

PERTEMUAN III

TEORI BAHASA & OPERASI MATEMATIS (2)

Mahasiswa memahami bahasa sebagai himpunan dan operasi²-nya, cara mendefinisikan bahasa, serta cara mengenali anggota² bahasa

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2005 – 2009

MATERI PERTEMUAN

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

GRAMMAR (1)

Grammar adalah sebuah alat untuk mendefinisikan bahasa secara rekursif.

Definisi konseptual :

Grammar adalah sebuah sistem matematis yang dapat mendefinisikan bahasa. Dan bahasa yang didefinisikan oleh grammar ini awalnya berupa himpunan string.

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-himpunan nonterminal :

$$V_N = \{I, L, D\}$$

Sub-himpunan terminal :

$$V_T = \{a, b, c, \dots, z, 0, 1, 2, \dots, 9\}$$

Sub-himpunan start symbol :

$$S = I$$

Sub-himpunan production :

$$\phi = \{I \rightarrow L, I \rightarrow IL, I \rightarrow ID, L \rightarrow a, L \rightarrow b, \dots, L \rightarrow z, \\ D \rightarrow 0, D \rightarrow 1, D \rightarrow 2, \dots, D \rightarrow 9\}$$

GRAMMAR (3)

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

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

Atau secara singkat dapat ditulis :

$$\begin{aligned} I &\rightarrow L \mid IL \mid ID \\ L &\rightarrow a \mid b \mid \dots \mid z \\ D &\rightarrow 0 \mid 1 \mid \dots \mid 9 \end{aligned}$$

Konsensus :

Biasanya non-terminal akan ditulis dengan huruf besar, selain itu (angka, karakter, simbol, tanda baca dan huruf kecil) merupakan terminal, yang ditulis dengan huruf kecil dan tebal/bold.



Contoh :

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

$V_N = \{ANGKA, DIGIT_AWAL, DIGIT_LAIN\}$

$V_T = \{0, 1, 2, \dots, 9\}$

$S = ANGKA$

$\phi = \{ANGKA \rightarrow DIGIT_AWAL, DIGIT_AWAL \rightarrow DIGIT_AWAL DIGIT_LAIN, DIGIT_AWAL \rightarrow 1, DIGIT_AWAL \rightarrow 2, \dots, DIGIT_AWAL \rightarrow 9, DIGIT_LAIN \rightarrow 0, DIGIT_LAIN \rightarrow 1, \dots, DIGIT_LAIN \rightarrow 9\}$

Atau himpunan production di atas dapat pula diekspresikan dalam bentuk :

$ANGKA \rightarrow DIGIT_AWAL$

$DIGIT_AWAL \rightarrow DIGIT_AWAL DIGIT_LAIN \mid 1 \mid 2 \mid \dots \mid 9$

$DIGIT_LAIN \rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9$

DERIVASI dan PARSE TREE (1)

Misalkan terdapat sebuah bahasa yang didefinisikan melalui grammar berikut :

$$\begin{aligned} I &\rightarrow L \mid IL \mid ID \\ L &\rightarrow a \mid b \mid \dots \mid z \\ D &\rightarrow 0 \mid 1 \mid \dots \mid 9 \end{aligned}$$

Prose penurunan/derivasi dapat dilakukan pada sebuah string (misal, **a15**) untuk menentukan keanggotan string tersebut terhadap bahasa di atas.

Leftmost Derivation

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

Rightmost Derivation

$$I \Rightarrow I\underline{D} \Rightarrow I\underline{5} \Rightarrow I\underline{D}5 \Rightarrow I\underline{1}5 \Rightarrow \underline{L}15 \Rightarrow \underline{a}15$$

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 (1)

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 grammar (sebuah bahasa) yang dijadikan rujukan dapat menghasilkan string atau kalimat seperti yang diberikan.

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

DERIVASI dan PARSE TREE (2)



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

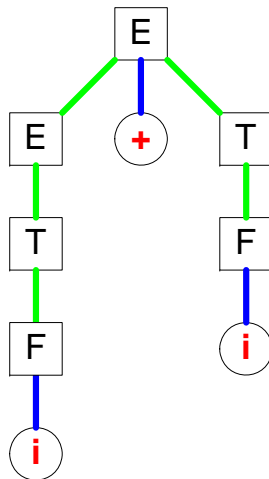
$$E \rightarrow T \mid E + T \mid E - T$$

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

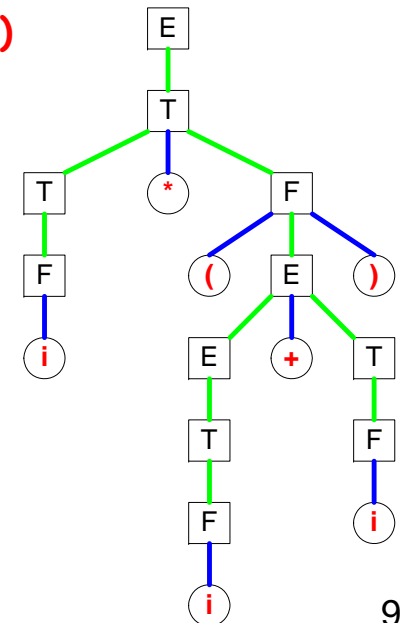
$$F \rightarrow i \mid (E)$$

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 : $i * (i + i)$





Sintaks bahasa pemrograman umumnya dinyatakan melalui grammar, yang secara garis besar dibagi menjadi 2 kelas 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-kelas :

- ◉ **Unrestricted Grammar** (grammar kelas 0) 

- ◉ **Restricted Grammar**, yang terdiri dari 3 sub sub-kelas :

- ◉ **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 pertama kali ditulis oleh Niclaus Wirth menggunakan format BNF.

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



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>
```



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 :

$S \rightarrow CD$	$Aa \rightarrow aA$	$C \rightarrow e$
$C \rightarrow aCA \mid bCB$	$Ab \rightarrow bA$	$D \rightarrow e$
$AD \rightarrow aD$	$Ba \rightarrow aB$	
$BD \rightarrow bD$	$Bb \rightarrow bB$	

Bahasa yang didefinisikan grammar di atas adalah :

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



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 :

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

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

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



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 mendefinifikan aturan-aturan sintaktik bahasanya.

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

$$\begin{aligned} S &\rightarrow aCa \\ C &\rightarrow aCa \mid b \end{aligned}$$

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

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



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 MINGGUAN III

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. $(i) + ((i)))$