Penggunaan Grammar atau Tata Bahasa Dengan Metode Parsing Menggunakan Analisis Sintaktik Dalam Tehnik Kompilasi

Linda Marlinda, S.Kom, M.M

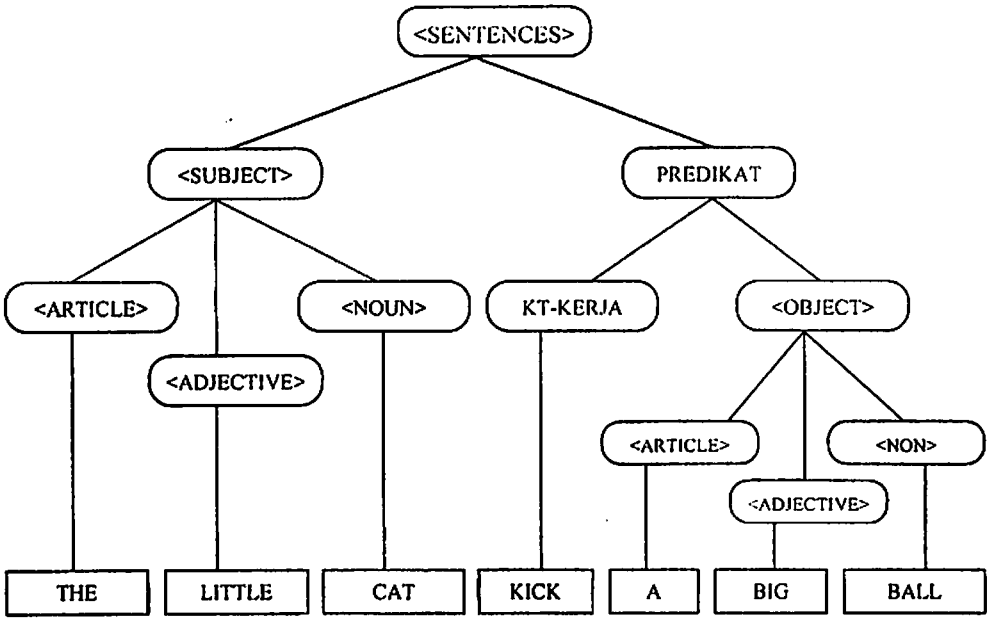
Abstrak. *Makalah ini menjelaskan tentang penggunaan metode parsing dengan menggunakan analisis sintak dalam tehnik kompilasi, yang bertugas mengubah program yang di tulis dalam bahasa sumber menjadi bahasa sasaran. Yang dimaksud dengan parsing dari sebuah kalimat adalah konstruksi atau pembentukan pohon sintaks untuk kalimat tersebut.Pohon sintaks untuk sebuah kalimat dari banyak bahasa selalu mempunyai sebuah simpul akar(root) yang diberi label sesuai simbol Start dari Grammarnya.Simpul daun dari pohon sintaks menyajikan simbol terminal dalam kalimat yang telah diuraikan. Semua simpul bukan daun menyajikan simbol nonterminal.*

Setiap simpul nonterminal mempunyai sejumlah cabang ke bawah yang berasal dari ruas kanan dari baris produksi yang dipergunakan.

oooOooo

Penggunaan Grammar atau Tata Bahasa Dengan Metode Parsing Menggunakan Analisis Sintaktik Dalam Tehnik Kompilasi

Linda Marlinda, S.Kom, M.M



Gambar 2

Disini tampak bahwa sintaks kalimat dalam bahasa Indonesia dibandingkan dengan bahasa Inggris agak berbeda. Tentunya salah, bila menulis SI KECIL KUCING MENENDANG BOLA BESAR, misalnya.

Dalam pembicaraan ini, pembahasan ditekankan pada sintaks kalimat suatu bahasa, bukan arti kalimat (semantik). Studi mengenai teori bahasa ini dikembangkan oleh banyak tokoh terkemuka, diantaranya NOAM CHOMSKY, JOHN BACKUS, PETER NAUR dan lain-lain lagi. Hal ini terutama untuk kebutuhan bahasa pemrograman computer.

Sekarang pembahasan dilakukan dengan lebih formal. Disini <KALIMAT>, <SUBJEK>, <PREDIKAT>, <OBJEK>, <KATA SANDANG>, <KATA BENDA>, <KATA KEADAAN> dan <KATA KERJA>disebut Kelas Sintaks, atau variabel, atau nonterminal. Disini menggunakan tanda < > untuk membedakannya dengan pengertian sehari-hari.

Sementara itu SI, KUCING, KECIL, MENENDANG, BOLA, dan BESAR disebut terminal atau token. Seluruh variabel harus dapat diuraikan/dibagi-bagi, sedangkan terminal tidak dapat diuraikan lagi. Dalam pohon sintaks, variabel disajikan sebagai simpul bukan daun, sedangkan terminal disajikan sebagai simpul daun. Diatas, simpul bukan daun digambarkan dengan lingkaran/elips, sedangkan daun dengan persegi panjang. Derivasi untai pertama diatas adalah :

<KALIMAT> → <SUBJEK><PREDIKAT>
→ <KATA SANDANG> <KT-BENDA> <KT-KEADAAN> <PREDIKAT>
→ <KATA SANDANG><KT-BENDA><KT-KEADAAN> <KT-KERJA> <OBJEK>

PENDAHULUAN.

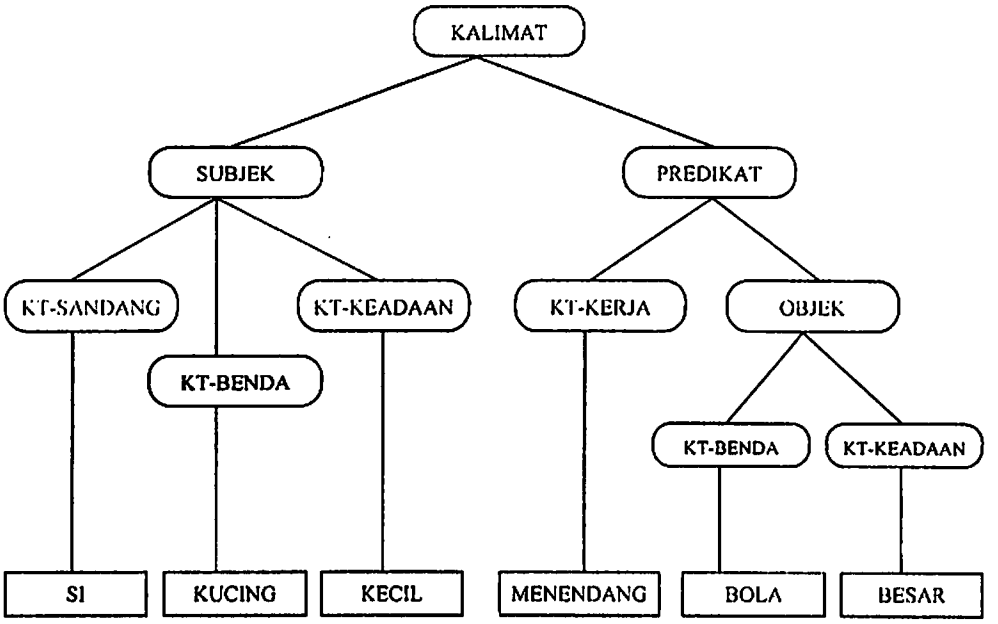
Analisa Sintaktik

Sintak adalah susunan kalimat dan aturan – aturan dalam membentuk kalimat disebut Gramar. Penganalisis sintak dalam bidang kompilasi sering disebut dengan Parser. Untuk menganalisa kalimat biasanya menggunakan bantuan parse-tree.

Contoh :

SI KUCING KECIL MENENDANG BOLA BESAR

Gambar penguraian kalimat diatas membentuk struktur pohon, yang disebut pohon sintaks dari kalimat. Disini kalimat dibagi-bagi berdasarkan jenis dan fungsi kata. Dari pelajaran bahasa Indonesia mengetahui bahwa kalimat diatas merupakan kalimat yang telah benar susunannya, atau telah benar tata bahasanya.



Gambar 1

Tetapi apakah susunan seperti diatas benar untuk kalimat dalam bahasa Inggris? Kalau kalimat diatas kita tulis dalam bahasa Inggris, dapat diperoleh sebuah sentences :

THE LITTLE CAT KICKS A BIG BALL

Pohon sintaknya akan berbeda, yakni :

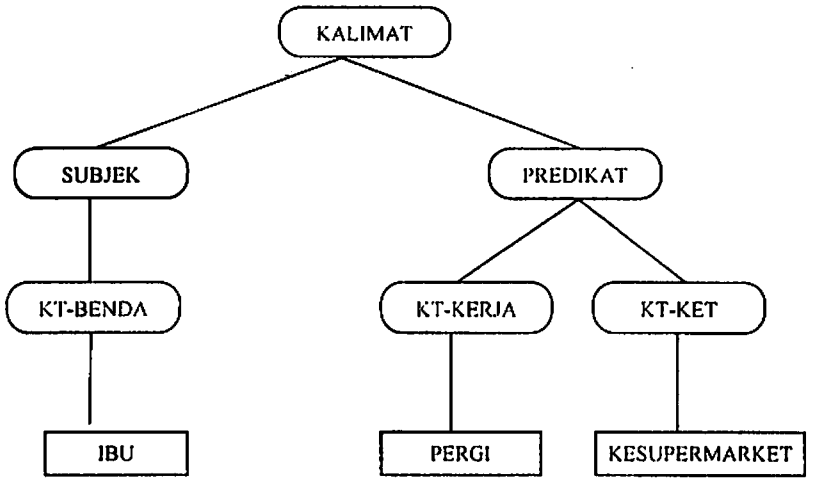
Himpunan Produksi :

<kalimat> → <Subjec><Predikat>
<Subjec> → <Ket.Keadaan><Kata Benda>
<Predikat> → <Kata Keja><Ket.Keadaan>
<Objec> → <Kata Benda><Ket.Keadaan>
<Ket. Keadaan> → Seekor
<Kata Benda> → Monyet | Pohon
<Kata Kerja> → Memanjat
<Kata Keterangan> → Kelapa

Derivasi :

Kalimat → <Subjec><Predikat>
→ <Ket.Keadaan><Kata Benda><Predikat>
→ <Ket.Keadaan><Kata Benda><Kata Kerja><Objec>
→ <Kaet.Keadaan> <Kata Benda> <Kata Kerja> <Kata Benda>
→ <Kata.keterangan>
→ Seekor Monyet Memanjat Pohon Kelapa

B. **Ibu Pergi Kesupermarket**



Himpunan Produksi :

<kalimat> → <subjec><predikat>
<subjec> → <Kata benda>
<predikat> → <kata kerja><kata keterangan>
<kata benda> → Ibu
<kata kerja> → Pergi
<kata keterangan> → KeSupermarket

→ <KATASANDANG><KT-BENDA><KT-KEADAAN>
<KT-KERJA><KT-BENDA><KT-KEADAAN>
→ SI KUCING KECIL MENENDANG BOLA BESAR

Contoh kedua berikut ini adalah contoh bagaimana membangun operand suatu bahasa pemrograman.

Start adalah <OPERAND> sebagai berikut :

<OPERAND> → <ID> | <INTEGER>
<ID> → <LETTER><LIST>
<LETTER> → x | y | z
<LIST> → <LETTER><LIST> | <DIGIT><LIST> | ^
<DIGIT> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<INTEGER> → <SIGN><DIGIT><DIT>
<SIGN> → + | -
<DIT> → <DIGIT><DIT> | ^

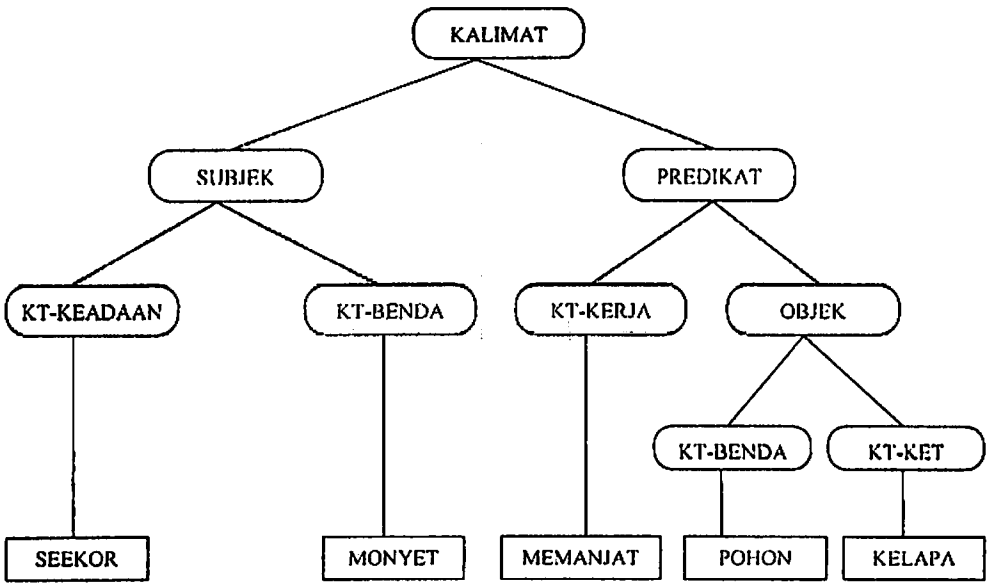
(Catatan : ^ adalah terminal / token hampa)

Sebagai contoh kita lakukan derivasi

<OPERAND> → <ID>
→ <LETTER><LIST>
→ y<LIST>
→ y<DIGIT><LIST>
→ y7<LIST>
→ y7^ atau y7

Contoh lain yang lebih ringkas,

A. **Seekor Monyet Memanjat Pohon Kelapa**



megidentifikasi kalimatmenggunakan aturan Chomsky, yaitu non-terminal dinyatakan dengan huruf besar (A, B, C, dst), terminal dinyatakan dengan huruf kecil bagian awal (a, b, c, dst.), sederet terminal dinyatakan dengan huruf kecil bagian akhir (w, x, y, dst.), dan campuran antara non-terminal dan terminal (sentential form) dinyatakan dengan huruf latin (α, β, γ,.....).

Cara yang untuk mendefenisikan kalimat adalah dengan BNF (Backus Nour Form). Aturan ini digunakan grammar yang yang bersifat Context-free. Simbol – simbol yang digunakan adalah:

Non terminal ditulis dalam tanda <>, tanda := sama dengan tanda panah, pengulangan dinyatakan dalam tanda { }, tanda [ ] menyatakan pilihan (optional) bila ada beberapa produksi dengan bagian kiri yang sama, seperti misalnya :

<expression> ::=<expression>+<expression>
<expression> ::=(<expression>)
<expression> ::= i dimana i = identifier = terminal symbol
Dapat ditulis :
<expression> ::=<expression>+<expression> | <expression> | i

Contoh lain dalam Turbo Pascal : Deklarasi variabel dengan var bila dinyatakan dalam BNF menjadi seperti berikut ini:
<integer-constant> ::=[+ | -] <digit> {<digit>}
Parse Tree dan Penurunannya

Contoh : Grammar sederhana untuk expression :
GE = (T,N,S,R),

Di mana T = { i,+,-,*,/,(,)},

N={E}
S=E

R={ E → E + E ,
E → E – E ,
E → E * E ,
E → E / E ,
E → (E) ,
E → i };

Cara penulisan grammar tersebut adalah cara penulisan yang lengkap menurut Chomsky, untuk selanjutnya hanya dituliskan bagian produksi nya saja.

Dengan menggunakan grammar di atas, dianalisis ekspresi :

(a + b) / (a – b)

Dari penganalisis leksikal diperoleh ekspresi yang berbentuk token (i + i) / (i- i). Kita mengenal ekspresi diatas adalah pembagian, dengan pembilang (i+i) dan penyebut (i-i). Oleh karena itu produksi yang cocok adalah E → E / E

DERIVASI :

Kalimat → <Subjec><Predikat>
→ <Kata Benda><Predikat>
→ <Kata Benda><Kata Kerja><Kata Keterangan>
→ Ibu Pergi Kesupermarket

a. 4 x y 2

<Operand> → <Digit><List>
→ 4 <Letter><List>
→ 4 x <Letter><List>
→ 4 x y <Digit><List>
→ 4 x y 2 <List>
→ 4 x y 2 ^
→ 4 x y 2

b. – 21 x

<Operand>→ <integer>
→ <Sign><Digit><Dit>
→ <Digit><Dit>
→ 2 <Dit>
→ 2 <Digit><Dit>
→ 2 1 ID <Letter><List>
→ 2 1 x

c. x – y + 2

<operand> → <integer><id><integer><id>
→ <sign><digit><dit><id><integer><id>
→ +<digit><dit><id><integer><id>
→ +0<dit><id><integer><id>
→ + ^ <id><integer><id>
→ +<id><integer><id>
→ - <sign>

Defenisi Formal dari Grammar

Grammar (G) merupakan fugsi dari (T,N,S,R), masing-masing adalah :

T : himpunan terminal symbol dog, gnawed, a, +, ...
N : himpunan non-terminal symbol
<senence>, <expression>, ...
N dan T adalah himpunan disdjoint (tak bersinggung)
S : simbol awal (starting symbol) yang unik <sentence>
S & N
R : himpunan produksi dengan bentuk $\alpha \rightarrow \beta$

α dan β adalah kumpulan non-terminal Dan terminal symbol. Produksi adalah setiap kejadian di mana string bagian kiri (α) dapat digantikan oleh string bagian kanan (β). Context-Free grammar adalah grammar yang berbentuk $\alpha \rightarrow \beta$ dimana α adalah non-terminal tunggal. Lambang – lambang yang digunakan dalam

Parse tree diatas dapat di nyatakan dalam bentuk turunan (derivation)
Dengan notasi α1 => α2 sebagai berikut :

Turunan (rightmost derivation)	Produksi
E => E / E	[dengan produksi E → E/E]
=> E/(E)	[dengan produksi E → (E)]
=> E/(E-E)	[dengan produksi E → (E-E)]
=> E/(E-i)	[dengan produksi E → i]
=> E/(i-i)	[dengan produksi E → i]
=> (E)/(i-i)	[dengan produksi E → (E)]
=> (E+E)/(i-i)	[dengan produksi E → (E+E)]
=> E+i(i-i)	[dengan produksi E → i]
=> (i+i)/(i-i)	[dengan produksi E → i]

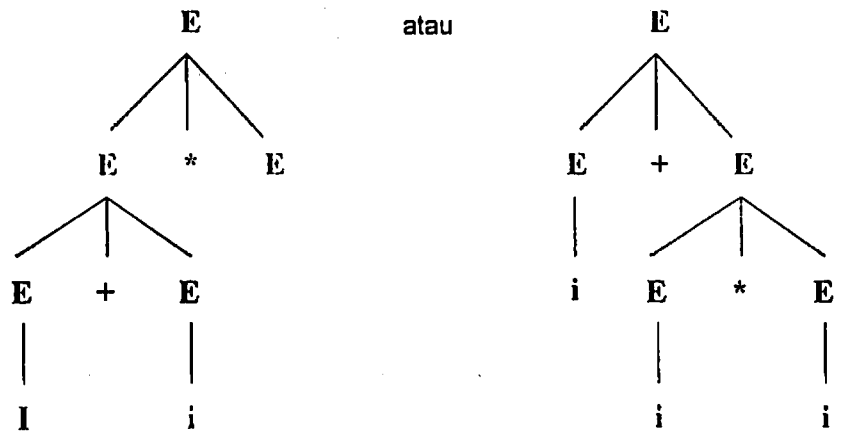
$L(G) = \{ w \mid S \overset{*}{\underset{G}{\Rightarrow}} W \}$ Bahasa (Language) dengan grammar G adalah himpunan semua string terminal yang dapat diturunkan dari S menggunakan aturan G.

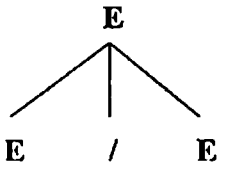
Penurunan seperti di atas dilakukan dengan penurunan kanan atau penurunan kiri

Ambiguous Grammar

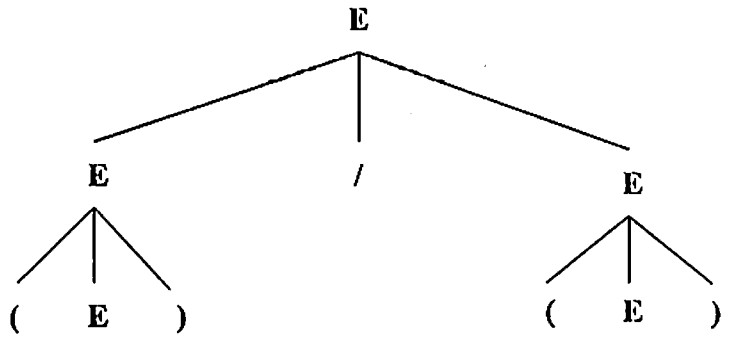
Sebuah kalimat yang dihasilkan oleh sebuah Grammar adalah Ambiguous, jika terdapat lebih dari satu pohon sintaks yang dapat dipakai untuk membentuk kalimat tersebut. Sebuah Grammar adalah Ambiguous, jika menghasilkan setidaknya satu kalimat Ambiguous. Selain itu Grammar disebut Unambiguous.
Jika ada beberapa Grammar berbeda dapat menghasilkan bahasa yang sama, dimana ada Grammar yang Ambiguous dan lainnya Unambiguous, maka bahasa tersebut dikatakan Inheretently Ambiguous.

Perhatikan ekspresi i+i *i. Ekspresi tersebut mengandung 2 macam arti, yaitu :

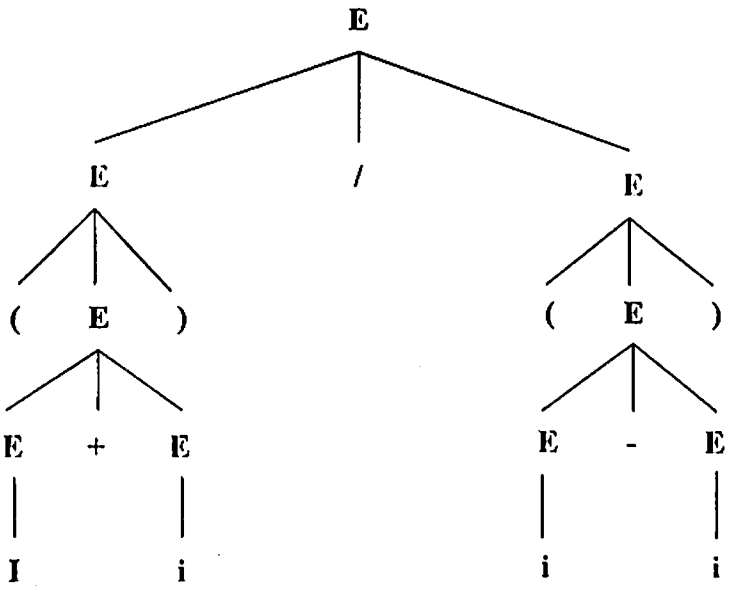




Perhatian bahwa sisi kiri menjadi induk dan bagian kanan menjadi anak dari parse-tree. Selanjutnya, karena pembilang dan penyebut adalah ekspresi bertanda kurung maka produksi untuk E adalah E→(E):



Ekspresi E yang ada didalam kurung pertama adalah penjumlahan, maka kita gunakan produksi E→E + E dan E yang kedua adalah pengurangan sehingga digunakan produksi E→E – E, dan E yang terakhir menghasilkan i sehingga digunakan produksi E→i.



satu cara untuk mengaplikasikan Bahasa bebas konteks untuk melakukan analisa sintaksis suatu program sumber. Pada parser ini, simbol terminal maupun non terminalnya bukan sebuah karakter, tetapi berupa besaran leksik sebagai simbol terminalnya dan besaran sintaks sebagai simbol non terminalnya.
Ciri utama dari parser ini adalah secara rekursif menurunkan semua simbol non terminal dari awal sampai bertemu simbol terminal dan tidak pernah mengambil token secara mundur (*no backtrack*).
Ciri lainnya adalah bahwa recursive descent parser sangat tergantung dari algoritma scan dalam mengambil token.

Kesimpulan

Dalam tahap analisis sintak, operasi – operasi yang dilakukan oleh program sumber ditentukan dan dicatat dalam suatu struktur pohon (tree) yang disebut dengan nama pohon sintak atau syntax tree. Dalam hal setiap node tersebut memberikan rgumen yang diperlukan, sehingga menggambarkan dari suatu operasi pemberian nilai dari suatu variabel.

Daftar Pustaka

Soediyono, Eko, 2005, "Tehnik Kompilasi teori dan praktek", Andi Offset , Yogyakarta.

Suhartanto, Heru, 1992, "Tehnik Kompilasi", Elex Media komputindo, Jakarta.

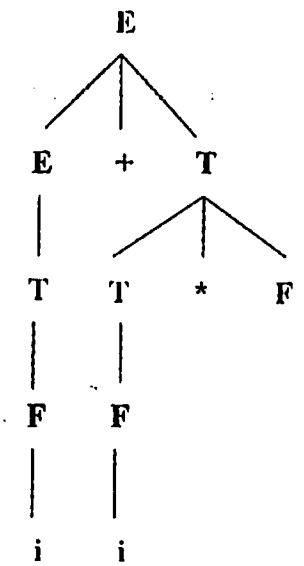
Suryadi, D, " Pengantar Automata bahasa formal dan kompilasi", Gunadarma, Jakarta.

oooOooo

Dari kedua gambar diatas, mana yang benar ?
Untuk mengatasi ambuguitas maka digunakan aturan tambahan yaitu dengan menggunakan aturan operator presdence dalam pascal, atau grammarnya ditulis ulang agar tidak terjadi ambiguitas

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

E menyatakan ekspresi, T menyatakan Term dan F menyatakan Faktor.



Ada dua metode dasar parsing, yaitu :

1. ***Top-down parsing***
   konstruksi pohon sintaks dimulai dari akar(root) dan dilanjutkan turun kebawah menuju daun
   Cara kerja :
   pada setiap urutan langkah, nonterminal paling kiri α dari bentuk sentensial Q1 α Q2, diganti dengan bagian kanan dari produksi $\alpha \rightarrow \beta$ untuk menghasilkan bentuk sentensial berikutnya.
   Contoh : G =({I, L, D}, {a,b, . .z,0,...9}, I, Q) dgn Q :
   $I \rightarrow L \mid IL \mid ID$
   $L \rightarrow a|b|c \ldots z$
   $D \rightarrow 0|1|2 \ldots 9$

2. ***Buttom-up parsing***
   konstruksi pohon sintaks dimulai dari daun bergerak keatas menuju akar (root) atau denga menggunakan Recursive descent Parser, adalah salah