BAB II
SINTAKS

2.1. SINTAKS

Sintaks merupakan kumpulan aturan yang mendefinisikan suatu bentuk bahasa. Sintaks mendefinisikan bagaimana suatu kalimat dibentuk sebagai barisan/urutan dari pemilihan suatu kata dasar. Kata bukan merupakan sesuatu yang mendasar. Kata dikonstruksikan dengan karakter-karakter alfabet. Dengan menggunakan aturan ini maka suatu kalimat dapat dikatakan legal atau tidak legal. Sebagai contoh, dalam *keyword* bahasa C (seperti while, do, if, dan else), identifier, angka, operator, dan seterusnya, merupakan kata dalam suatu bahasa. Sintaks dalam bahasa C mengatur cara mengombinasikan kata-kata tersebut ke dalam suatu statemen dengan bentuk yang benar sehingga dapat disusun suatu program yang dapat berjalan dengan benar. Sintaks tidak mengerti apapun tentang isi atau arti dari suatu kalimat, aturan semantik yang bisa mengartikannya.

Sintaks dari bahsa pemrograman didefinisikan dengan dua kumpulan aturan, yaitu aturan ***lexical*** dan aturan ***syntactic***. Aturan leksikal menspesifikasikan kumpulan karakter yang terdapat dalam alfabet dari bahasa dan cara supaya karakter-karakter tersebut dikombinasikan ke dalam kata-kata yang valid dan dapat diterima. Sebagai contoh adalah bahasa Pascal yang tidak memperhatikan huruf besar dan huruf kecil, sedangkan bahasa C dan Ada memperhatikan dan membedakan hal tersebut.

Sintaks berfungsi menyediakan bentuk-bentuk notasi untuk komunikasi antar *programmer* dan pemroses bahasa pemrograman sehingga dapat mempermudah pembuatan suatu program.

Suatu bahasa pemrograman juga dibangun berdasarkan elemen-elemen *syntactic*, yang dapat membentuk suatu statemen-statemen dalam bahasa pemrograman. Elemen-elemen tersebut antara lain:

a. Himpunan karakter
   Himpunan karakter yang sering dijumpai dan digunakan adalah himpunan karakter ASCII, yang berisi dasar karakter-karakter angka dan huruf. Karakter ASCII secara umum dapat langsung digunakan pada kebanyakan perangkat I/O. namun ada bahasa pemrograman yang tidak menggunakan himpunan karakter ASCII, contohnya bahasa APL. APL menggunakan karakter yang tidak dapat digunakan secara langsung pada kebanyakan pernagkat I/O, tidak seperti bahasa C yang menggunakan himpunan karakter ASCII.

b. Identifier
   Sintaks dasar dari *identifier* yang sering digunakan adalah string dari huruf dan angka yang dimulai dengan huruf. Namun, banyak juga variasi dari bahasa pemrograman yang menggunakan suatu *identifier* dengan tanda "." Atau "-". Hal ini akan berpengaruh pada kemudahan dalam pembacaan program.

c. Simbol untuk operator
   Kebanyakan bahasa pemrograman menggunakan karakter "+" dan "-" untuk merepresentasikan dua buah operasi dasar aritmetika, dan menggunakan kombinasi serta memanfaatkan karakter-karakter spesial untuk suatu *operator*, atau menggunakan string untuk operator seperti

pada FORTRAN dengan .EQ. untuk membandingkan kesamaan dan ** untuk perpangkatan.

d. Keyword dan reserved word
Keyword merupakan merupakan suatu *identifier* yang digunakan sebagai bagian tetap dari sintaks suatu statemen, misalnya **IF** untuk memulai suatu statemen kondisi atau **DO** untuk memulai suatu perulangan dalam bahasa FORTRAN. *Keyword* merupakan *reserved word* jika tidak digunakan sebagai *identifier* yang dipilih oleh *programmer*. Misalnya dalam bahasa FORTRAN, identifier **IF** dan **DO** dapat dipilih sebagai nama variabel oleh programmer sehingga suatu statemen yang dimulai dengan IF belum tentu merupakan statemen kondisi.

e. Noise word
Merupakan kata pilihan yang disisipkan dalam statemen untuk meningkatkan *readibility*. Sebagai contoh adalah statemen **GO TO** *label*. **GO** merupakan *keyword* yang harus ada, sedangkan **TO** merupakan optional yang akan meningkatkan *readibility*.

f. Komentar
Penambahan komentar dalam suatu program merupakan hal penting dari dokumentasi suatu program. Bahasa pemrograman mengijinkan komentar dalam beberapa bentuk.
- Baris komentar yang terpisah di dalam program seperti pada FORTRAN.
- Penggunaan karakter khusus yang tidak memedulikan baris seperti /* dan */ di bahasa C.
- Dimulai dari sembarang tempat disuatu baris dengan diawali suatu karakter khusus, seperti "-" di Ada, "//" di C++ atau "!" di FORTRAN.

g. Blank
Aturan penggunaan spasi pada bahasa pemrograman sangat beragam. Sebagai contoh adalah bahasa C. Biasanya spasi diabaikan, kecuali jika ada simbol "=+" yang merupakan operator tunggal. Bila dipisahkan dengan spasi maka akan terjadi kesalahan sintaks.

h. Delimiter dan tanda kurung
Delimiter merupakan elemen syntactic yang digunakan untuk menandai suatu awalan atau akhiran dari suatu syntactic unit seperti statemen atau ekspresi. Tanda kurung biasanya berpasangan dengan delimiter, misalnya kurung kurawal atau pasangan kata begin … end. Delimiter berguna untuk meningkatkan readability suatu program dan juga dapat menghilangkan ambiguitas suatu statemen karena dapat digunakan secara eksplisit untuk memisahkan statemen-statemen yang mirip.

i. Ekspresi
Merupakan suatu fungsi yang mengakses data dalam suatu program dan mengembalikan suatu nilai. Ekspresi merupakan dasar dari blok syntactic dari statemen yang dibangun.

2.2. TATA BAHASA (GRAMMAR)

Tata bahasa dalam bahasa pemrograman merupakan suatu kumpulan aturan (disebut *production*) yang menentukan urutan-urutan karakter (*lexical token*). Suatu tata bahasa secara alamiah menerangkan struktur hirarki dan banyak membentuk bahasa pemrograman.Misalnya perintah if-then-else:

If ekspresi **then** perintah **else** perintah

Dalam hal ini, suatu perintah adalah gabungan dari kata kunci **if**, ekspresi, kata kunci **then,** perintah, kata kunci **else** dan perintah lainnya. Bila digunakan nama variabel *expr* untuk menyatakan suatu ekspresi dan variabel *stmt* untuk menyatakan suatu perintah, maka struktur aturan ini dapat dinyatakan sebagai

stmt → **if** *expr* **then** *stmt* **else** *stmt*

dimana tanda panah di atas dibaca sebagai "dapat berbentuk sesuatu". Aturan seperti ini disebut juga suatu produksi (production).

Tata Bahasa G didefinisikan sebagai pasangan 4 tuple yaitu V, T, P dan S, yang dituliskan: G = (V,T,P,S), dimana:

V : Himpunan simbol-simbol terminal
T : Himpunan simbol-simbol non-terminal
P : Himpunan produksi
S : Simbol awal

2.2.1. Unsur Pembentuk Tata Bahasa

a. **Terminal** merupakan sombol dasar dari suatu rangkaian yang terbentuk. Kata 'token' merupakan persamaan dari 'terminal' jika kita berbicara dalam bahasa pemrograman. Kata kunci **if**, **then**, **else** adalah terminal.
b. **Non-Terminal** adalah variabel sintaktik yang menyatakan kumpulan dari rangkaian. variabel *stmt* dan *expr* adalah non-terminal. Non-terminal mendefinisikan kumpulan dari rangkaian yang membantu bahasa yg dibentuk oleh tata bahasanya. Non-terminal juga memberikan struktur hirarki pada suatu bahasa yang sangat bermanfaat dalam proses analisis sintak dan translasi.
c. Dalam suatu tata bahasa, satu non-termnal berfungsi sebagai **simbol awal**, dan kumpulan rangkaian yang dinyatakannya merupakan bahasa yang didefinisikan oleh tata bahasa itu.
d. **Produksi**-produksi dalam satu tata bahasa menentukan perilaku dimana terminal dan non-terminal dapat digabungkan untuk membentuk rangkaian. setiap produksi terdiri dari non-terminal, diikuti oleh tanda panah (simbol → atau simbol ::=), lalu diikuti oleh rangkaian dari suatu non-terminal dan terminal.

Dalam tata bahasa, anggota alfabet dinamakan simbol terminal atau token, kalimat adalah deretan hingga simbol-simbol terminal, sedangkan bahasa adalah himpunan kalimat-kalimat.

Ketentuan notasi

a. Simbol-simbol berikut adalah terminal:
   - Huruf kecil awal alfabet seperti a,b dan c
   - Simbol operator seperti +, -, dan sebagainya.
   - Simbol tanda baca seperti tanda kurung, koma dan sebagainya.
   - Digit 0,1,2, …, 9
   - Rangkaian tercetak tebal seperti **if**, **then**, **else**
b. Simbol-simbol berikut adalah non-terminal:
   - Huruf besar awal alfabet seperti A, B, dan C.
   - Huruf S, jika muncul biasanya dianggap sebagai simbol awal
   - Nama dengan huruf kecil seperti expr atau stmt.

c. Huruf besar akhir alfabet seperti X, Y, Z merepresentasikan simbol tata bahasa (mungkin terminal atau non-terminal)
d. Huruf kecil akhir suatu alfabet seperti u,v,…z merupakan rangkaian dari suatu terminal.
e. Huruf kecil Yunani seperti α, β, γ sebagai contoh merupakan rangkaian dari simbol tata bahasa. Maka suatu produksi seperti A → α, menyatakan ada satu non-terminal A disebelah kiri tanda panah dan rangkaian dari simbol tata bahasa α disebelah kanan tanda panah (sebelah kanan produksi)
f. Jika A → α_1, A → α_2, …, A → α_k adalah semua produksi dengan A disebelah kiri (A-produksi), dapat ditulis A → α_1 | α_2 | α_k . α_1 , α_2 , …, α_k sebagai alternatif untuk A.
g. Jika tidak disebutkan, sisi kiri dari produksi yang pertama adalah simbol awal.

Bentuk umum produksi dari tata bahasa bebas konteks (*Context-Free Grammar*) adalah:

$$\alpha \rightarrow \beta,\ \alpha \in V,\ \beta \in (V|T)$$

Contoh 2.1.
Berikut ini adalah tata bahasa "barisan dari angka-angka yang dipisahkan oleh tanda plus atau minus".

list → *list* + *digit*
list → *list* – *digit*
list → *digit*
digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Bagian kanan dari produksi dengan unsur non-terminal *list* di bagian kiri dapat dikelompokkan menjadi satu produksi yang setara yaitu:

list → *list* + *digit* | *list* – *digit* | *digit*

Token dari tata bahasa adalah simbol + - 0 1 2 3 4 5 6 7 8 9, sedangkan unsur non-terminal adalah nama-nama yang digaris miring seperti list dan digit, dimana list adalah unsur terminal awal karena merupakan produksi yang pertama.

Suatu produksi dikatakan produksi untuk unsur non-terminal bila unsur non-terminal timbul di bagian kiri dari produksi, sedangkan barisan token adalah barisan dari nol atau lebih token. Unsur yang mengandung nol token ditulis sebagai ε, dan disebut dengan nama barisan kosong.

Suatu bahasa diperoleh dari barisan-barisan yang dimulai dari simbol awal dan mengganti unsur non-terminal dengan bagian kanan dari produksi non-terminal. Barisan token yang dapat diperoleh dari simbol awal membentuk apa yang disebut bahasa yang didefinisikan oleh tata bahasa yang bersangkutan.

Contoh 2.2.
G_0 adalah tata bahasa untuk "fragment of English"
Kategori gramatikal : S, NP, VP, D, N, V.
Kata-kata : a, the, cat, mouse, ball, boy, girl, ran, bounced, caught

Tata Bahasa :

S → NP VP
NP → N
NP → D N
VP → V
VP → V NP
V → ran | bounced | caught
D → a | the
N → cat | mouse | ball | boy | girl

Kategori yang paling atas adalah S yang membentuk sebuah kalimat.

Dalam tata bahasa bebas konteks, kategori gramatikal disebut variabel atau non terminal, kata-kata disebut token atau terminal, tata bahasa yang menurunkan tata bahasa lain disebut dengan produksi dan kategori yang paling atas disebut simbol awal.

2.2.2. Derivasi

Derivasi adalah cara untuk melihat bagaimana suatu tata bahasa mendefinisikan suatu bahasa. Dengan menggunakan contoh 2.2, sebuah kalimat "the cat caught the mouse" dapat diderivasi sebagai berikut:

S → NP VP
→ D N VP
→ the N VP
→ the cat VP
→ the cat V NP
→ the cat caught NP
→ the cat caught D N
→ the cat caught the N
→ the cat caught the mouse

Derivasi seperti di atas disebut derivasi paling-kiri (*leftmost manner*).

2.3. PENGURAIAN (PARSING)

Tatabahasa dapat digunakan baik untuk penurunan (generation) atau penguraian (parsing) sebuah kalimat. Baik penurunan (generation) maupun penguraian (parsing) membutuhkan urutan dan aturan penulisan sebuah aplikasi yang dimulai dari simbol awal tatabahasa dan diakhiri dengan kalimat.

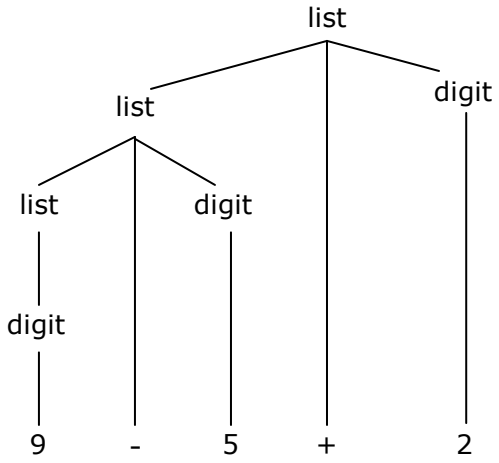
Penguraian adalah suatu proses untuk menentukan apakah suatu rangkaian dari token yang dihasilkan analisis leksikal termasuk dalam suatu tata bahasa tertentu. Ada dua metode penguraian yaitu penguraian puncak-ke-bawah (*top-down*) dan dasar-ke-atas (*bottom-up*).

2.3.1. Penguraian Puncak-ke-bawah (Top-Down)

Penguraian puncak-ke-bawah dapat dipandang sebagai suatu usaha untuk mencari derivasi paling kiri (*leftmost*) dari suatu rangkaian masukan. Bila diberikan kalimat x sebagai input, maka penguraian akan 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 pengurai jika dibaca dari kiri ke kanan.

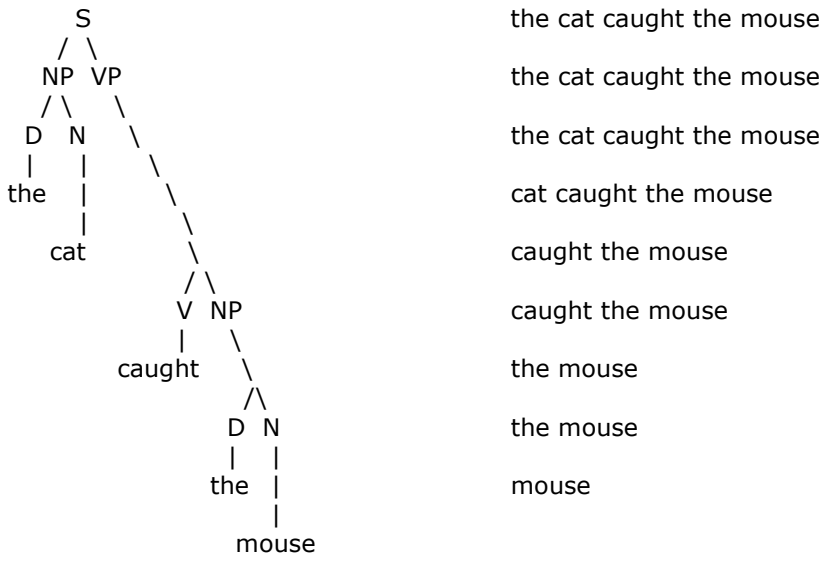
Semua produksi pada contoh 2.1. adalah produksi-produksi yang diperlukan untuk dapat mendefinisikan bahasa yang diinginkan. Sebagai contoh suatu ekspresi 9-5+2 merupakan salah satu anggota dari bahasa tersebut atau suatu *list* juga. Hal ini ditunjukkan sebagai berikut:

a. 9 adalah *list* karena 9 adalah suatu angka dari produksi *list* → *digit*
b. 9-5 adalah *list* karena 9 dan 5 adalah *list* yang dihubungkan dengan tanda minus (-).
c. 9-5+2 adalah *list* karena 9-5+2 adaah *list* yang dihubungkan dengan tanda plus (+).



Gambar 2.1. Pohon urai puncak-ke-bawah dari ekspresi 9-5+2

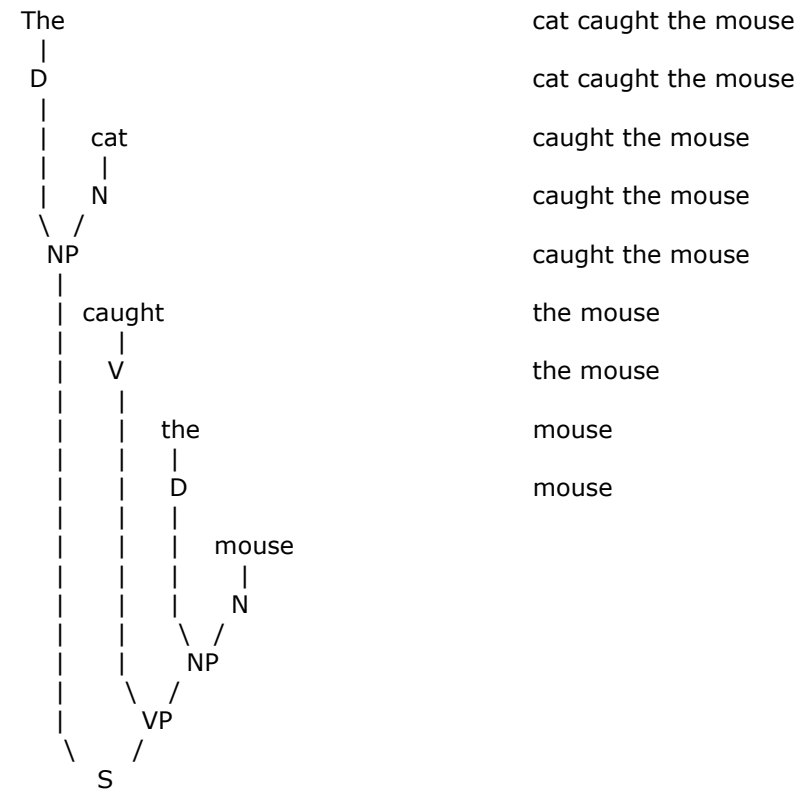
Penguraian puncak-ke-bawah untuk contoh 2.2 adalah:



Gambar 2.2. Penguraian puncak-ke-bawah dari contoh 2.2

2.3.1. Penguraian Dasar-ke-Atas (Bottom-Up)

Bila diberikan kalimat x sebagai input, maka penguraian dimulai dari kalimat x yang nyata dari pembacaan semua *leaf* pohon pengurai dari kiri ke kanan sampai tiba disimbol awal S (atau tidak sampai di S jika kalimat x memang tidak bisa diturunkan dari S.
Penguraian dasar-ke-atas untuk contoh 2.2:

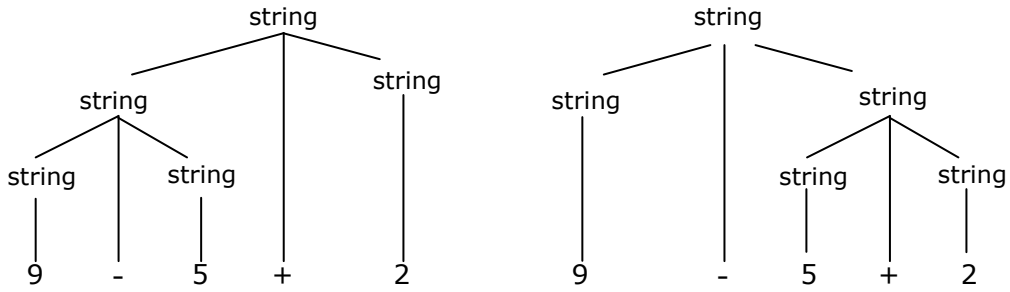


Gambar 2.3. Penguraian dasar-ke-atas dari contoh 2.2

2.4. TATA BAHASA BERARTI GANDA (AMBIGUOUS)

Suatu tata bahasa yang menghasilkan lebih dari satu pohon urai disebut ambiguous. Dengan perkataan lain, tata bahasa yang ambiguous adalah tata bahasa yang menghasilkan lebih dari satu derivasi leftmost maupun rightmost untuk kalimat yang sama. Untuk pengurai jenis tertentu, sebaiknya tata bahasa itu dibuat tidak ambiguous, sebab kalau tidak kita tidak dapat menentukan secara unik pohon urai untuk memilih suatu kalimat.
Contoh 2.3
Misalkan tidak dibedakan antara angka dan list pada contoh 2.1, maka tata bahasa yang melibatkan angka dan tanda plus dan minus dapat dituliskan sebagai:

String → string + string | string – string | 0 | 1 | … | 9



Gambar 2.4. Dua pohon urai dari ekspresi 9-5+2

Kedua pohon urai diatas pada dasarnya berhubungan dengan dua cara pemberian tanda kurung pada ekspresi di atas yaitu (9-5)+2 dan 9-(5+2). Pada cara pemberian kurung yang kedua nilai ekspresi yang diperoleh adalah 2, sedangkan pada cara yang umumnya digunakan adalah cara pemberian kurung yang pertama dan nilai yang diperoleh adalah 6.