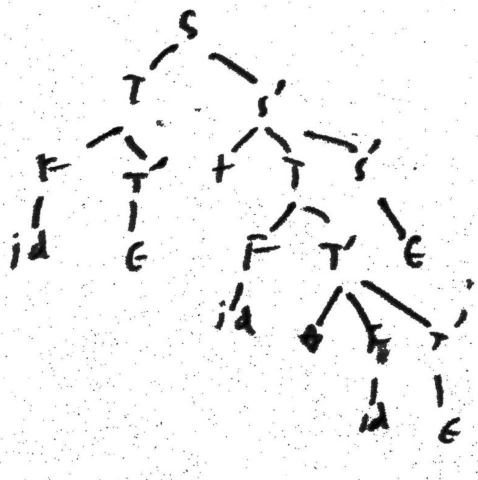
Parsing INPUT String a) id+id*id

stack	input	output
$S	id+id*id$	
$S'T	id+id*id$	S -> TS'
$S'T'F	id+id*id$	T -> FT'
$S'T'id	id+id*id$	F -> id
$S'T'	+id*id$	
$S'	+id*id$	T' -> ε
$S'T+	+id*id$	S' -> +TS'
$S'T	id*id$	
$S'T'F	id*id$	T -> FT'
$S'T'id	id*id$	F -> id
$S'T'	*id$	
$S'T'F*	*id$	
$S'T'	$	
$S'	$	T' -> ε
$	$	S' -> ε

Parse Tree:

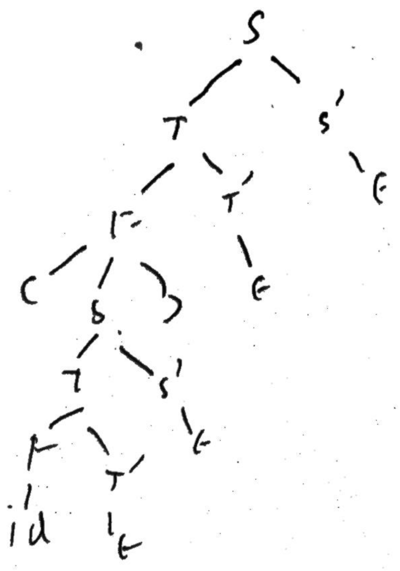


b) (id)

Parsing input string

stack	Input	Output
$ S	(id) $	
$ S' T	(id) $	S -> TS'
$ S' T' F	(id) $	T -> FT'
$ S' T') S (	(id) $	F -> (S)
$ S' T') S	id) $	
$ S' T') S' T	id) $	S -> TS'
$ S' T') S' T' F	id) $	T -> FT'
$ S' T') S' T' id	id) $	F -> id
$ S' T') S' T'	) $	
$ S' T') S'	) $	T' -> ε
$ S' T')	) $	S' -> ε
$ S' T'	$	
$ S'	$	T' -> ε
$	$	S' -> ε

Parse Tree:



2. Grammar:

$G \rightarrow SG'$

$G' \rightarrow +SG' \mid \epsilon$

$S \rightarrow FS'$

$S' \rightarrow *FS' \mid \epsilon$

$F \rightarrow id \mid (G)$

First and follow

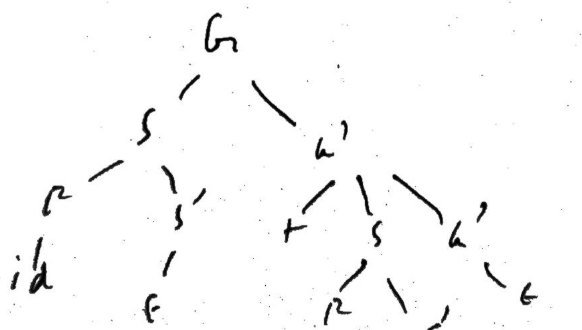
Non-Terminal	First	Follow
G	(, id	$,)
G'	+, ε	$,)
S	(, id	+, $,)
S'	+, ε	+, $,)
F	(, id	*, +, $,)

	*	+	(	)	id	$
G			G->SG'		G->SG'	
G'		G'->+SG'		G'->ε		G'->ε
S			S->FS'		S->FS'	
S'	S'->*FS'	S'->ε		S'->ε		S'->ε
F			F->(G)		F->id	

Parsing string:

Stack	Input	Output
$G	id+id*id$	
$G'S	id+id*id$	G→SG'
$G'S'R	id+id*id$	S→RS'
$G'S'id	id+id*id$	R→id
$G'S+	+id*id$	S'→ε
$G'S	id*id$	
$G'S'R	id*id$	R→id
$G'S'	*id$	S'→*RS'
$G'S'id	id$	
$G'S'	$	S'→ε
$	$	G'→ε

Parse Tree:



3 CFG:

a) $S \to AX \mid b$

$A \to aA \mid a$

$X \to bX \mid \epsilon$

Removing ϵ productions

$S \to A \mid AX \mid b$

$A \to aA \mid a$

$X \to b \mid bX$

Converting to CNF

$S \to A \mid AX \mid b$

$A \to aA \mid a$

$X \to b \mid bX$

$\Rightarrow$

$S \to A \mid AX \mid b$

$A \to CA \mid a$

$X \to b \mid DX$

$C \to a$

$D \to b$

$\Rightarrow$

$S \to CA \mid AX \mid a \mid b$

$A \to CA \mid a$

$X \to b \mid DX$

$C \to a$

$D \to b$

Converting to GNF:

$S \to aA \mid AX \mid a \mid b$

$A \to aA \mid a$

$X \to b \mid bX$

$\Rightarrow$

$S \to aA \mid aAX \mid a \mid b \mid aX$

$A \to aA \mid a$

$X \to b \mid bX$

b) $S \to aA \mid aB$

$A \to aaA \mid \epsilon$

$B \to bB \mid bbC$

$C \to b$

Removing ϵ productions:

$S \to a \mid aaaA \mid aB$

$A \to aa \mid aaA$

$B \to bB \mid bbb$

Converting to CNF:

$S \to a \mid aaaA \mid aB$

$A \to aa \mid aaA$

$B \to bB \mid bbb$

$\Rightarrow$

$S \to a \mid CA \mid CB$

$C \to a$

$D \to b$

$R \to CA$

$F \to DD$

$B \to DB \mid DF$ $\;$ $A \to CC \mid CR$

$G \to CA$

3) $S \rightarrow aSB \mid aA \mid AB$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

Remove ε productions:

$S \rightarrow aS \mid aSbB \mid a \mid aaA \mid aA \mid bB \mid aAbB$

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

CNF:

$S \rightarrow CS \mid EB \mid a \mid CB \mid CA \mid DB \mid EB$

$A \rightarrow a \mid CA$ $\quad\quad E \rightarrow aA$

$B \rightarrow b \mid DB$ $\quad\quad E \rightarrow DB$

$C \rightarrow a,\ D \rightarrow b$ $\quad\quad G \rightarrow aS$

Convert to GNF:

$S \rightarrow aS \mid aSDB \mid a \mid aCA \mid aA \mid bB \mid bB \mid aAD$

$C \rightarrow a$

$D \rightarrow b$

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

4) $E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid id$

```
lookahead = None
match (char)
{ if (lookahead == c)
            { lookahead = next char }
    else
         { raise exception }
}
```

```
E()
{ if (lookahead == '(')
   { match('(')
    E()
     match(')')
   }
   else if (lookahead == '1')
       { match('1')
       }
 }
main()
{  E() }
```

5 S → Aa | bAc | dc | bda

A → d

augmented Grammar:

1) S' → S

2) S → Aa

3) S → bAc

4) S → dc

5) S → bda

6) A → d

```
E()
{   T()
    E'()
 }
E'()
{   if (lookahead == '+')
      { match('+')
          T()
          E'()
      }
    else
      { return }
  }
T()
{   F()
  T'()
}
T'()
{ if (lookahead == '*')
      {match('*')
       F()
       T'()
    }
  else;
    return
 }
```

5) S → Aa | bAC | dc | bda

A → d

Augmented Grammar

1) S' → S

S' → da

S → bAC

S → dc

S → bda

A → d

I_0

S' → .S, $

S → .Aa, $

S → .bAC, $

S → .dc, $

S → .bda, $

A → .d, a

etc...

6) S → if R then S else S

S → id := num

R → id < num

R → id > num

R → id == num

S' → S

S → id

R → id

R → id

R → id

12) Code Generation

-> final phase of a compiler where intermediate representation of a program is translated into machine code that can be executed by a computer

Challenges

-> stalls have to be avoided
-> Code must be generated correctly
-> RISC/CISC architectures need different

a) Constant Folding :

Replaces expressions with constant values
Eg: i+4 - x = 2*3 => 2*3 is replaced with 6

b) Constant Propagation:

Replaces variables with known constant values
a=10

c) Dead code elimination

Removes code never executed

d) Loop optimization

moves calculations that are loop independent

e) Loop Unrolling:

Expands the loop to reduce number of iterations

f) Sub expression elimination

Removes repetitive expressions