

## BAB III

### CFG DAN PARSING

#### TUJUAN PRAKTIKUM

- 1) Memahami dan mengerti CFG.
- 2) Memahami dan mengerti metode parsing.

#### TEORI PENUNJANG

### 3.1. Pendahuluan

Bentuk umum produksi CFG adalah :

$$\alpha \rightarrow \beta, \alpha \in V_N, \beta \in (V_N \cup V_T)^*$$

Analisis sintaks adalah penelusuran sebuah kalimat (atau sentensial) sampai pada symbol awal grammar. Analisis sintaks dapat dilakukan melalui derivasi atau parsing. Penelusuran melalui parsing menghasilkan *pohon sintaks*.

Contoh 1 :

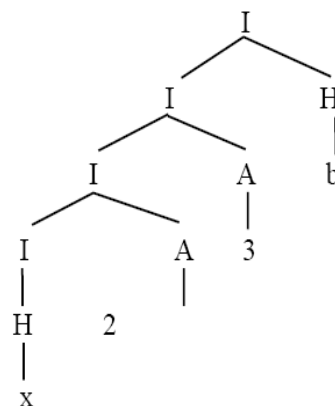
Diketahui grammar  $G_1 = \{I \rightarrow H|IH|IA, H \rightarrow a|b|c|\dots|z, A \rightarrow 0|1|2|\dots|9\}$

dengan I adalah simbol awal. Berikut ini kedua cara analisa sintaks untuk kalimat x23b.

**Cara 1 (derivasi) :**

$I \Rightarrow IH$   
 $\Rightarrow IAH$   
 $\Rightarrow IAAH$   
 $\Rightarrow HAAH$   
 $\Rightarrow xAAH$   
 $\Rightarrow x2AH$   
 $\Rightarrow x23H$   
 $\Rightarrow x23b$

**Cara 2 (parsing) :**

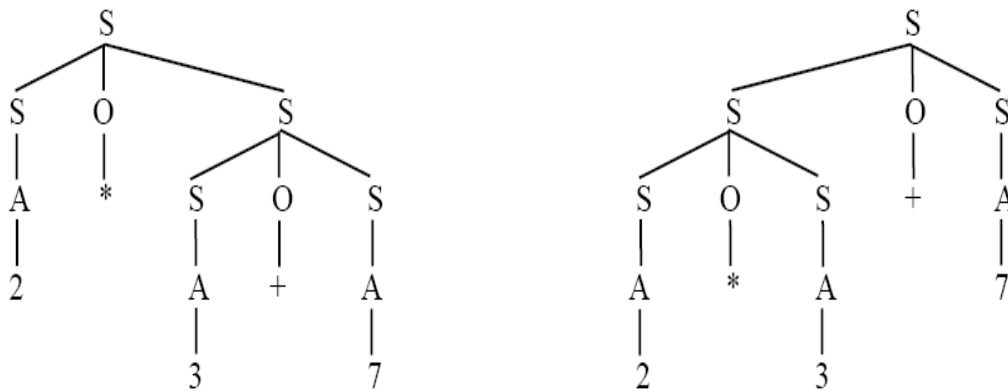


*Sebuah kalimat dapat saja mempunyai lebih dari satu pohon.*

Contoh 2 :

Diketahui grammar  $G_2 = \{S \rightarrow SOS|A, O \rightarrow *|+, A \rightarrow 0|1|2|\dots|9\}$

Kalimat :  $2*3+7$  mempunyai dua pohon sintaks berikut :



Sebuah kalimat yang mempunyai lebih dari satu pohon sintaks disebut *kalimat ambigu* (*ambiguous*). Grammar yang menghasilkan paling sedikit sebuah kalimat ambigu disebut *grammar ambigu*.

### 3.2. Metode Parsing

Ada 2 metoda parsing : *top-down* dan *bottom-up*.

- Parsing *top-down* :

Diberikan kalimat  $x$  sebagai input. Parsing dimulai dari simbol awal  $S$  sampai kalimat  $x$  nyata (atau tidak nyata jika kalimat  $x$  memang tidak bisa diturunkan dari  $S$ ) dari pembacaan semua *leaf* dari pohon parsing jika dibaca dari kiri ke kanan.

- Parsing *bottom-up* :

Diberikan kalimat  $x$  sebagai input. Parsing dimulai dari kalimat  $x$  yang nyata dari pembacaan semua *leaf* pohon parsing dari kiri ke kanan sampai tiba di simbol awal  $S$  (atau tidak sampai di  $S$  jika kalimat  $x$  memang tidak bisa diturunkan dari  $S$ ).

### 3.2.1 Parsing Top-Down

Ada 2 kelas metoda parsing *top-down*, yaitu kelas metoda *dengan backup* dan kelas metoda *tanpa backup*. Contoh metoda kelas *dengan backup* adalah metoda *Brute-Force*, sedangkan contoh metoda kelas *tanpa backup* adalah metoda *recursive descent*.

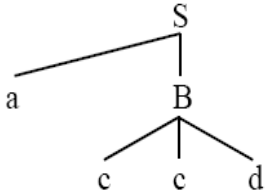
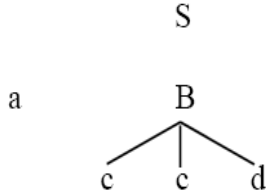
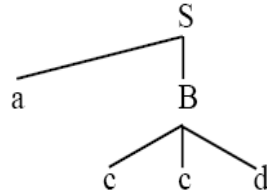
#### 3.2.1.1 Metode Brute-Force

Kelas metoda *dengan backup*, termasuk metoda *Brute-Force*, adalah kelas metoda parsing yang menggunakan produksi alternatif, jika ada, ketika hasil penggunaan sebuah produksi tidak sesuai dengan simbol input. Penggunaan produksi sesuai dengan nomor urut produksi.

Contoh 3 :

Diberikan grammar  $G = \{S \rightarrow aAd|aB, A \rightarrow b|c, B \rightarrow ccd|ddc\}$ . Gunakan metoda Brute-Force untuk melakukan analisis sintaks terhadap kalimat  $x = accd$ .

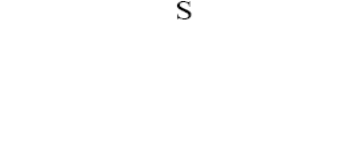
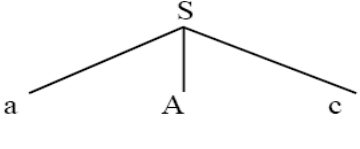
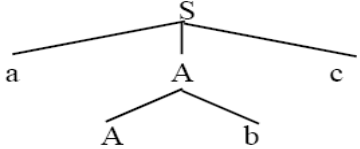
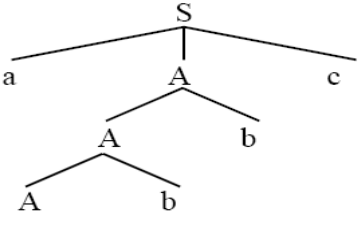
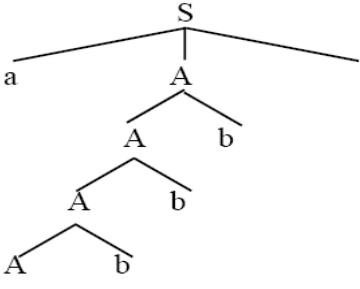
<p>S</p> <p>Hasil : Input :           Sisa : accd <u>Penjelasan</u> : Gunakan produksi S pertama. Masukkan simbol terkiri kalimat sebagai input.</p>	<p>S</p> <p>a           A           d</p> <p>Hasil : a Input : a           Sisa : ccd <u>Penjelasan</u> : Hasil = Input. Gunakan produksi A pertama.</p>	<p>S</p> <p>a           A           d</p> <p>                      b</p> <p>Hasil : ab Input : ac           Sisa : cd <u>Penjelasan</u> : Hasil <math>\neq</math> Input. <u>Backup</u> : Gunakan produksi A alternatif pertama.</p>
<p>S</p> <p>a           A           d</p> <p>                      c</p> <p>Hasil : ac Input : ac           Sisa : cd <u>Penjelasan</u> : Hasil = Input. Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.</p>	<p>S</p> <p>a           A           d</p> <p>                      c</p> <p>Hasil : acd Input : acc           Sisa : c <u>Penjelasan</u> : Hasil <math>\neq</math> Input. Tidak ada lagi produksi A alternatif, <u>backup</u> : gunakan produksi S alternatif pertama.</p>	<p>S</p> <p>a                           B</p> <p>Hasil : a Input : a           Sisa : ccd <u>Penjelasan</u> : Hasil = Input. Gunakan produksi B pertama.</p>

 <p>           Hasil : ac            Input : ac      Sisa : cd  <u>Penjelasan</u> : Hasil = Input.            Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.         </p>	 <p>           Hasil : acc            Input : acc      Sisa : d  <u>Penjelasan</u> : Hasil = Input.            Karakter berikutnya adalah simbol terminal, Hasil dibandingkan dengan Input.         </p>	 <p>           Hasil : accd            Input : accd      Sisa :  <u>Penjelasan</u> : Hasil = Input.            SELESAI, SUKSES         </p>
--	---	--

Metoda *Brute-Force* tidak dapat menggunakan grammar rekursi kiri, yaitu grammar yang mengandung produksi rekursi kiri (*left recursion*) :  $A \rightarrow A\alpha$ . Produksi rekursi kiri akan menyebabkan parsing mengalami looping tak hingga.

Contoh 4 :

Diberikan grammar  $G = \{S \rightarrow aAc, A \rightarrow Ab|e\}$ . Gunakan metoda *Brute-Force* untuk melakukan analisis sintaks terhadap kalimat  $x = ac$ .

 <p>           Hasil :            Input :      Sisa : ac  <u>Penjelasan</u> : Masukkan simbol ter kiri kalimat sebagai input. Gunakan produksi S pertama.         </p>	 <p>           Hasil : a            Input : a      Sisa : c  <u>Penjelasan</u> : Hasil = Input. Gunakan produksi A pertama.         </p>	 <p>           Hasil : a            Input : a      Sisa : c  <u>Penjelasan</u> : Hasil = Input. Gunakan produksi A pertama.         </p>
		<p>dan seterusnya..... (looping)</p>

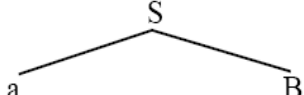
Agar tidak menghasilkan looping tak hingga, grammar rekursi kiri harus ditransformasi. Untuk contoh di atas transformasi berarti merubah produksi  $A \rightarrow Ab$  menjadi  $A \rightarrow bA$ .

### 3.2.1.2 Metode Recursive-Descent

- Kelas metoda *tanpa backup*, termasuk metoda *recursive descent*, adalah kelas metoda parsing yang tidak menggunakan produksi alternatif ketika hasil akibat penggunaan sebuah produksi tidak sesuai dengan simbol input. Jika produksi A mempunyai dua buah ruas kanan atau lebih maka produksi yang dipilih untuk digunakan adalah *produksi dengan simbol pertama ruas kanannya sama dengan input yang sedang dibaca*. Jika tidak ada produksi yang demikian maka dikatakan bahwa parsing tidak dapat dilakukan.
- Ketentuan produksi yang digunakan metoda *recursive descent* adalah : *Jika terdapat dua atau lebih produksi dengan ruas kiri yang sama maka karakter pertama dari semua ruas kanan produksi tersebut tidak boleh sama*. Ketentuan ini tidak melarang adanya produksi yang bersifat rekursi kiri.

Contoh 5 :

Diketahui grammar  $G = \{S \rightarrow aB \mid A, A \rightarrow a, B \rightarrow b \mid d\}$ . Gunakan metoda *recursive descent* untuk melakukan analisis sintaks terhadap kalimat  $x = ac$ .

<p style="text-align: center;">S</p> <p>Hasil : Input :            Sisa : ab <u>Penjelasan</u> : Masukkan simbol terkiri kalimat sebagai input. Gunakan produksi S dengan simbol pertama ruas kanan = a</p>	<p style="text-align: center;">  </p> <p>Hasil : a Input : a            Sisa : c <u>Penjelasan</u> : Hasil = Input. Gunakan produksi B dengan simbol pertama ruas kanan = c. Karena produksi demikian maka parsing gagal dilakukan.</p>	<p style="text-align: center;">SELESAI,  PARSING GAGAL</p>
---	--	--

### 3.2.2 Parsing Bottom-Up

Salah satu contoh menarik dari parsing *bottom-up* adalah parsing pada *grammar preseden sederhana* (GPS). Sebelum sampai ke parsing tersebut, akan dikemukakan beberapa pengertian dasar serta relasi yang ada pada GPS, yaitu :

- Jika  $\alpha$  dan  $x$  keduanya diderivasi dari simbol awal grammar tertentu, maka  $\alpha$  disebut *sentensial* jika  $\alpha \in (V T \mid V N)^*$ , dan  $x$  disebut *kalimat* jika  $x \in (V T)^*$
- Misalkan  $\alpha = Q_1 \beta Q_2$  adalah sentensial dan  $A \in V N$  :
  - $\beta$  adalah *frase* dari sentensial  $\alpha$  jika :  $S \Rightarrow \dots \Rightarrow Q_1 A Q_2$  dan  $A \Rightarrow \dots \Rightarrow \beta$
  - $\beta$  adalah *simple frase* dari sentensial  $\alpha$  jika :  $S \Rightarrow \dots \Rightarrow Q_1 A Q_2$  dan  $A \Rightarrow \beta$
  - Simple frase ter kiri dinamakan *handel*
  - Frase, simple frase, dan handel adalah string dengan panjang 0 atau lebih.

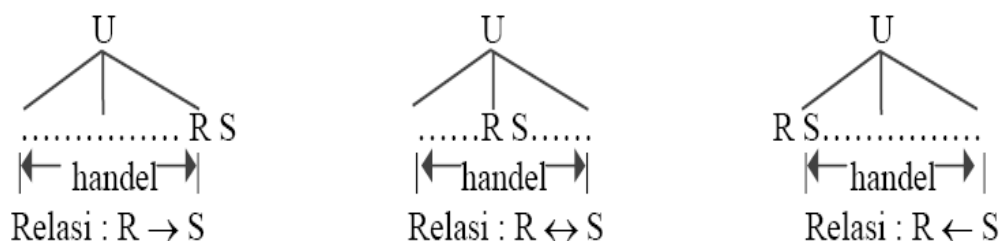
Contoh 6 :

- (1)  $I \Rightarrow I H H b$  adalah sentensial dan  $b$  adalah simple frase  
 $\Rightarrow H H$  (dibandingkan dengan  $Q_1 \beta Q_2$  maka  $Q_1 = H$ ,  $\beta = b$ , dan  $Q_2 = \epsilon$ )  
 $\Rightarrow H b$  Perhatikan : simple frase ( $b$ ) adalah yang terakhir diturunkan
- (2)  $I \Rightarrow I H H b$  adalah sentensial dan  $H$  adalah simple frase  
 $\Rightarrow I b$  (dibandingkan dengan  $Q_1 \beta Q_2$  maka  $Q_1 = \epsilon$ ,  $\beta = H$ , dan  $Q_2 = b$ )  
 $\Rightarrow H b$  Perhatikan : simple frase ( $H$ ) adalah yang terakhir diturunkan

Sentensial  $Hb$  mempunyai dua simple frase ( $b$  dan  $H$ ), sedangkan handelnya adalah  $H$ .

#### 3.2.2.1 Relasi Preseden dan Grammar Preseden Sederhana

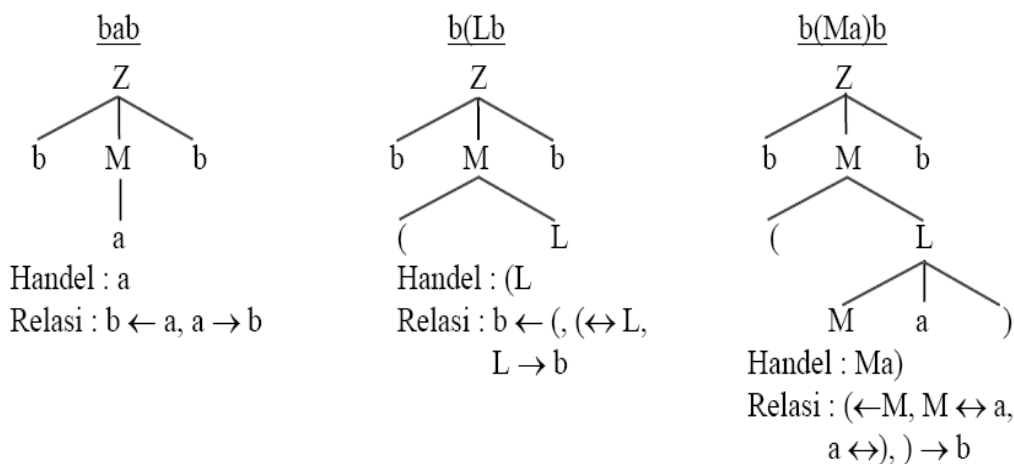
- Relasi preseden adalah relasi antara 2 simbol grammar (baik  $V T$  maupun  $V N$ ) dimana paling tidak salah satu simbol tersebut adalah komponen handel.
- Misalkan  $S$  dan  $R$  adalah 2 simbol. Ada 3 relasi preseden yang :  $\leftarrow$ ,  $\leftrightarrow$ , dan  $\rightarrow$



Gambar 3.1 : Komponen handel selalu ‘menunjuk’ yang simbol lainnya

Contoh 7 :

Diketahui grammar dengan  $G = \{Z \rightarrow bMb, M \rightarrow (L \mid a, L \rightarrow Ma)\}$ . Dari 3 sentensial :  $bab, b(Lb, b(Ma)b$ , tentukan handel dan relasi yang ada.



- Secara umum : jika  $A \rightarrow aBc$  adalah sebuah produksi maka :
  - $aBc$  adalah handel dari sentensial yang mengandung string “ $aBc$ ”
  - relasi preseden antara  $a, B$ , dan  $c$  adalah :  $a \leftrightarrow B, B \leftrightarrow c$
- Dengan memperhatikan ruas kanan produksi yang ada serta berbagai sentensial yang dapat diderivasi dari  $Z$  maka semua relasi preseden tercantum dalam tabel berikut :

Tabel 3.1. Tabel Relasi Preseden

	Z	b	M	L	a	(	)
Z							
b			$\leftrightarrow$		$\leftarrow$	$\leftarrow$	
M		$\leftrightarrow$			$\leftrightarrow$		
L		$\rightarrow$			$\rightarrow$		
a		$\rightarrow$			$\rightarrow$	$\leftarrow$	$\leftrightarrow$
(			$\leftarrow$	$\leftrightarrow$	$\leftarrow$	$\leftarrow$	
)		$\rightarrow$					

Grammar G disebut *grammar preseden sederhana* jika :

1. paling banyak terdapat satu relasi antara setiap dua simbolnya
2. tidak terdapat dua produksi produksi dengan ruas kanan yang sama

### 3.2.2.2 Parsing Grammar Preseden Sederhana

Prosedur parsing :

1. Buat tabel 3 kolom dengan label : sentensial dan relasi, handel, dan ruas kiri produksi.
2. Tuliskan kalimat (atau sentensial) yang diselidiki pada baris pertama kolom pertama.
3. Dengan menggunakan tabel relasi preseden cantumkan relasi preseden antara setiap dua simbol yang bertetangga.
4. Tentukan handel dari sentensial tersebut. Handel adalah string yang dibatasi " $\leftarrow$ " terakhir dan " $\rightarrow$ " pertama jika dilakukan penelusuran dari kiri atau yang saling mempunyai relasi " $\leftrightarrow$ ". Handel tersebut pastilah merupakan ruas kanan produksi, karena itu tentukan ruas kiri dari handel tersebut.
5. Ganti handel dengan ruas kiri produksinya. GOTO 3.
6. Kalimat yang diselidiki adalah benar dapat diderivasi dari simbol awal jika kolom "ruas kiri produksi" menghasilkan simbol awal.

Contoh 8 :

Lakukan parsing atas kalimat  $x = b(aa)b$  berdasarkan grammar G di atas.



sentensial dan relasi	handel	ruas kiri produksi
$b \leftarrow (\leftarrow a \rightarrow a \leftrightarrow) \rightarrow b$	a	M
$b \leftarrow (\leftarrow M \leftrightarrow a \leftrightarrow) \rightarrow b$	Ma)	L
$b \leftarrow (\leftrightarrow L \rightarrow b$	(L	M
$b \leftrightarrow M \leftrightarrow b$	bMb	Z

Prosedur parsing sampai di simbol awal (Z). Maka kalimat “b(aa)b” memang dapat diderivasi dari simbol awal Z dengan menggunakan grammar G.

### LAPORAN PENDAHULUAN

1. Jelaskan tentang CFG !
2. Sebutkan dan jelaskan metode parsing yang kamu ketahui !

### LAPORAN AKHIR

1. Jelaskan bentuk umum dari produksi CFG !
2. Jelaskan dan beri contoh metode parsing Top-Down dan Bottom-Up !