OTOMATA 6

Konsep Grammar dan Klasifikasi Grammar

Mahasiswa memahami bahasa sebagai himpunan dan operasi^2-nya, cara mendefinisikan bahasa, serta cara mengenali anggota^2 bahasa

MATERI PERTEMUAN

- Konsep Dasar Grammar
- Derivasi dan Parse Tree
- Klasifikasi Grammar
- Tugas Mingguan III

GRAMMAR (1)

Grammar mendefinisikan bahasa secara rekursif

Definisi konseptual :

Grammar adalah sebuah sistem matematis yang digunakan untuk mendefinisikan bahasa.

Definisi formal :

Sebuah grammar G memiliki 4 tupel **(V_N, V_T, S, θ)** dengan **V_N adalah himpunan berhingga non-terminal**, **V_T adalah himpunan berhingga terminal**, **S adalah salah satu anggota V_N yang dijadikan start symbol**, dan **θ adalah himpunan berhingga production yang berbentuk $\alpha \rightarrow \beta$**

(dimana α adalah salah satu simbol dari himpunan V_N dan β berbentuk rangkaian terminal dan/atau non-terminal)

GRAMMAR (2)

Contoh :

Misal terdapat sebuah grammar $G = (V_N, V_T, S, \phi)$ untuk pembentukan identifier pada bahasa pemrograman, seperti berikut :

Sub-h{I, L, D}
impunan nonterminal :
V_N =
Sub-himpunan terminal :
V_T = {a, b, c, …, z, 0, 1, 2, …, 9}

Sub-himpunan start symbol :
S = I

Sub-himpunan poduction :
ϕ = {I $\rightarrow$ L, I $\rightarrow$ IL, I $\rightarrow$ ID, L $\rightarrow$ a, L $\rightarrow$ b, …, L $\rightarrow$ z,
D $\rightarrow$ 0, D $\rightarrow$ 1, D $\rightarrow$ 2, …, D $\rightarrow$ 9}

GRAMMAR (3)

Grammar untuk pembentukan identifier tersebut dapat pula diekspresikan/dituliskan/dinyatakan dalam bentuk lain seperti berikut :

$I \rightarrow L$
$I \rightarrow IL$
$I \rightarrow ID$

$L \rightarrow a$
$L \rightarrow b$
...
$L \rightarrow z$

$D \rightarrow 0$
$D \rightarrow 1$
...
$D \rightarrow 9$

Atau secara singkat dapat ditulis :

$I \rightarrow L \mid IL \mid ID$
$L \rightarrow a \mid b \mid \ldots \mid z$
$D \rightarrow 0 \mid 1 \mid \ldots \mid 9$

Konsensus :
Biasanya non-terminal ditulis dengan huruf besar. Selain itu (angka, karakter, simbol, tanda baca dan huruf kecil) adalah terminal, dan ditulis dengan huruf tebal/bold.

GRAMMAR (4)

Contoh :

Sebuah grammar $G = (V_N, V_T, S, \phi)$ untuk pembentukan bilangan bulat positif.

V_N = {ANGKA, DIGIT_AWAL, DIGIT_LAIN}

V_T = {0, 1, 2, …, 9}

S = ANGKA

ϕ = {ANGKA → DIGIT_AWAL, DIGIT_AWAL → DIGIT_AWAL DIGIT_LAIN,
DIGIT_AWAL → 1, DIGIT_AWAL → 2, …, DIGIT_AWAL → 9,
DIGIT_LAIN → 0, DIGIT_LAIN → 1, …, DIGIT_LAIN → 9}

Atau himpunan production di atas dapat pula diekspresikan dalam bentuk :

ANGKA → DIGIT_AWAL
DIGIT_AWAL → DIGIT_AWAL DIGIT_LAIN | 1 | 2 | … | 9
DIGIT_LAIN → 0 | 1 | 2 | … | 9

DERIVASI dan PARSE TREE (1)

Misalkan terdapat sebuah bahasa yang didefinisikan melalui grammar berikut :

$I \rightarrow L \mid IL \mid ID$
$L \rightarrow a \mid b \mid \ldots \mid z$
$D \rightarrow 0 \mid 1 \mid \ldots \mid 9$

Proses penurunan/derivasi dapat dilakukan pada sebuah string untuk menentukan keanggotan string tersebut terhadap bahasa di atas. Misal untuk string : **a15**

Leftmost Derivation

$I \Rightarrow \underline{I}D \Rightarrow \underline{I}DD \Rightarrow \underline{L}DD \Rightarrow \mathbf{a}\underline{D}D \Rightarrow \mathbf{a1}\underline{D} \Rightarrow \mathbf{a15}$

Rightmost Derivation

$I \Rightarrow I\underline{D} \Rightarrow \underline{I}\mathbf{5} \Rightarrow I\underline{D}\mathbf{5} \Rightarrow \underline{I}\mathbf{15} \Rightarrow \underline{L}\mathbf{15} \Rightarrow \mathbf{a15}$

Proses derivasi selalu diawali dari start symbol. Tetapi jika tidak diketahui, maka non-terminal yang berada pada ujung kiri atas grammar dapat dianggap sebagai start symbol.

DERIVASI dan PARSE TREE (2)

Derivasi dan Parse Tree adalah sebuah alat untuk mengidentifikasi keanggotaan sebuah bahasa. Jika diberikan sebuah string, maka melalui derivasi atau parsing dapat diketahui apakah string tersebut merupakan anggota dari bahasa yang bersangkutan atau bukan.

Disebut derivasi atau parsing karena prosesnya adalah men-derive ('menurunkan' / menelusuri) apakah dari sebuah grammar dapat dihasilkan string atau kalimat seperti yang diinginkan.

Dalam aplikasinya, derivasi menggunakan pendekatan substitusi linier, yakni dengan mengganti setiap nonterminal dengan terminal yang bersesuaian. Sedangkan parsing menggunakan 'pohon parsing' (parse tree) untuk membantu proses substitusi tersebut.

DERIVASI dan PARSE TREE (3)

Misalkan terdapat sebuah bahasa yang digunakan untuk mengekspresikan persamaan matematis sederhana seperti berikut :

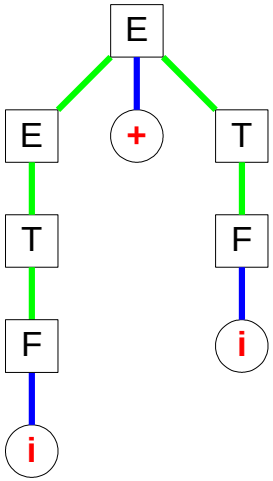
$E \rightarrow T \mid E + T \mid E - T$

$T \rightarrow F \mid T * F \mid T / F$

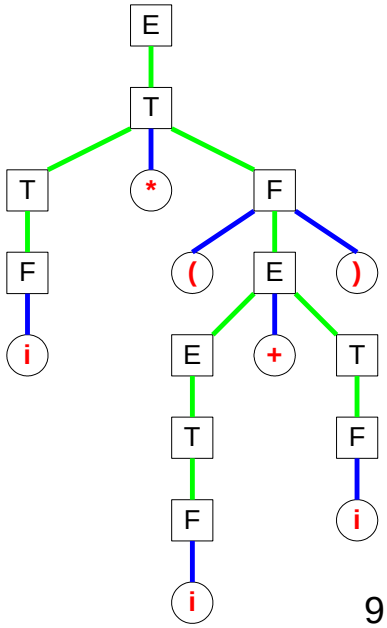
$F \rightarrow i \mid (E)$

Selain melalui derivasi, pengenalan sebuah string (ekspresi matematis) dapat pula dilakukan melalui teknik parsing (dengan cara membentuk parse tree)

Pohon parsing untuk ekspresi : **i + i**



Pohon parsing untuk ekspresi : **i * (i + i)**



KLASIFIKASI GRAMMAR (1)

Sintaks bahasa pemrograman umumnya dinyatakan melalui grammar, yang secara garis besar dibagi menjadi 2 klas utama, yaitu :

- **Backus-Naur Form (BNF)**
  Sebuah meta-language yang dikembangkan oleh Johan Backus dan Peter Naur. Didalam perkembangannya, cakupan BNF ini diperluas dan cara mengekspresikannya pun dirampingkan menjadi **EBNF** (***Extended Backus-Naur Form***).

- **Chomsky Normal Form (CNF)**
  Terbagi ke dalam 2 sub-klas :
  - **Unrestricted Grammar** (grammar kelas 0)
  - **Restricted Grammar**, yang terdiri dari 3 sub sub-klas :
    - **Context-Sensitive Grammar** (grammar kelas 1)
    - **Context-Free Grammar** (grammar kelas 2)
    - **Regular Grammar** (grammar kelas 3)

KLASIFIKASI GRAMMAR (2)

Backus-Naur Form (BNF)

Non-terminal ditulis <non-terminal>
Terminal ditulis terminal
Simbol " → " ditulis ::=

Contoh:

Grammar bahasa Pascal yang kali pertama ditulis oleh Niclaus Wirth menggunakan format BNF.

```
<identifier> ::= <letter> | <identifier> <letter> |
                 <identifier> <digit>
<letter>     ::= a | b | c | … | z
<digit>      ::= 0 | 1 | 2 | … | 9
dst…
```

KLASIFIKASI GRAMMAR (3)

Extended Backus-Naur Form (EBNF)

Perluasan EBNF berupa penambahan :

{ z } pengulangan nol atau lebih karakter z

[z] sintaks optional, dapat dipilih atau tidak

Contoh :

EBNF dalam bahasa Pascal.

<program> ::= <program_heading> ; <program_block>.
<program_heading> ::= <program> <identifier> [(<program_parameter>)]
<program_parameter> ::= <identifier_list>
<identifier_list> ::= <identifier> { , <identifier>}
<program_block> ::= <block>
<block> ::= <label_declaration_part>
    <constant_declaration_part>
    <type_declaration_part>
    <variable_declaration_part>
    <procedure_and_function_declaration_part>
    <statement_part>

KLASIFIKASI GRAMMAR (4)

Grammar Klas 0 : Unrestricted Grammar

Aturan-aturan sintaktik (productions) yang digunakan untuk membentuk kalimat tidak mempunyai batasan yang jelas.

Contoh :

$G = (\{S, A, B, C, D\}, \{a, b\}, S, \theta)$, dengan θ adalah :

$S \rightarrow CD$
$C \rightarrow aCA \mid bCB$
$AD \rightarrow aD$
$BD \rightarrow bD$

$Aa \rightarrow aA$
$Ab \rightarrow bA$
$Ba \rightarrow aB$
$Bb \rightarrow bB$

$C \rightarrow e$
$D \rightarrow e$

Bahasa yang didefinisikan grammar di atas adalah :

$$L(G) = \{ ww \mid w \in \{a, b\}\}$$

KLASIFIKASI GRAMMAR (5)

Grammar Klas 1 : Context-Sensitive Grammar

Grammar dengan production berbentuk $\alpha \rightarrow \beta$, dimana $|\alpha| \leq |\beta|$

Contoh :

$G = (\{S, A, B, C, D\}, \{a, b\}, S, \theta)$, dengan θ adalah :

$S \rightarrow aSBC \mid abC$ $\quad bB \rightarrow bb$ $\quad bC \rightarrow bc$ $\quad CB \rightarrow BC$ $\quad cC \rightarrow cc$

Misal diberi input string $a^2b^2c^2$, maka proses derivasi akan tampak seperti berikut :

$S \Rightarrow aSBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbcC \Rightarrow aabbcc$

KLASIFIKASI GRAMMAR (6)

Grammar Klas 2 : Context-Free Grammar (CFG)

Grammar dengan production yang berbentuk $\alpha \rightarrow \beta$, dimana $|\alpha| \leq |\beta|$ dengan $\alpha \in V_n$ dan $|\alpha| = 1$. Dengan demikian, production-production pada klas grammar ini hanya memiliki satu non-terminal di sisi kiri setiap productionnya.

Bahasa yang didefinisikan oleh CFG ini disebut **Context-Free Language**.

CFG merupakan satu-satunya klas grammar yang telah memiliki algoritma parsing yang optimal. Sehingga hampir semua bahasa pemrograman menggunakan CFG untuk mendefinisikan aturan-aturan sintaktik bahasanya.

Contoh : Bahasa = $\{ a^n b a^n \mid n \geq 1 \}$ didefinisikan melalui grammar berikut :

$S \rightarrow aCa$

$C \rightarrow aCa \mid b$

Derivasi untuk input string $a^3 b a^3$ adalah sebagai berikut :

$S \Rightarrow aCa \Rightarrow aaCaa \Rightarrow aaaCaaa \Rightarrow aaabaaa$

KLASIFIKASI GRAMMAR (7)

Grammar Klas 3 : Regular Grammar

Grammar dengan production yang berbentuk $\alpha \rightarrow \beta$, dimana $|\alpha| \leq |\beta|$ dengan $\alpha \in V_n$ dan $|\alpha| = 1$. Sedangkan β mempunyai bentuk aB atau a ($a \in V_T$ dan $B \in V_N$).

Bahasa yang didefinisikan oleh Regular Grammar ini disebut **Regular Language**.

Bahasa pemrograman yang menggunakan aturan sintaktik bahasa regular ini antara lain adalah javascript, perl, dll.

Contoh : Bahasa = $\{ a^n b a^m \mid n \geq 1 \}$ didefinisikan melalui grammar berikut :

$S \rightarrow aS \mid aB$

$C \rightarrow aC \mid a$

$B \rightarrow bC$

Derivasi untuk input string $a^3 b a^2$ adalah sebagai berikut :

$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaB \Rightarrow aaabC \Rightarrow aaabaC \Rightarrow aaabaa$

TUGAS

1. **Bahasa apakah yang didefinisikan oleh CFG berikut :**

   $S \rightarrow XbaaX \mid aX$

   $X \rightarrow Xa \mid Xb \mid \varepsilon$

   carilah sebuah contoh string yang dapat dikenali oleh CFG di atas melalui 2 derivasi yang berbeda.

2. **Gambarkan parse tree untuk input string di bawah menggunakan setiap CFG yang ada :**

   i). $S \rightarrow aS \mid aSb \mid X$
   $X \rightarrow aXa \mid a$

   ii). $S \rightarrow aAS \mid a$
   $A \rightarrow SbA \mid SS \mid ba$

   iii). $S \rightarrow aB \mid bA$
   $A \rightarrow a \mid aS \mid bAA$
   $B \rightarrow b \mid bS \mid aBB$

   aaaa, abaa, abab

3. **Melalui grammar di bawah :**

   $S \rightarrow E$
   $E \rightarrow T + E \mid T$
   $T \rightarrow F * T \mid F$
   $F \rightarrow (E) \mid i$

   buatlah left-most derivation untuk input string :

   a. ((i) * (i + i)) + i
   b. (i) + ((i))