23/01/23

Vidyalekhan™
DATE
PAGE

Compiler

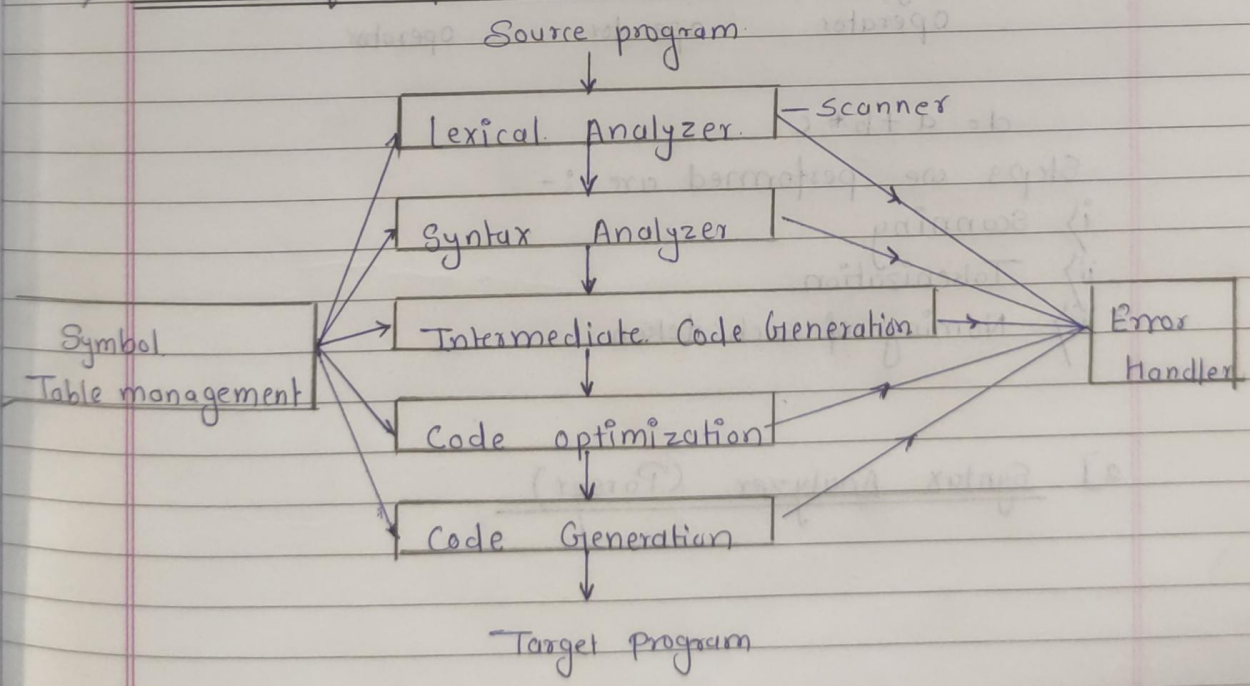
Source program → Translator (compiler) → Target Program

High level languages

(machine code)

Example of Translator — ① Interpreter
② Assembler — { ADD 1,2.
MOV A,B

★ Phase of compiler (13 Marks Question)



For (i = 0 ; i <= 10 ; i++)

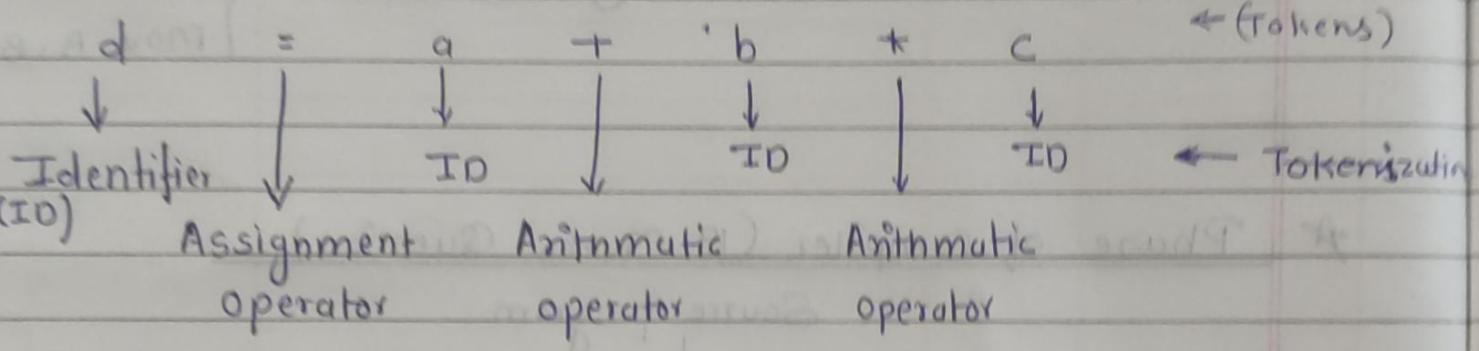
- For — keyword
- i — id
- = — assignment operator
- 0 — constant
- (— open parenthesis
- ; — punctuation mark
- i — id
- <= — conditional operator
- 10 — constant
- ; — punctuation mark
- i++ — incremental operator
-) — close parenthesis

If we have an expression

d = a + b * c

1] Lexical Analyzer (Scanner)

i) left to Right Scanning
ii) Then seperation of continuous expression into single element which coe will call it 'Token'



d = a + b * c

Steps coe performed are :-

i) Scanning
ii) Tokenization
iii) Naming of each token

2] Syntax Analyzer (Parser)

7/02/23

Vidyalekhan
DATE
PAGE

★ Cross Compiler:

Cross compiler is a compiler that runs on one machine & produces the object code on another machine.
The cross compiler is used to implement the compiler which is characterized by 3 languages.

1) The source language
2) The object language
3) The language in which it is written.

★ Bootstrap Compiler.

The compiler which is written in its own language is called bootstrap compiler.

Unit II Syntax Analysis (Parser)

★ Grammar

Set of 4 tuples.

$G = \{T, N, P, S\}$

T → Teminal Symbol
N → Non-teminal symbol
P → Production.
S → Start Symbol.

S → +aA;
A → b

- **Terminal Symbol.**
  Small ~~sy~~ alphabate, digit 0-9 and special symbol
  ∴ $T \rightarrow \{+, a, b, ;\}$

- **Non-teminal Symbol**
  Capital alphabate and which is produced production.

  ∴ $N \rightarrow \{A, S\}$

- **Production**
  $S \rightarrow +aA;$ & $A \rightarrow b$ are a production

- **Start Symbol**
  The S is a start symbol in $S \rightarrow +aAb$ because A is replaced by ~~A~~ b according to $A \rightarrow b$.

Vidyalekhan™
DATE
PAGE

★ Types of Grammar

1) Context Free Grammar

$X \rightarrow Y$

where, X is always Non-terminal and
Y is a combination of terminal & non-terminal or only terminal

eg $A \rightarrow b$ — only Terminal
$A \rightarrow bB$ — combination of terminal & non terminal

2) Context Sensitive Grammar.

$X \rightarrow Y$

where X and Y be the combination of terminal and non-terminal and x&y is the ~~non~~-terminal only

eg $aA \rightarrow b$ — X be combination of terminal & nonterminal
Y be the terminal.
$aA \rightarrow bB$ — X and Y both are combination

3) Right Linear Grammar

$A \rightarrow \omega B$
$B \rightarrow \omega$

where ω is always string of terminal symbol

eg $A \rightarrow aB$ in a is a string of terminal symbol and
$B \rightarrow a$ production of another.

left Right
Right left
A → cb/c
A → Right
cb/c → left hand side

4) Left Linear Grammar

$A \rightarrow B\omega$
$B \rightarrow \omega$

where ω is a string of terminal symbol

eg $A \rightarrow Bq$
$B \rightarrow q$

Syntax Analyzer (Parser)

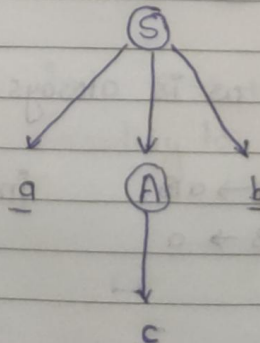
Two types of parser.

1) Top-down parser.
2) Bottom-Up parser.

1) Top-down Parser

S → Start Symbol
ω → String of teminal symbol

$S \rightarrow qAb$
$A \rightarrow cd/c$ } $A \rightarrow cd$, $A \rightarrow c$
$\omega \rightarrow abc$ } - Required string



Backtraking is possible in top-down parser?

8/02/23

Vidyalekhan

DATE
PAGE

* Why top-down parser is left most derivative in parser?

* Predictive Top-Down parser

1) FIRST() function. — For finding the set of terminal symbol

- How to calculate FIRST().

Rules → FIRST(α) → set of these terminals with which the strings derivable from α start.

If α = XYZ, then FIRST(α) is computed as follows

1) FIRST(α) = FIRST(XYZ) = {X} if X is terminal.

2) FIRST(α) = FIRST(XYZ) = FIRST(X) if X does not derive to an empty string.
FIRST(X) does not contains ∈

3) If FIRST(X) contains ∈, then FIRST(α) = FIRST(XYZ) = FIRST(X) - {∈} ∪ FIRST(YZ)

Rule 1) S → aBb
B → b

FIRST(S) → FIRST(aBb)
→ {a}

FIRST(B) → FIRST(b)
→ {b}

- Consider the grammer $S \rightarrow ACB / CbB / Ba$

  $A \rightarrow da / B\epsilon$

  $B \rightarrow g / \epsilon$

  $C \rightarrow h / \epsilon$

  Calcalate FIRST for each production.

$\Rightarrow$ FIRST(S) $\rightarrow$ FIRST(ACB) $\cup$ FIRST (CbB) $\cup$ FIRST (Ba) ——①

FIRST(A) $\rightarrow$ FIRST (da) $\cup$ FIRST(BC) —— ②

FIRST (B) $\rightarrow$ FIRST(g) $\cup$ FIRST (ϵ)

$\rightarrow \{g, \epsilon\}$

FIRST(C) $\rightarrow$ FIRST(h) $\cup$ FIRST(ϵ)

$\rightarrow \{h, \epsilon\}$

Vidyalekhan™
DATE
PAGE

② ⇒ FIRST(A) → FIRST(dq) ∪ FIRST(BC)
→ {d} ∪ FIRST(BC) — ③
→ {d} ∪ {g, h, ∈}
→ {d, g, h, ∈}

③ ⇒ FIRST(BC) → FIRST(B) ∪ FIRST(C)
→ {g, ∈} - {∈} ∪ {h, ∈}
→ {g, h, ∈} (.. put in eq ③)

① ⇒ First(Bq) → FIRST(B) - {∈} ∪ FIRST(q)
→ {g, ∈} - {∈} ∪ {q}
→ {g, q}

① ⇒ FIRST(CbB) → FIRST(C) - {∈} ∪ FIRST(bB)
→ {h, ∈} - {∈} ∪ {b}
→ {h, b}

① ⇒ FIRST(ACB) → FIRST(A) - {∈} ∪ FIRST(CB) — ④
→ {d, g, h, ∈} - {∈} ∪ {h, g, ∈}
→ {d, g, h, ∈}

④ ⇒ FIRST(CB) → FIRST(C) - {∈} ∪ FIRST(B)
→ {h, ∈} - {∈} ∪ {g, ∈}
→ {h, g, ∈}

① ⇒ FIRST(ACB) ∪ FIRST(CbB) ∪ FIRST(Bq)
→ {d, g, h, ∈} ∪ {h, b} ∪ {g, q}
→ {a, b, d, g, h, ∈}

13/02/23

Vidyalekhan
DATE
PAGE

- Consider the following grammer calculate the FIRST for every production.

$S \rightarrow ABC$
$A \rightarrow a / \epsilon$
$B \rightarrow r / \epsilon$
$C \rightarrow b / \epsilon$

FIRST(S) $\rightarrow$ FIRST(ABC) —①

FIRST(A) $\rightarrow$ FIRST(a) $\cup$ FIRST{ϵ}
$\rightarrow$ {a, ϵ}

FIRST(B) $\rightarrow$ FIRST(r) $\cup$ FIRST(ϵ)
$\rightarrow$ {r, ϵ}

FIRST(C) $\rightarrow$ FIRST(b) $\cup$ FIRST(ϵ)
$\rightarrow$ {b, ϵ}

① $\Rightarrow$ FIRST(S) $\rightarrow$ FIRST(ABC)
$\rightarrow$ FIRST(A) - {ϵ} $\cup$ FIRST(BC) —②
$\rightarrow$ ~~{r, b}~~ {a, ~~ϵ~~} - {~~ϵ~~} $\cup$ {r, b, ϵ}
$\rightarrow$ {a, r, b, ϵ}

② $\Rightarrow$ FIRST(BC) $\rightarrow$ FIRST(B) - {ϵ} $\cup$ FIRST(C)
$\rightarrow$ {r, ~~ϵ~~} - {~~ϵ~~} $\cup$ FIRST {b, ϵ}
$\rightarrow$ {r, b, ϵ}

① $\Rightarrow$ FIRST(S) $\rightarrow$ {a, r, b, ϵ}

Vidyalekhan™
DATE
PAGE

- Consider the following grammer find FIRST for every production

F → XYZa
X → x/ϵ
Y → y/ϵ
Z → z/ϵ

⇒ FIRST(F) → FIRST(XYZa)

FIRST(X) → FIRST(x) ∪ FIRST(ϵ)
→ {x, ϵ}

FIRST(Y) → FIRST(y) ∪ FIRST(ϵ)
→ {y, ϵ}

FIRST(Z) → FIRST(z) ∪ FIRST(ϵ)
→ {z, ϵ}

① ⇒ FIRST(F) → FIRST(X) - {ϵ} ∪ FIRST(YZa) — ②

② ⇒ FIRST(Y) → FIRST(Y) - {ϵ} ∪ FIRST(Za) — ③

③ ⇒ FIRST(Za) → FIRST(Z) - {ϵ} ∪ FIRST(a)
→ {z, ϵ} - {ϵ} ∪ {a}
→ {z, a}

② ⇒ FIRST(YZa) → FIRST(Y) - {ϵ} ∪ FIRST(Za)
→ {y, ϵ} - {ϵ} ∪ {z, a}
→ {y, z, a}

① ⇒ FIRST(F) → FIRST(X) - {ϵ} ∪ FIRST(YZa)
→ {x, ϵ} - {ϵ} ∪ {y, z, a}
→ {x, y, z, a}

15/02/23

Vidyalekhan
DATE
PAGE

★ Predictive Table Driver Parser

Consider the following grammer

S → aAb
A → cd / ef / ∈ - ?
calculate first()

FIRST(S) → FIRST (aAb) → {a} → {S, a}

FIRST (A) → FIRST (cd) U FIRST (ef) → A → cd = {A, c}
→ {c, e} A → ef = {A, e}

Table

	a	b	c	d	e	f	$
S	S → aAb						
A			A → cd		A → ef		

We use follow() function for finding the epsalon table if any production contain epsalon (∈).

★ How to Calculate

The derivation by A → ∈ is a right choice when the parser is on verge of the non-terminal A and the next input symbols happens to be a terminal which can occur immediately following

Vidyalekhan
DATE
PAGE

A in any string occuring on the right side of production. It is will late to the expansion of $A \rightarrow \epsilon$ and the next lead in the parse tree will be considered which is labeled by the symbol immediate following A and therefore may match next input symbol. Therefore we conclude that $A \rightarrow \epsilon$ is to be added in the table at [A, b] for every small b immediately follows A in any of the production is a right hand string. To compute the set of such terminals we make the use of function FOLLOW(A) where, A is a non-terminal and as defined below

* Rules -

FOLLOW(A) → Set of Terminal that immediately follow A in any string occuring on the right side of production of grammer.

If $A \rightarrow \alpha B \beta$ is a production, then follow(B) will be computed as

Rule 1) FOLLOW(B) = FOLLOW(β) if first(β) does not contain ϵ

Rule 2) FOLLOW(B) = ~~FOLLOW~~ FIRST(β) - {ϵ} ∪ FOLLOW(A) if first(β) contains ϵ

17/02/23

FOLLOW(S) ⟶ {$}
When S is not getting the RHS of production rule

eg1) S ⟶ aAb
A ⟶ ε
FOLLOW(A) ⟶ FIRST(b)
⟶ {b}

S ⟶ aAbc
A ⟶ c
FOLLOW(A) ⟶ FIRST(bc)
⟶ {b}

eg2) S ⟶ aAB
A ⟶ d
B ⟶ c/ε

FOLLOW(S) ⟶ {$}
FOLLOW(A) ⟶ FIRST(B) - {ε} U FOLLOW(S) —①
FIRST(B) ⟶ {c, ε}
FOLLOW(A) ⟶ {c, ε} - {ε} U {$}
⟶ {c, $}

- Consider the following grammer & generate the predictive parsing table

S ⟶ aABb
A ⟶ c/ε
B ⟶ d/ε

FIRST(S) → FIRST(aABb) ⟶ {a} (S, a)
FIRST(A) → FIRST(c) U FIRST(ε)
⟶ {c, ε}
FIRST(B) → FIRST(d) U FIRST(ε)
⟶ {d, ε}

Vidyalekhan
DATE
PAGE

$S \rightarrow aABb$

FOLLOW(A) $\rightarrow$ FIRST(Bb) — ①

FIRST(Bb) $\rightarrow$ FIRST(B) $\cup$ FIRST(b)

$\rightarrow$ FIRST(B) $- \{\epsilon\} \cup$ FIRST(b)

$\rightarrow \{d, \not{\epsilon}\} - \{\not{\epsilon}\} \cup \{b\}$

$\rightarrow \{d, b\}$

∴ FOLLOW(A) $\rightarrow \{d, b\}$ (A→d), (A,b)

FOLLOW(B) $\rightarrow$ FIRST(b)

$\rightarrow \{b\}$ (B,b)

Predictive parsing Table

	a	b	c	d	$
S	S→aABb				
A		A→ϵ	A→c	A→ϵ	
B		B→ϵ		B→d	

20/02/23
1st class

- Consider the grammer

  $F \rightarrow XYZa$

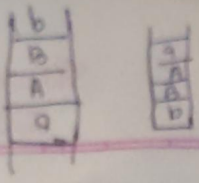
  $X \rightarrow x \mid \epsilon$

  $Y \rightarrow y \mid \epsilon$

  $Z \rightarrow z \mid \epsilon$

1) Find the FIRST() & FOLLOW()
2) Generate the parsing table.

⇒ as per the schedule of 13/02/23. for calculating of FIRST



Vidyalekhan
DATE
PAGE

$F \rightarrow XYZa$

FOLLOW(X) → FIRST(YZa)
→ {y, z, a}

FOLLOW(Y) → FIRST(Za)
→ {z, a}

FOLLOW(Z) → FIRST(a)
→ {a}

Predictive parsing Table

	x	y	z	a	$
F	$F \rightarrow XYZa$	$F \rightarrow XYZa$	$F \rightarrow XYZa$	$F \rightarrow XYZa$	
X	$X \rightarrow x$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	
Y		$Y \rightarrow y$	$Y \rightarrow \epsilon$	$Y \rightarrow \epsilon$	
Z			$Z \rightarrow z$	$Z \rightarrow \epsilon$	

20/02/23
2nd class

- consider the grammer
  $S \rightarrow aABb$
  $A \rightarrow c / \epsilon$
  $B \rightarrow d / \epsilon$

1) string $\omega \rightarrow acdb$ 2) $\omega \rightarrow ab$

The string is parse or not by stacking method/ ~~Stack im~~ top-down parser with stack?

2)

1) $\omega \rightarrow acdb$

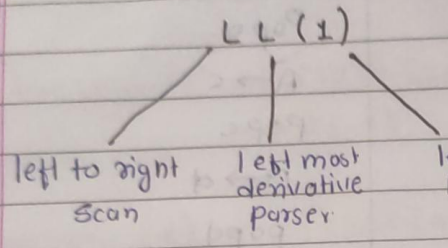
Stack	unspent I/p	Moves
$ S	acdb $	$S \rightarrow aABb$
$ bBAa	acdb $	Pop a
$ bBA	cdb $	$A \rightarrow c$
$ bBc	cdb $	Pop c
$ bB	db $	$B \rightarrow d$
$ bd	db $	pop d
$ b	b $	pop b
$	$	String is parse

2) $\omega \rightarrow ab$

Stacks	unspent I/p	Moves
$ S	ab $	$S \rightarrow aABb$
~~$ aABb~~	~~acdb $~~	
$ bBAa	ab $	pop a
$ bBA	b $	$A \rightarrow \epsilon$
$ bBϵ	ϵb $	pop ϵ
$ bB	b $	$B \rightarrow \epsilon$
$ b	b $	pop b
$	$	string is parse

Vidyalekhan
DATE
PAGE

★ LL(1) — parser.
LL(1) is top-down parser.



If the predictive table contain only single produ in single cell.

→ Algorithm
For every pair of production.
$A \rightarrow \alpha / \beta$

FIRST(α) ∩ FIRST(β) $\neq \phi$
{
grammer is LL(1)
}

21/02/23

• Test whether the grammer is LL(1) or not and construct a predictive parsing table for it.
$S \rightarrow AaAb / BbBa$
$A \rightarrow \epsilon$
$B \rightarrow \epsilon$

→ $S \rightarrow \underbrace{AaAb}_{\alpha} / \underbrace{BbBa}_{\beta}$

FIRST(α) ∩ FIRST(β) = ϕ — (1)

FIRST(AaAb) → FIRST(A) − {ϵ} ∪ FIRST(aAb)
→ {ϵ} − {ϵ} ∪ {a}
→ {a}

FIRST(BbBa) → FIRST(B) − {ϵ} ∪ FIRST(bBa)
→ {ϵ} − {ϵ} ∪ {b}
→ {b}

① ⇒ {a} ∩ {b} → {ϕ} → Hence, the grammer is LL(1)

iii) ⇒ FOLLOW(S) → {$}

i) FOLLOW(A) → FIRST(aAb) ∵ S → AaAb
→ {a}

ii) FOLLOW(A) → FIRST(b) ∵ S → AaAb
→ {b}

from ① and ⑪
FOLLOW(A) = {b, a}

i) FOLLOW(B) → FIRST(bBa) ∵ S → BbBa
→ {b}

ii) FOLLOW(B) → FIRST(a) ∵ S → BbBa
→ {a}

from ① and ⑪
FOLLOW(B) = {b, a}

FIRST(S) → FIRST(AaAb) ∪ FIRST(BbBa)
→ {a} ∪ {b}
→ {a, b} → (S, a), (S, b)

FIRST(A) → ϵ
FIRST(B) → ϵ

	a	b	$
S	S → AaAb	S → BbBa	
A	A → ϵ	A → ϵ	
B	B → ϵ	B → ϵ	

* Elimination of Left Recursion.

S → aIJh
I → IbSe / c
J → kLkr / ϵ
k → d / ϵ
L → p / ϵ

FIRST (S) → FIRST (aIJh)
→ {a}

FIRST (I) → FIRST (IbSe) ∪ FIRST (c)
→

In that we are getting the left Recursion

~~Consider the gram~~

If a grammer contains a pair of production of the form A → Aα/β then the grammer is called left recursive grammer.

Vidyalekhan™
DATE
PAGE

consider the grammer containing left recursion.
$A \rightarrow A\alpha / \beta$ where β does not begin with A to eliminate the left recursion replace the pair of production with $A \rightarrow \beta A'$

$A' \rightarrow \alpha A' / \epsilon$

New production will be.

Hence,

$S \rightarrow aIIh$ $\quad I \rightarrow \underline{Ib}\overset{\alpha}{\underline{Se}} / \overset{\beta}{\epsilon}$

$\rightarrow$

1) $I \rightarrow cI'$
2) $I' \rightarrow bSeI' / \epsilon$

Now, $S \rightarrow aIIh$
$I \rightarrow cI'$
$I' \rightarrow bSeI' / \epsilon$
$J \rightarrow klkr / \epsilon$
$k \rightarrow q / \epsilon$
$l \rightarrow p / \epsilon$