KONSEP
dan NOTASI BAHASA

Konsep dan Notasi bahasa

- Teknik Kompilasi merupakan kelanjutan dari konsep-konsep yang telah kita pelajari dalam teori bahasa dan automata
- Thn 56-59 Noam chomsky melakukan penggolongan tingkatan dalam bahasa, yaitu menjadi 4 class
- Penggolongan tingkatan itu disebut dengan hirarki Comsky
- 1959 Backus memperkenalkan notasi formal baru untuk syntax bahasa yang lebih spesifik
- Peter Nour (1960) merevisi metode dari syntax. Sekarang dikenal dengan BNF (backus Nour Form)

Konsep dan Notasi bahasa

- Tata bahasa (grammar) adalah sekumpulan dari himpunan variabel-variabel, simbol-simbol terminal, simbol non-terminal, simbol awal yang dibatasi oleh aturan-aturan produksi
- Aturan produksi adalah pusat dari tata bahasa yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu string ke bentuk lainnya

Konsep dan Notasi bahasa

- Syntax : suatu aturan yang memberitahu apakah sesuatu kalimat (string) adalah valid dalam program atau tidak
- Semantic : suatu aturan-aturan yang memberikan arti kepada program

Konsep dan Notasi bahasa

Penggolongan Chomsky

Bahasa	Mesin Automata	Aturan Produksi
Tipe 3 Atau Regular	Finite state automata (FSA) meliputi; deterministic Finite Automata (DFA) & Non Deterministic Finite Automata (NFA)	α adalah simbol variabel β maksimal memiliki sebuah simbol variabel yang bila ada terletak diposisi paling kanan
Tipe 2 Atau Contex Free	Push Down Automata	α adalah simbol variabel
Tipe 1 Atau Contex Sensitive	Linier Bounded Automata	$\|\alpha\| <= \|\beta\|$
Tipe 0 Atau Unrestricted/ Phase Structure/ natural language	Mesin Turing	Tidak ada Batasan

Keterangan

- α menyatakan simbol – simbol yang berada di ruas kiri aturan produksi
- β menyatakan simbol – simbol yang berada di ruas kanan aturan produksi
- Simbol-simbol terdiri dari simbol terminal dan non terminal/variabel (masih bisa diturunkan lagi)
- Simbol terminal biasanya dinyatakan dengan huruf kecil, sementara non terminal dengan huruf besar

Aturan Produksi

- **Tipe O** / Unrestricted: Tidak Ada batasan pada aturan produksi

  $Abc \rightarrow De$

- **Tipe 1** / Context sensitive: Panjang string ruas kiri harus lebih kecil atau sama dengan ruas kanan

  $Ab \rightarrow DeF$

  $CD \rightarrow eF$

- **Tipe 2** / Context free grammar: Ruas kiri haruslah tepat satu simbol variable

  $B \rightarrow CDeFg$

  $D \rightarrow BcDe$

- **Tipe 3** / Regular: Ruas kanan hanya memiliki maksimal 1 simbol non terminal dan diletakkan paling kanan sendiri

  $A \rightarrow e$
  $A \rightarrow efg$
  $A \rightarrow efgH$
  $C \rightarrow D$

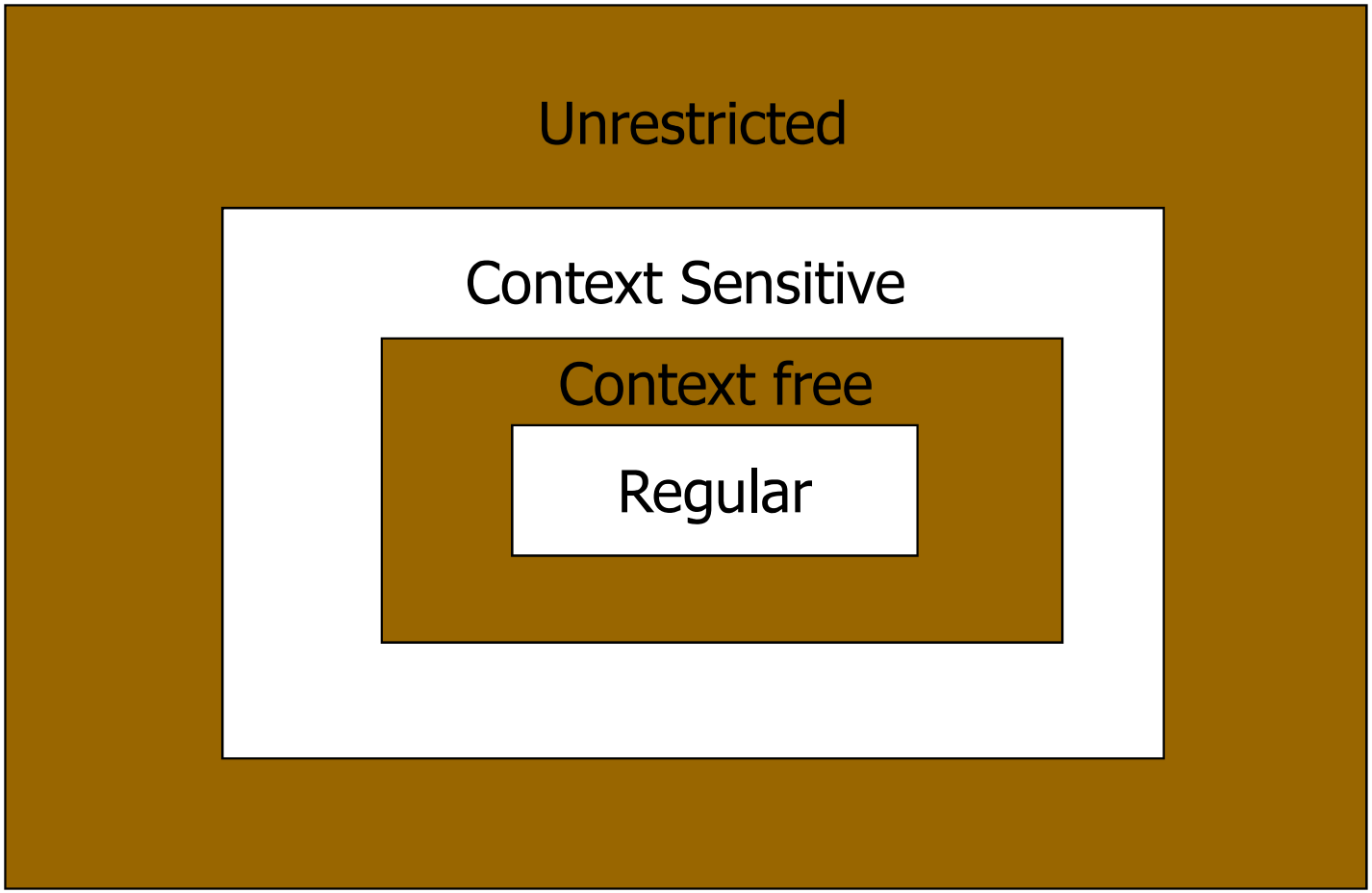
Aturan produksi yang tidak legal !!!

- Simbol ε tidak boleh berada pada ruas kiri

  misal $\varepsilon \rightarrow$ Abd
- Aturan produksi yang ruas kirinya hanya memuat simbol terminal saja

  misal : a $\rightarrow$ bd atau ab $\rightarrow$ bd

Hirarki Comsky



Contoh Tata Bahasa Sederhana

- <program> → **BEGIN** <Statement-list> **END**

- <Statement-list> → <statement> | <statement>; <statement-list>
- <statement> → <var> **:=** <expression>
- <Expression> → <term> | <term><op1> <expression>
- <term> → <factor> | <factor> <op2> <term>
- <factor> → <var> | <constant>

- <var> → **A|B|** ….| **Z**
- <op1> → + | - | =
- <op2> → **^** | * | /
- <constant> → <real_number> | <integer_part>

- <real_number> → <integer_part> . <fraction>
- <integer_part> → <digit> | <integer_part> < digit>
- <fraction> → <digit> | <digit> <fraction>
- <digit> → 0|1|….|9

Contoh

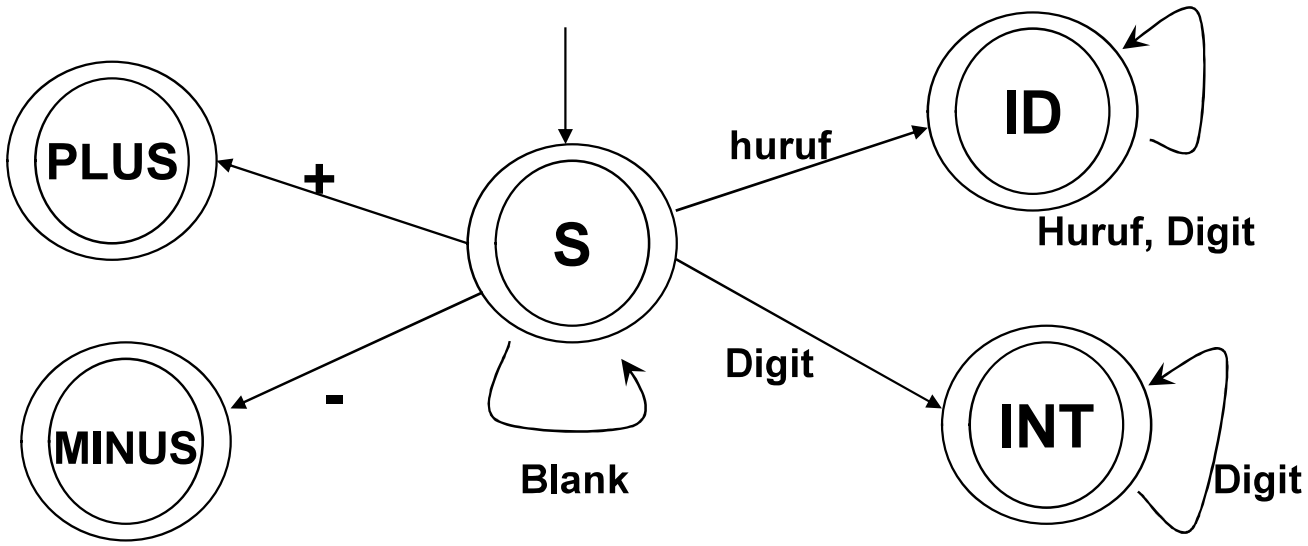
```
Begin
   A := 1;
   B := A + 2
END
```

Diagram State

- Digunakan untuk mendapatkan token, mempermudah melakukan analisis lexical

- **Token** adalah simbol **terminal** dari teori bahasa dan automata

Contoh : suatu tata bahasa memiliki himpunan simbol terminal/token berikut (ID, PLUS, MINUS, dan INT)
token ID untuk karakter huruf a-z, 0-9, token INT untuk digit, token PLUS untuk Penjumlahan dan token MINUS untuk Pengurangan



Notasi BNF (Backus-Nour Form)

- Aturan Produksi bisa dinyatakan dengan notasi BNF
- BNF menggunakan abstraksi untuk struktur syntax

::= sama identik dengan simbol $\rightarrow$

| sama dengan atau

< > pengapit simbol non terminal

{ } Pengulangan dari 0 sampai n kali

Misalkan aturan produksi sbb:

E $\rightarrow$ T | T+E | T-E

T $\rightarrow$ a

Notasi BNFnya adalah

E ::= <T> | <T> + <E> | <T> - <E>

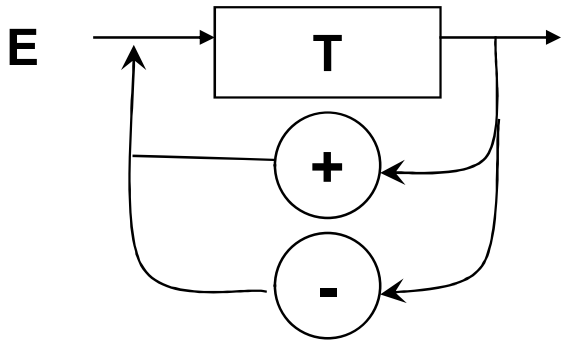
T ::= a

Diagram Syntax

- Alat bantu (tools) dalam pembuatan parser/ analisis sintaksis
- Menggunakan simbol persegi panjang untuk non terminal
- Lingkaran untuk simbol terminal

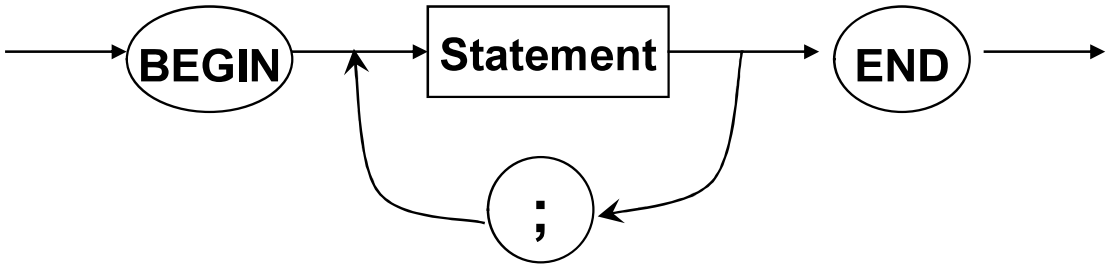
Misalnya

$E \rightarrow T \mid T+E \mid T-E$



BNF: <Block> ::= BEGIN <statement> { SEMICOL <statement>}

END



Kualitas dari Compiler

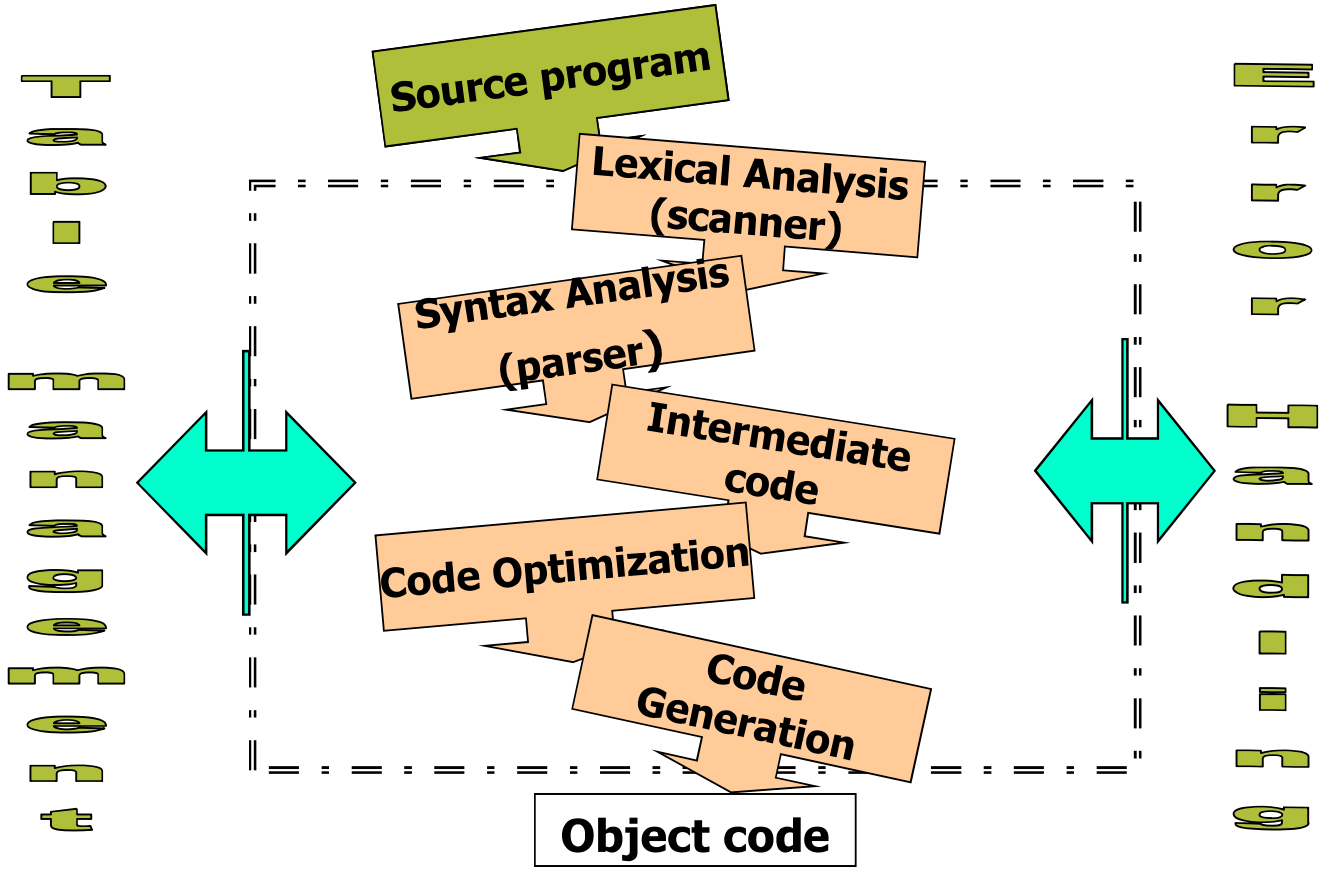
- Waktu yang dibutuhkan untuk kompilasi;

  tergantung dari

  - ✓ Algoritma compiler
  - ✓ Pembuat (compilator) Compiler itu sendiri

- Kualitas dari obyek program yang dihasilkan
  - ✓ Ukuran yang dihasilkan
- Fasilitas-fasilitas Integrasi yang lainnya
  - ✓ IDE (integrated Development Environment)

Struktur COMPILER



Keterangan

- Lexical Analyzer = scanner, Syntax Analyzer, dan Intermediate Code merupakan fungsi Analisis dalam compiler, yang bertugas mendekomposisi program sumber menjadi bagian-bagian kecil
- Code generation dan Code optimization adalah merupakan fungsi synthesis yang berfungsi melakukan pembangkitan / pembuatan dan optimasi program (object program)
- Scanner adalah mengelompok-an program asal/sumber menjadi token
- Parser (mengurai) bertugas memeriksa kebenaran dan urutan dari token-token yang terbentuk oleh scanner

Lexical Analysis (scanner) - berhubungan dengan bahasa

- Mengidentifikasikan semua besaran yang membuat suatu bahasa
- Mentransformasikan ke token-token
- Menentukan jenis dari token-token
- Menangani kesalahan
- Menangani tabel simbol
- Scanner, didesign untuk mengenali - keyword, operator, identifier
- Token : separates characters of the source language into group that logically belong together
- Misalnya : konstanta, nama variabel ataupun operator dan delimiter (atau sering disebut menjadi besaran lexical)

Lexical Analysis (*Besaran leksikal*)

- Identifier dapat berupa keyword atau nama kunci, seperti IF..ELSE, BEGIN..END (pada Pascal), INTEGER (pascal), INT, FLOAT (Bhs C)
- Konstanta : Besaran yang berupa bilangan bulat (integer), bilangan pecahan (float/Real), boolean (true/false), karakter, string dan sebagainya
- Operator; Operator arithmatika (+ - * /), operator logika (< = >)
- Delimiter; Berguna sebagai pemisah/pembatas, seperti kurung-buka, kurung -tutup, titik, koma, titik-dua, titik-koma, white-space
- White Space: pemisah yang diabaikan oleh program, seperti enter, spasi, ganti baris, akhir file

Lexical Analysis - Contoh

- **Contoh 1:**

  ada urutan karakter yang disebut dengan statement

  fahrenheit := 32 + celcius * 1.8,

  Maka akan diterjemahkan kedalam token-token seperti dibawah ini

identifier	→	fahrenheit
operator	→	:=
integer	→	32
operator penjumlahan	→	+
Identifier	→	celcius
operator perkalian	→	*
real / float	→	1.8

Lexical Analysis - Contoh 2

- Setiap bentuk dari token di representasi sebagai angka dalam bentuk internal, dan angkanya adalah unik
- ***Misalnya*** nilai 1 untuk variabel, 2 untuk konstanta, 3 untuk label dan 4 untuk operator, dst
- Contoh instruksi :
  - ***Kondisi : IF A > B THEN C = D***;
- Maka scanner akan mentransformasikan kedalam token-token, sbb:

Lexical Analysis - Contoh 2

Kondisi	3
:	26
IF	20
A	1
>	15
B	1
THEN	21
C	1
D	1
;	27

Token-token ini sebagai inputan untuk *syntax Analyser* , token-token ini bisa berbentuk pasangan item. Dimana Item pertama menunjukkan alamat atau lokasi dari token pada tabel simbol. Item kedua adalah representasi internal dari token. Semua token direpresentasikan dengan informasi yang panjangnya tetap (konstan), suatu alamat (address atau pointer) dan sebuah integer (bilangan bulat)

Syntax Analyzer

- Pengelompokan token-token kedalam class syntax (bentuk syntax), seperti procedure, Statement dan expression
- Grammar : sekumpulan aturan-aturan, untuk mendefinisikan bahasa sumber
- Grammar dipakai oleh syntax analyser untuk menentukan struktur dari program sumber
- Proses pen-deteksian-nya (pengenalan token) disebut dengan parsing

Syntax Analyzer

- Maka Syntax analyser sering disebut dengan

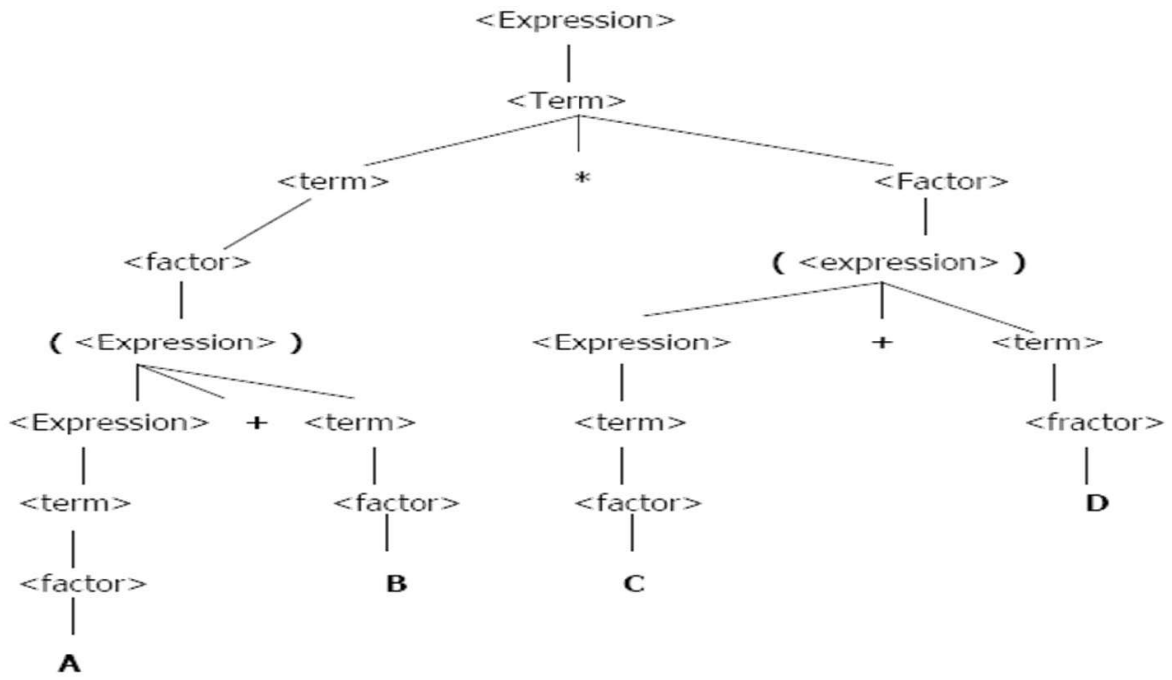
- Pohon sintaks yang dihasilkan digunakan untuk semantics analyser yang bertugas untuk menentukan ‘maksud’ dari program sumber.

- Misalnya operator penjumlahan maka semantics analyser akan mengambil aksi apa yang harus dilakukan

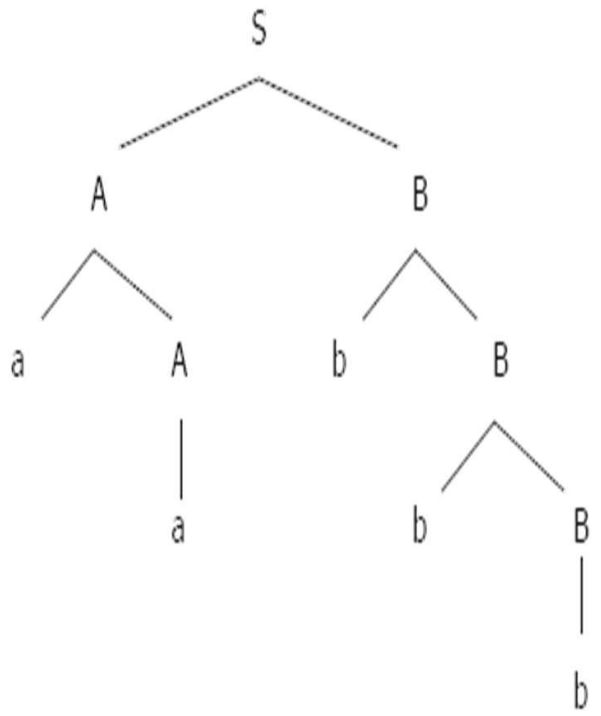
Contoh

- Terdapat statement : **(A + B) * (C + D)**
- Akan menghasilkan bentuk sintaksis:

  <factor>, <term> & <expression>



Syntax tree



- Pohon sintaks/ Pohon penurunan (syntax tree/ parse tree) beguna untuk menggambarkan bagaimana memperoleh suatu *string* dengan cara menurunkan simbol-simbol variable menjadi simbol-simbol terminal.

- Misalnya: $S \rightarrow AB$
  $A \rightarrow aA \mid a$
  $B \rightarrow bB \mid B$

Penurunan untuk menhasilkan string aabbb