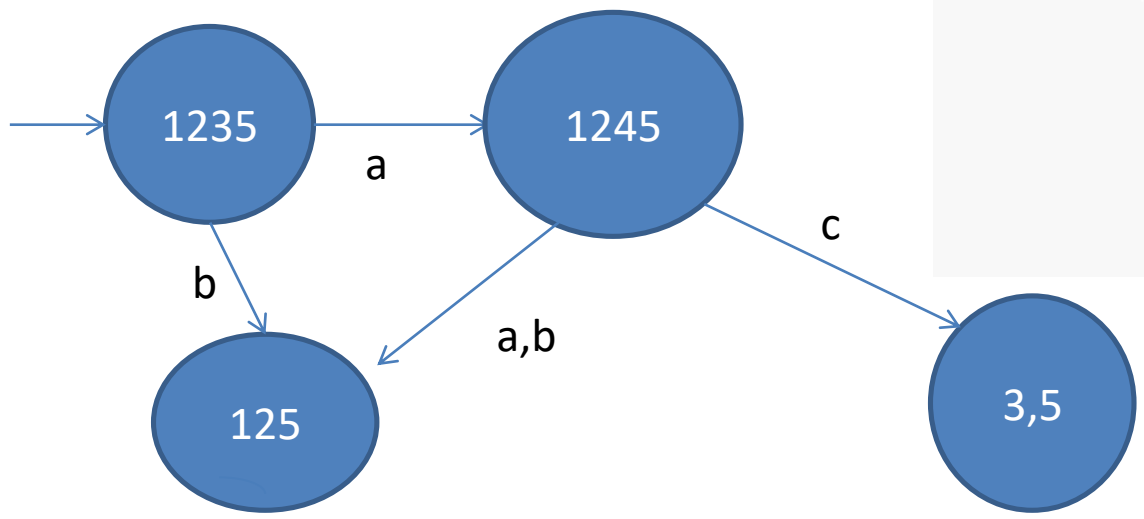
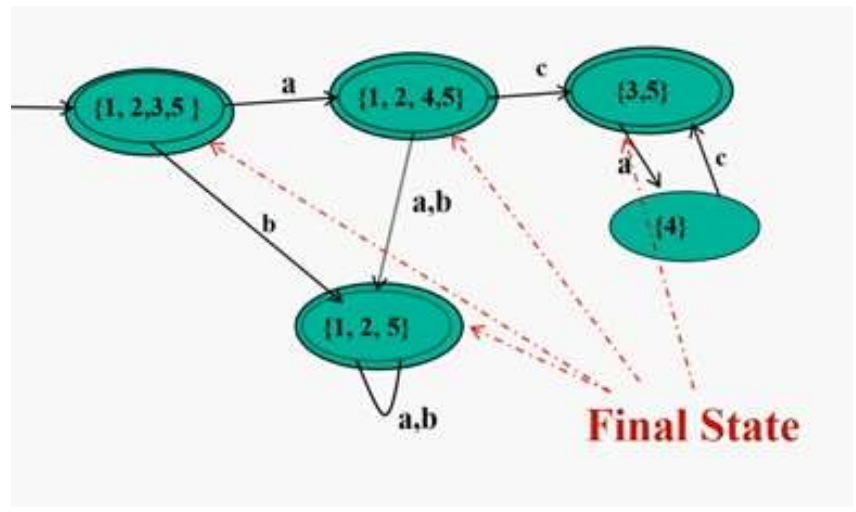


$((a + b)^* + (a \cdot c)^*) \cdot \#$



		FOLLOWPO S (*,.)
--	--	---------------------

a	1	1,2,5
b	2	1,2,5
a	3	4
c	4	3,5
#	5	-





- **Topics covered**

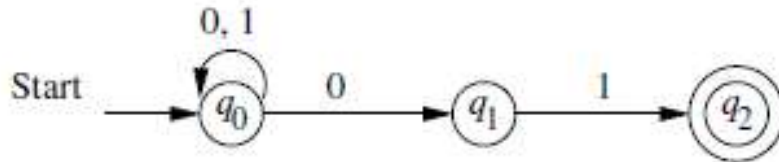
1. Regular expression to Automata
2. Regular expression to ϵ -NFA (Thompsons construction)
3. ϵ -NFA to DFA
4. Regular Expression to DFA
 1. Direct conversion of Regular Expression to DFA
 2. NFA to DFA (Subset Construction Method) – UNIT I



Equivalence of NFA & DFA/ *NFA to DFA*



- NFA which accepts all the strings ending with 01**

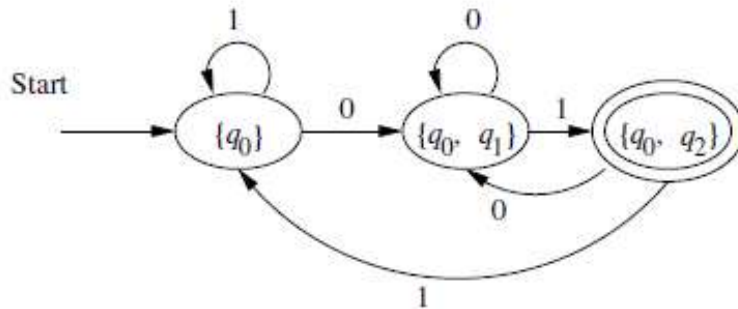


	0	1
q ₀	q ₀ , q ₁	q ₀
q ₁	-	q ₂
q ₂	-	-

- States $\rightarrow q_0, q_1, q_2 \rightarrow 3$ states
 $\rightarrow \{\text{null}, q_0, q_1, q_2, q_0q_1, q_0q_2, q_2q_3, q_0q_1q_2\}$

DFA Construction

- $Q' = \text{NULL}$
- $Q' = \{q_0, \{q_0, q_1\}, \{q_0, q_2\}\}$



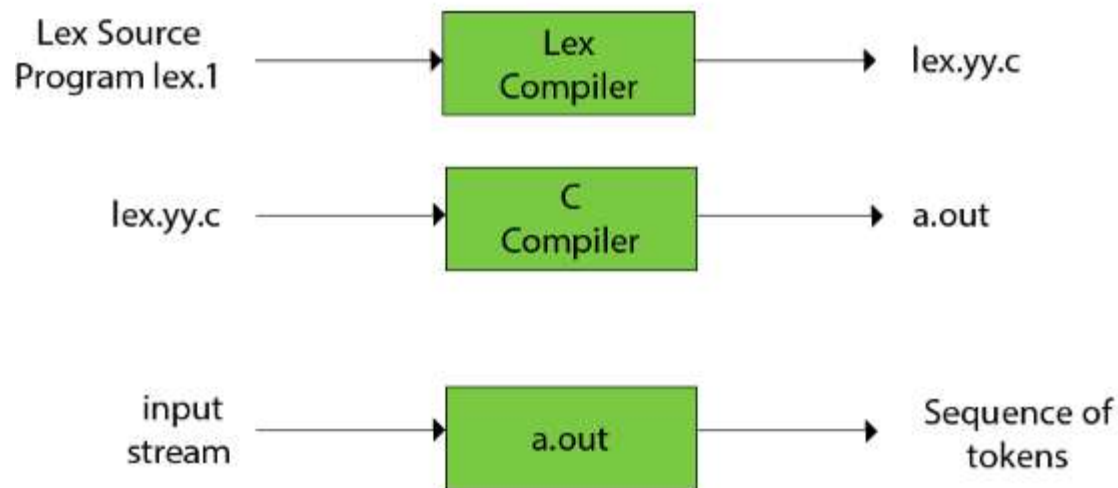
	0	1
(initial) q ₀	{q ₀ , q ₁ }	q ₀
{q ₀ , q ₁ }	{q ₀ , q ₁ }	{q ₀ , q ₂ }
(Final) {q ₀ , q ₂ }	{q ₀ , q ₁ }	q ₀



Tool for generating Lexical Analyzer



- LEX
 - Program – lexical analyzer. YACC parser generator



- Flex (Fast Lexical Analyzer Generator)
- <https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/>



Summary (UNIT II)

- Phases of compiler
- Lexical Analysis / Tokenizer / scanner
- Lexemes → Tokens
- (Type 3) – Regular Grammar - FSA
 - DFA
 - NFA
 - Epsilon NFA
 - Conversion
 - RE to NFA
 - RE to DFA
 - Direct Method
 - Subset Construction