Jan 5

Lexical Analysis

→ divide input into tokens
→ identifier

int 97x3; → not valid, does not need to be parsed

input file (regex) → Lexer Gen → Lexer.java

Language is a set of strings

L('a') is {"a"}

L('a*') is {"a", "aa", "aaa", ... }

Regex

"a"	{"a"}
ε	{ }
s|t	$L(s) \cup L(t)$
st	$\{uw \mid u \in L(s), w \in L(t)\}$
s^*	$\{""\} \cup \{uw \mid u \in L(S), w \in L(S^*)\}$

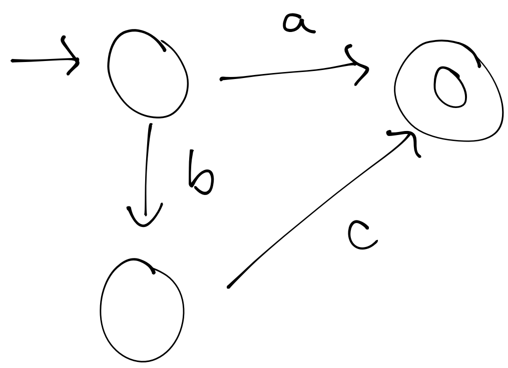
(0|1|2|3) = [0-3]

Identifier in the unamed language

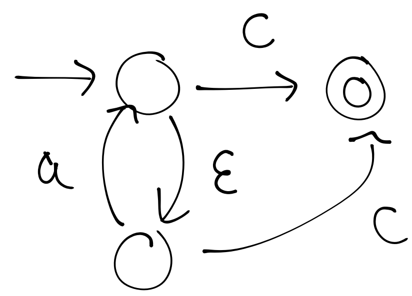
([a-z]|[A-Z]|'_')([0-9]|[a-z]|[A-Z]|'_')*

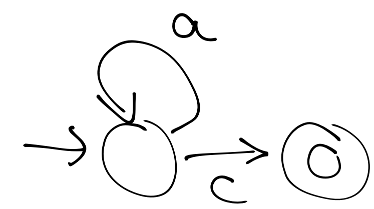
Jan 9 NFA

a | bc

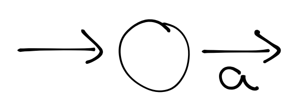
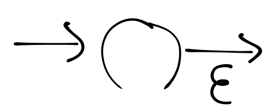
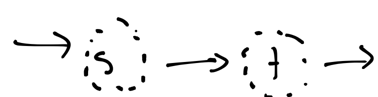
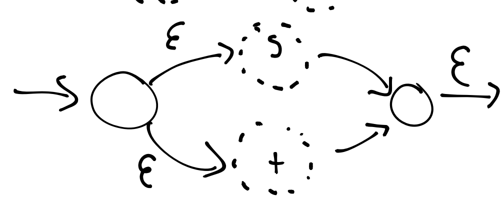
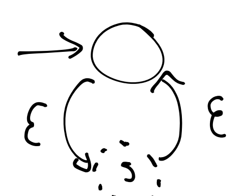
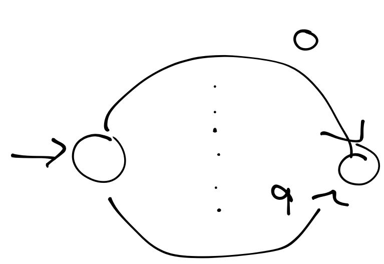


a* c





Convert a RE to NFA

Regex	Fragment
'a'	
'ε'	
st	
s|t	
s*	
[0-9]	

Jan 12

Derivation

Consider productions as rewrite rules whenever we have a non terminal.

$A \to axz$
$A \to tpQ$

$- A -$
$\Downarrow$
$- axz -$

① $\Rightarrow$ derivation
$\alpha N \beta \Rightarrow \alpha y \beta$, if $\exists N \to y$

② $\alpha \Rightarrow \alpha$

③ $\alpha \Rightarrow y$ if $\exists \beta$ such that $\alpha \Rightarrow \beta$
$\beta \Rightarrow y$

Grammar G

L(G) the set $\{w \in T^* \mid S \Rightarrow w\}$
↑ start production

FuncBody → '{' VarDecl * Statement* '}'
VarDecl → Type id ';'
Type → 'int'
→ 'float'

Statement → ';'
→ Expr ';'
→ id '=' expr ';'

Expr → int literal
→ id

Can you derive { int x; x=9; }

FuncBody

⇓

{ VarDec* Statement* }

⇓

{ Type id ';' Statement* }

⇓

{ int id(x); Statement* }

⇓

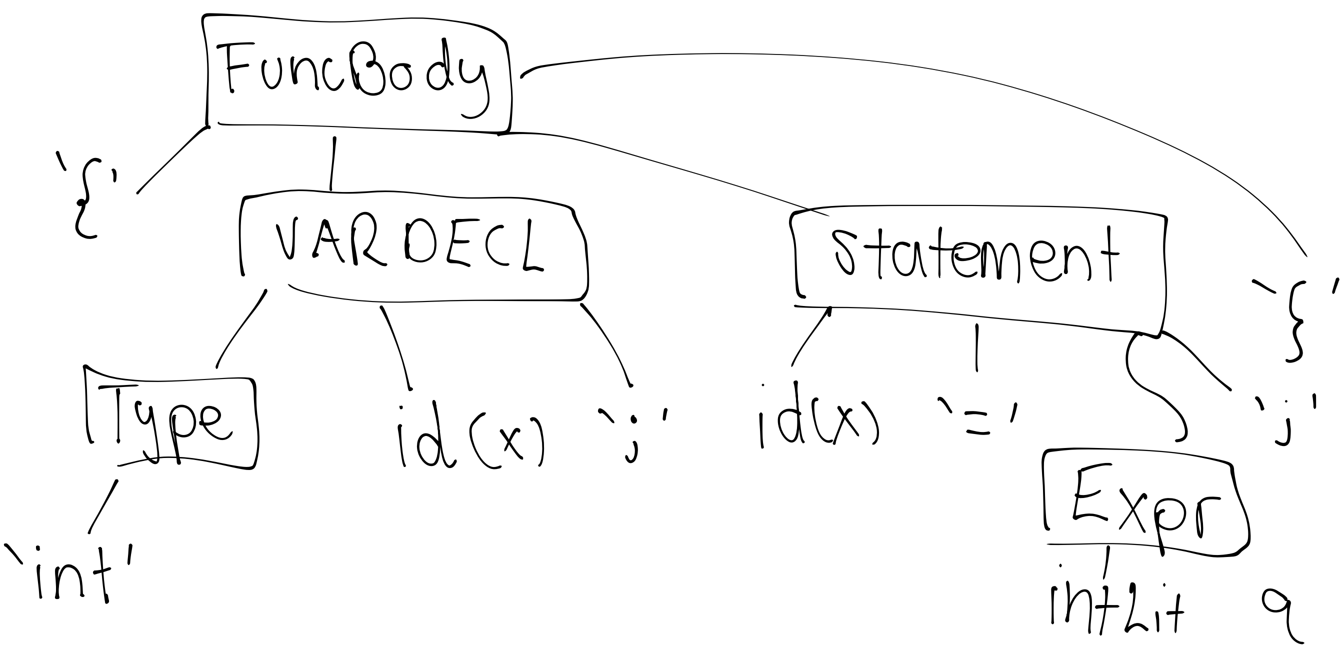
id '=' expr;

⇓

a

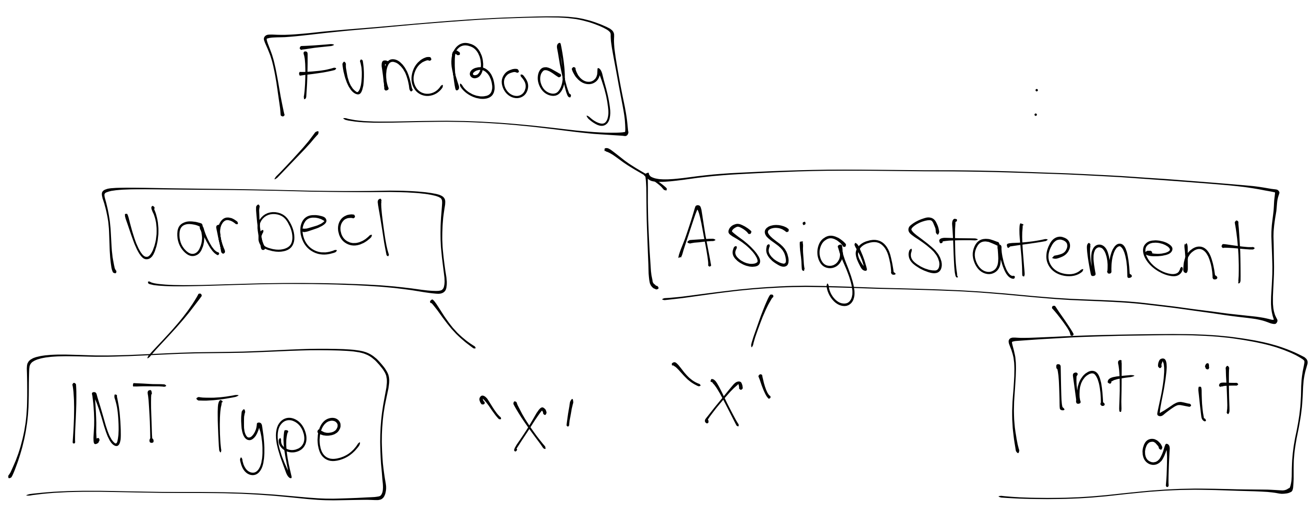
Parse Tree

{ int x; x=9 }



Abstract Syntax Tree (AST)

Each box is a class in Java



Nullable

$\text{Nullable}(\varepsilon) = \text{true}$

$\text{Nullable}(a) = \text{false}$

$\text{Nullable}(\alpha\beta) = \text{Nullable}(\alpha) \wedge \text{Nullable}(\beta)$

$\text{Nullable}(N) = \text{Nullable}(\alpha_1) \vee \dots \text{Nullable}(\alpha_n)$

where $N \to \alpha_1 \quad \dots \quad N \to \alpha_n$

First

$\text{First}(\varepsilon) = \emptyset$

$\text{First}(a) = \{a\}$

$$\text{First}(\alpha\beta) = \begin{cases} \text{First}(\alpha) \cup \text{FIRST}(\beta) & \text{if Nullable}(\alpha) \\ \text{First}(\alpha) & \text{else} \end{cases}$$

$\text{FIRST}(N) \quad \text{FIRST}(\alpha_1) \cup \dots \cup \text{FIRST}(\alpha_n)$

$N \to \alpha_1, \dots \; N \to \alpha_n$

Remove Left Recursion:

$A \to A\alpha_1 \mid A\alpha_2 \mid \dots A\alpha_n \mid \beta_1 \mid \dots \mid \beta_m$

α nonempty sequence of non-terminals and terminals

β: sequence of non-terminals and terminals not starting with A

Replace these with two sets of production:

$A \to \beta_1 A' \mid \dots \beta_m A'$

$A' \to \alpha_1 A' \mid \dots \mid \alpha_n A' \mid \varepsilon$

RHS

$T \to R$	T	$\{b\}$
$T \to aTc$	F	$\{a\}$
$R \to \varepsilon$	T	$\emptyset$
$R \to bR$	F	$\{b\}$
T	T	
R	T	

① Add non terminal $S' \to S\,\$$ as a new start symbol

② For each non-terminal N, locate all occurances of N on the right-hand sides of productions for each occurance:

(2.1) Let B be the RHS after the occurance of N

α, β may be empty

$[M \to \alpha N \beta]^*$

if the RHS contains several occurances of N we split for each one.

(2.2) Let $m = \text{FIRST}(B)$

Add constant

$m \subseteq \text{FOLLOW}(N)$

(2.3) If NULLABLE (B), add constant

$\text{FOLLOW}(M) \subseteq \text{FOLLOW}(N)$

Note if B is empty, NULLABLE (B) is true If M=N does not add anything

$T' \to T\$$

$T \to R$

$T \to aTc$

$R \to$

$R \to bR$

$\{\$\} \subseteq \text{Follow}(T)$ ①

$c \subseteq \text{FOLLOW}(T)$ ②

$\text{FOLLOW}(T) \subseteq \text{FOLLOW}(R)$ ③

T:

$T' \to T\$ \qquad m \to \alpha N \beta$ ①

$T' \to \alpha T \beta$

$\{\$\} \subseteq \text{Follow}(T)$

$T \to aTc \qquad T \to \alpha T \beta$ ②

$m = \text{FIRST}(\beta) = c$

$c \subseteq \text{FOLLOW}(T)$

R:

$T \to R \qquad T \to \alpha R \beta$ ③

$\text{NULLABLE}(\beta)$

$\text{FOLLOW}(T) \subseteq \text{FOLLOW}(R)$

$R \to bR \qquad R \to \alpha R \beta$

$\text{FOLLOW}(R) \subseteq \text{FOLLOW}(R)$

		i1	i2
FOLLOW(T)	∅	{\$, c}	{\$, c}
FOLLOW(R)	∅	{\$, c}	{\$, c}

Solve by

① Assume empty follow for all non-terminals

② Hande all constraints of the form

$$\text{FIRST}(\beta) \subseteq \text{FOLLOW}(N)$$

add

③ Hande all constraints of the form

$\text{FOLLOW}(M) \subseteq \text{FOLLOW}(N)$

$FOLLOW(M) \subseteq FOLLOW(N)$

add

④ Iterate until no changes.

Choose production $N \to \alpha$ on input symbol c if

- $c \in FIRST(\alpha)$

  OR

- $NULLABLE(\alpha)$ and c in $FOLLOW(N)$

LL(1)

Left to Right input processing
Left derivation
1 symbol look ahead

Midterm

① RE
② NFA, RE → NFA * NO DFA minimizing or conversion
③ Context Free Grammar
④ Parse Trees, derivation
⑤ Remove left recursion
operator precedence in LL grammars
⑥ FIRST, NULLABLE, FOLLOW
⑦ LL(1)

Follow: A terminal symbol a is in Follow(N) iff there is a derivation from the start symbol S of the grammar such that $S \Rightarrow \alpha N a \beta$ where α and β

are possibly empty sequence of symbols

Ambiguous grammar: find two different ways to derive a string

Exp → Exp + Exp2
Exp → Exp - Exp2
Exp → Exp2
Exp2 → Expr2 * Expr3
Exp2 → Exp 2 / Expr3
Expr2 → Expr3
Exp3 → (Exp)

A larger example

Nothing is nullable, we start w/ all false

First			
AB	$\emptyset$		FIRST(A)
BA	$\emptyset$		FIRST(B)
a	$\emptyset$	{a}	{a}
CAC	$\emptyset$		FIRST(C)
b	$\emptyset$	{b}	{b}
CBC	$\emptyset$		FIRST (C)
N	$\emptyset$		FIRST(A) ∪ FIRST(B)
A	$\emptyset$		FIRST(C) ∪ {a}
B	$\emptyset$		FIRST(C) ∪ {b}
C	$\emptyset$	{a, b}	

AB	$\emptyset$	$\{a\}$	$\{a,b\}$
BA	$\emptyset$	$\{b\}$	$\{a,b\}$
a	$\{a\}$	$\{a\}$	$\{a\}$
CAC	$\emptyset$	$\{a,b\}$	$\{a,b\}$
b	$\{b\}$	$\{b\}$	$\{b\}$
CBC	$\emptyset$	$\{a,b\}$	$\{a,b\}$

N	$\emptyset$	$\{a,b\}$	$\{a,b\}$
A	$\{a\}$	$\{a,b\}$	$\{a,b\}$
B	$\{b\}$	$\{a,b\}$	$\{a,b\}$
C	$\{a,b\}$	$\{a,b\}$	$\{a,b\}$

$S \to N\$$

$S \to N\$$	$\{\$\} \subseteq$ FOLLOW(N)
$N \to AB$	FIRST(B) $\subseteq$ FOLLOW(A), FOLLOW(N) $\subseteq$ FOLLOW(B)
$N \to BA$	FIRST(A) $\subseteq$ FOLLOW(B), FOLLOW(N) $\subseteq$ FOLLOW(A
$A \to a$	
$A \to CAC$	FIRST(C) $\subseteq$ FOLLOW(A), FOLLOW(A) $\subseteq$ FOLLOW(C)
	? FIRST(A) $\subseteq$ FOLLOW(C)
$B \to b$	FIRST(B) $\subseteq$ FOLLOW(C), FIRST(C) $\subseteq$ FOLLOW(B
$B \to CBC$	FOLLOW(B) $\subseteq$ FOLLOW(C)

$C \to a$
$C \to b$

FOLLOW(N)	$\emptyset$	$\{\$\}$
FOLLOW(B)	$\emptyset$	FOLLOW(N), FIRST(A), FIRST(C)
FOLLOW(A)	$\emptyset$	FIRST(B), FOLLOW(N), FIRST(C)
FOLLOW(C)	$\emptyset$	FOLLOW(A), FIRST(A), FIRST(B)

FOLLOW(N) $\{\$\}$
FOLLOW(B) $\{\$, a, b\}$

FOLLOW (B) {$, a, b}
FOLLOW (A) {$, a, b}
FOLLOW (C) {$, a, b}

Construction of LL(1) parsers

1. Eliminate ambiguity
2. Eliminate left-recursive
3. Perform left factorization as required
4. Add an extra start production $S' \rightarrow S\$$
5. Calculate FIRST for every production and FOLLOW for every non-terminal
6. For non terminal N and input symbol c choose $N \rightarrow a$ when:
   - $c \in$ FIRST(a) or
   - Nullable (a) and $c \in$ Follow(N)

Left factorized

Stat → id := Exp
Stat → if Exp then Stat Elsepart
Elsepart → else Stat
Elsepart →