Lecture 6 - Lexical Analyzer (Last)

Adnan Ferdous Ashrafi

Stamford University Bangladesh

DFA from REGEX

To begin with how to go directly from a regular expression to a DFA, we must first dissect the NFA construction and consider the roles played by various states.

The important states of the NFA correspond directly to the positions in the regular expression that hold symbols of the alphabet. It is useful, as we shall see, to present the regular expression by its syntax tree, where the leaves correspond to operands and the interior nodes correspond to operators. An interior node is called a cat-node, or-node, or star-node if it is labeled by the concatenation operator $\bullet$, union operator $|$, or star operator $*$, respectively.

Syntax Tree

Leaves in a syntax tree are labeled by ε or by an alphabet symbol. To each leaf not labeled ε, we attach a unique integer. We refer to this integer as the position of the leaf and also as a position of its symbol. Note that a symbol can have several positions. The positions in the syntax tree correspond to the important states of the constructed NFA.

Construction of Syntax Tree

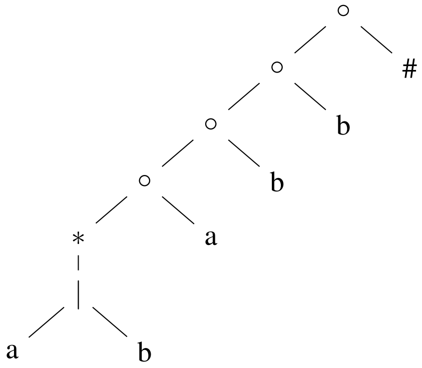
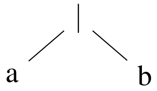


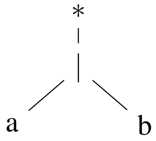
Figure 1: Syntax Tree for $(a|b)*abb\#$

Breakdown - Construction of Syntax Tree



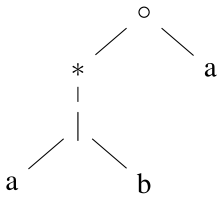
Syntax Tree for $\boxed{(a|b)} * abb\#$

Breakdown - Construction of Syntax Tree



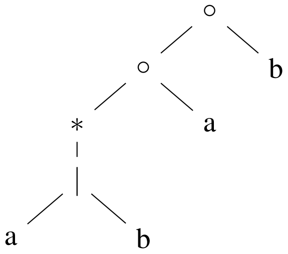
Syntax Tree for $\boxed{(a|b)*}abb\#$

Breakdown - Construction of Syntax Tree



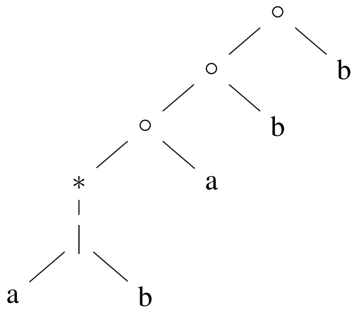
Syntax Tree for $\boxed{(a|b)*a}bb\#$

Breakdown - Construction of Syntax Tree



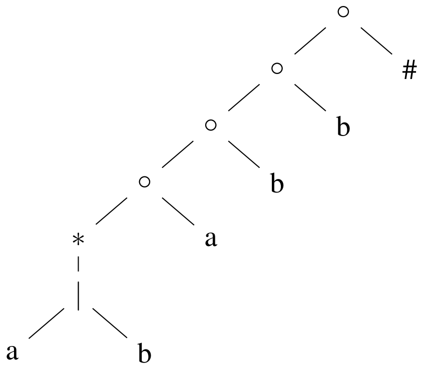
Syntax Tree for $\boxed{(a|b)*ab}b\#$

Breakdown - Construction of Syntax Tree



Syntax Tree for $\boxed{(a|b)*abb}$#

Breakdown - Construction of Syntax Tree



Syntax Tree for $\boxed{(a|b)*abb\#}$

Functions computed from a Syntax Tree

To construct a DFA directly from a regular expression, we construct its syntax tree and then compute four functions:

1. *nullable*
2. *firstpos*
3. *lastpos*
4. *followpos*

which are defined as follows. Each definition refers to the syntax tree for a particular augmented regular expression $(r)\#$.

Functions computed from a Syntax Tree

$nullable(n)$

nullable(n) is true for a syntax-tree node n if and only if the subexpression represented by n has E in its language. That is, the subexpression can be "made null" or the empty string, even though there may be other strings it can represent as well.

$firstpos(n)$

firstpos(n) is the set of positions in the subtree rooted at n that correspond to the first symbol of at least one string in the language of the subexpression rooted at n.

Functions computed from a Syntax Tree

$lastpos(n)$

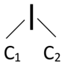
lastpos(n) is the set of positions in the subtree rooted at n that correspond to the last symbol of at least one string in the language of the subexpression rooted at n.

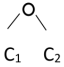
$followpos(n)$

followpos(p), for a position p, is the set of positions q in the entire syntax tree such that there is some string $x = a_1a_2 \dots a_n$, in $L((r)\#)$ such that for some i, there is a way to explain the membership of x in $L((r)\#)$ by matching a_i to position p of the syntax tree and a_{i+l} to position q.

Rules for calculation of functions

node n	***nullable(n)***	***firstpos(n)***	***lastpos(n)***
leaf ε	true	Ø	Ø
leaf i	false	{i}	{i}

	*nullable(C₁)*
	or
	nullable(C₂)
		firstpos(C₁)
		U
		firstpos(C₂)
			lastpos(C₁)
			U
			lastpos(C₂)

	*nullable(C₁)*
	and
	nullable(C₂)
		if *nullable(C₁)*
		then *firstpos(C₁)*
		U
		firstpos(C₂)
		else *firstpos(C₁)*
			if *nullable(C₂)*
			then *lastpos(C₁)*
			U
			lastpos(C₂)
			else *lastpos(C₂)*

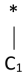
	true	*firstpos(C₁)*	*lastpos(C₁)*

Figure 2: Rules for computing nullable, firstpos, lastpos

Rules for calculating followpos(n)

There are only two ways that a position of a regular expression can be made to follow another.

1. If n is a cat-node with left child C_1 and right child C_2, then for every position i in $lastpos(C_1)$, all positions in $firstpos(C_2)$ are in $followpos(i)$.
2. If n is a star-node, and i is a position in $lastpos(n)$, then all positions in $firstpos(n)$ are in $followpos(i)$.

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 1 - Augment Regular Expression

Augmented regular expression after appending # infront of the regular expression:

$$(a|b)*abb\#$$

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 2 - Create Syntax Tree

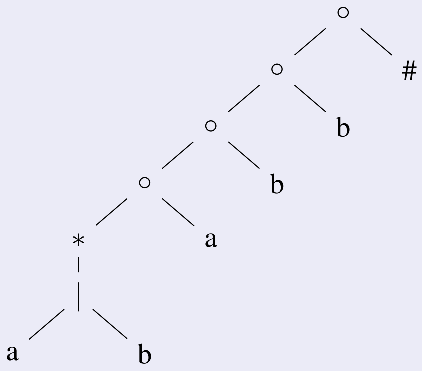


Figure 3: Syntax Tree for $(a|b)*abb\#$

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 3 - Calculate $\mathit{nullable}(n), \mathit{firstpos}(n), \mathit{lastpos}(n)$

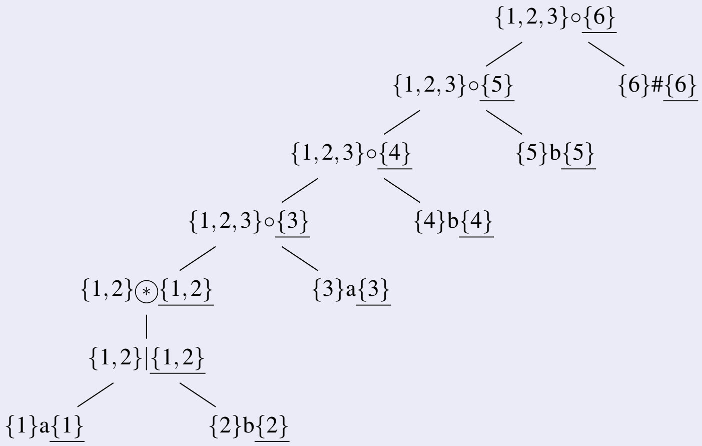


Figure 4: Syntax Tree for $(a|b)*abb\#$

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 4 - Calculate $followpos(n)$

Node n	**followpos(n)**
1	$\{1,2,3\}$
2	$\{1,2,3\}$
3	$\{4\}$
4	$\{5\}$
5	$\{6\}$
6	$\emptyset$

Table 1: Function $followpos(n)$ for $(a|b)*abb\#$

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 5 - Construct DFA transition table for $(a|b)*abb\#$

Symbol / D_{state}	**a**	**b**
$\{1,2,3\}$	$\{1,2,3,4\}$	$\{1,2,3\}$
$\{1,2,3,4\}$	$\{1,2,3,4\}$	$\{1,2,3,5\}$
$\{1,2,3,5\}$	$\{1,2,3,4\}$	$\{1,2,3,6\}$
$\{1,2,3,6\}$	$\{1,2,3,4\}$	$\{1,2,3\}$

Table 2: DFA transition table for $(a|b)*abb$

REGEX to DFA

- Construct a DFA from the regular expression $(a|b)*abb$

Step 6 - Construct DFA transition diagram for $(a|b)*abb$

$$\begin{tikzcd}
\text{start} \arrow[r] & \{1,2,3\} \arrow[loop above, "b"] \arrow[r, "a"] & \{1,2,3,4\} \arrow[loop above, "a"] \arrow[r, "b"] & \{1,2,3,5\} \arrow[r, "b"] \arrow[l, bend left, "a"] & \{1,2,3,6\} \arrow[ll, bend right, "a"'] \arrow[lll, bend left, "b"]
\end{tikzcd}$$

Figure 5: DFA Transition Diagram for $(a|b)*abb$

Complete DFA for $(a|b)*abb$

Symbol / D_{state}	**a**	**b**
$\{1,2,3\}$	$\{1,2,3,4\}$	$\{1,2,3\}$
$\{1,2,3,4\}$	$\{1,2,3,4\}$	$\{1,2,3,5\}$
$\{1,2,3,5\}$	$\{1,2,3,4\}$	$\{1,2,3,6\}$
$\{1,2,3,6\}$	$\{1,2,3,4\}$	$\{1,2,3\}$

Table 3: DFA transition table for $(a|b)*abb$

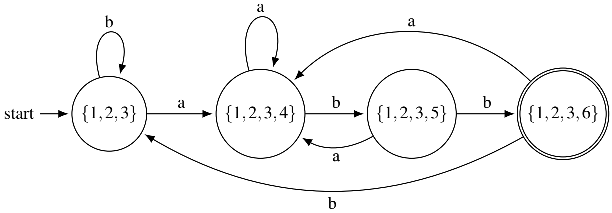


Figure 6: DFA Transition Diagram for $(a|b)*abb$

Do it yourself

Exercise

Find the Deterministic Finite Automata (NFA) for the following regular expressions:

1. $((abc)*|a)c+b$
2. $((ab)+(bc)*)*cba$
3. $((a*b)|c+)ab*c+$

Thank you.
Any Questions?