PEMERIKSAAN TATA BAHASA DAN MAKNA KATA DALAM KALIMAT BAHASA INGGRIS MENGGUNAKAN ALGORITMA *LEFT CORNER PARSING*

TUGAS AKHIR

Disusun oleh :

SEGER SUPRAPTO
NRP. 2694 100 035

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2000

PEMERIKSAAN TATA BAHASA DAN MAKNA KATA DALAM KALIMAT BAHASA INGGRIS MENGGUNAKAN ALGORITMA *LEFT CORNER PARSING*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember Surabaya

Mengetahui / Menyetujui.

Pembimbing I	Pembimbing II
Ir. Esther Hanaya, MSc.	**Ir. Aris Tjahyanto M.Kom**
NIP. 130 816 212	NIP. 131 933 299

SURABAYA
2000

KATA PENGANTAR

Syukur alhamdulillah, segala puja dan puji syukur Penulis panjatkan kepada Allah SWT. atas limpahan rahmat dan inayah-Nya sehingga Penulis dapat menyelesaikan tugas akhir ini dengan baik.

Berbagai hambatan, keragu-raguan dan tantangan betul – betul Penulis alami sepanjang penyusunan tugas akhir ini. Akan tetapi berkat dorongan semangat, motivasi, dan bantuan dari berbagai pihak akhirnya Penulis mampu menyelesaikan tugas akhir ini dengan baik, lancar dan memuaskan.

Buku tugas akhir ini Penulis beri judul :

PEMERIKSAAN MAKNA KATA DAN TATA BAHASA BAHASA INGGRIS DENGAN MENGGUNAKAN ALGORITMA *LEFT CORNER PARSING*

Tugas akhir ini disusun guna memenuhi sebagian persyaratan untuk memperoleh gelar sarjana pada Jurusan Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya.

Melalui kesempatan ini, Penulis menyampaikan terima kasih dan penghargaan yang sebesar-besarnya kepada :

1. Bapak Dr. Ir. Arif Djunaidy, selaku Ketua Jurusan Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, Surabaya.
2. Ibu Esther Hanaya, M.Sc. selaku dosen pembimbing dalam tugas akhir ini dan atas ilmu, pertimbangan dan nasehat yang telah banyak diberikan selama Penulis menyelesaikan masa perkuliahan.

3. Bapak Ir. Aris Tjahyanto, Mkom., selaku dosen pembimbing yang telah banyak memberikan arah dan dorongan atas penyelesaian tugas akhir ini.
4. Bapak Ir Muh. Husni MKom, selaku dosen wali Penulis.
5. Bapak dan Ibu, Adikku Siti, dan Suradi yang sangat saya cintai
6. Istriku, Lia Dwi Istanti
7. Terima kasih spesial kepada Joko, Yuli, dan Yudi selaku keluarga GL-13 yang telah banyak memberikan motivasinya
8. Rekan-rekan Madani, dan semua rekan se-*geng* C-0A yang tak dapat disebutkan satu persatu.
9. Rekan-rekan Limaxindo Intersistem yang senantiasa menjadi juara
10. Semua pihak yang telah membantu hingga terselesaikannya tugas akhir ini.

Penulis akan senang hati menerima saran dan kritik yang membangun. *Macan mati meninggalkan kulit, gajah mati meninggalkan gading*, demikian juga penulis berharap tugas akhir ini dapat bermanfaat setelah penulis meninggalkan kampus.

Surabaya, Juli 2000

Penulis

PEMERIKSAAN TATA BAHASA DAN MAKNA KATA DALAM KALIMAT BAHASA INGGRIS MENGGUNAKAN ALGORITMA *LEFT CORNER PARSING*

ABSTRAK

Dalam Bahasa Inggris, terdapat beberapa kata yang morfologinya sama, namun memiliki jenis dan makna yang berbeda dalam suatu kalimat. Sebuah kata dapat menjadi kata benda, kata sifat, kata kerja atau kata keterangan sehingga mempunyai lebih dari satu macam makna. Demikian juga dengan penggunaan kata manjemuk, yang dalam istilah Bahasa Inggris disebut dengan acronym, dapat juga menyebabkan makna yang lain dari makna dasar kata tersebut. Adanya kata-kata seperti ini dapat menyebabkan perbedaan interpretasi sehingga dapat menyimpangkan maksud kalimat tersebut yang sebenarnya.

Penggunaan kamus elektronik konvensional yang sekedar mengambil data terjemahan kata tidak akan dapat menyelesaikan permasalahan ini dengan baik. Hal ini karena di dalam kalimat, bisa jadi suatu kata benda dianggap kata kerja atau sebaliknya. Penyelesaian masalah ini dapat dilakukan dengan menganalisis makna yang memiliki tiga tahapan : sintaktik, semantik dan pragmatik.

Algoritma Left Corner parsing dalam penelitian Tugas Akhir ini digunakan untuk membantu menentukan maksud suatu kata dalam sebuah kalimat. Algoritma ini akan menelusuri pola tata bahasa yang dapat digunakan untuk memilih jenis yang paling sesuai pada kata tertentu. Kemudian dengan mengetahui jenis kata ini, akan ditentukan makna yang paling tepat untuk kata tersebut sesuai dengan basisdata yang tersedia.

Algoritma Left Corner parsing akan mengurai kalimat kata per kata penyusunnya secara bottom up dan top down secara parallel. Kata-kata ini disusun ke dalam tree berdasarkan kategori frase yang berlaku dalam tata Bahasa Inggris. Tree ini disusun berdasarkan pola kalimat tersebut dan diatur dengan mengggunakan CFG (Context Free Grammar) Bahasa Inggris. Kata-kata yang sudah umum seperti pronoun, determiner, preposition, atau auxiliary verb dijadikan bahan dasar untuk analisis kata yang lain. Dari tree yang diproses ini akan dapat diketahui jenis kata-kata yang menyusunnya, yang dapat dijadikan parameter untuk memilih makna yang sesuai dari dalam basis data.

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

I.1. Latar Belakang

Bahasa Inggris sebagai bahasa Internasional merupakan sarana komunikasi utama bagi manusia di segenap permukaan bumi. Oleh karena itu kemudahan dalam mempelajari, memahami dan menggunakan bahasa tersebut harus terus diupayakan agar hubungan antar bangsa semakin mudah dilakukan, tanpa harus kesulitan dengan tata bahasa yang dipakainya.

Pemahaman pada sebuah bahasa setidaknya harus melalui tiga tahap : tahap sintaktik, semantik dan pragmatik. Tahap sintaktik meliputi pemakaian dan penyusunan kata sehingga memenuhi persyaratan sebuah kalimat menurut aturan tata bahasa yang benar. Tahap semantik adalah untuk mengetahui arti kata-kata yang menyusun kalimat tersebut sesuai dengan penempatannya. Sedangkan tahap pragmatik adalah untuk memahami tujuan penyampaian isi kalimatnya.

Tahap-tahap inilah yang dilakukan oleh manusia untuk memahami maksud suatu kalimat, namun hanya dilakukan berdasarkan kemampuan interpretasi masing-masing. Memahami Bahasa Inggris dengan cara ini memiliki kelemahan pada tingkat kemampuan individu. Hal ini terkait dengan kemampuan masing-masing manusia yang terbatas dan berbeda satu sama lainnya, sehingga kemampuan menggunakan dan menginterpretasi Bahasa Inggris pun berbeda. Kondisi ini akan dapat menyebabkan terjadinya perbedaan arti yang timbul pada satu kalimat yang sama.

Salah satu faktor penghambat dalam memahami Bahasa Inggris adalah adanya kata-kata ambigius yang mempunyai makna yang tidak spesifik, melainkan bisa lebih dari satu atau rancu.

Untuk memahami maksud kata ini dapat digunakan algoritma *parsing*. *Parsing* adalah algoritma untuk mengurai suatu string (kalimat) menjadi frase-frase atau kata-kata yang menyusunnya. *Parsing* dapat digunakan untuk menganalisis suatu kalimat menurut pola tata bahasa yang berlaku. Saat ini telah banyak ditemukan berbagai algoritma *parsing* misalnya : *left right, top down, bottom up* dan *left corner.* Dalam Tugas Akhir ini akan digunakan implementasi algoritma left corner parsing untuk mengecek tata bahasa dan melakukan analisis makna kata yang berlaku dalam Bahasa Inggris.

I.2. Permasalahan

Ketiga tahap interpretasi sintaktik, semantik dan pragmatik ini memiliki permasalahan yang berbeda. Pada tahap sintaktik, masalah yang ditemui adalah bagaimana menyusun kalimat dengan susunan huruf dan kata menurut tata bahasa yang benar. Pada tahap semantic adalah bagaimana memilih arti yang tepat bagi kata-kata dalam kalimat dengan susunan sintaks yang benar tersebut. Dan pada tahap pragmatik adalah memahami apa yang dimaksud dengan isi kalimat tersebut. Poin utama penyelesaian permasalahan dalam perangkat lunak ini adalah mengidentifikasikan kata-kata yang menyusun sebuah kalimat Bahasa Inggris. Identifikasi ini meliputi beberapa atribut seperti jenis kata, jenis frase, imbuhan, dan artinya. Karena satu kata dapat memiliki beberapa macam arti,

maka dengan kombinasi beberapa atribut ini, akan dapat dilakukan pencarian arti yang sedapat mungkin paling tepat dari dalam basisdata.

I.3. Tujuan dan Manfaat

Pembuatan Tugas Akhir ini bertujuan untuk membuat perangkat lunak yang dapat mengimplementasikan algoritma Left Corner Parsing dalam memeriksa kebenaran tata bahasa dan memilih atau menentukan makna yang sesuai pada suatu kata yang menyusun kalimat Bahasa Inggris. Tahap analisis yang bisa dilakukan adalah sampai pada analisis semantik, yaitu pemeriksaan terhadap makna sebuah kata, bukan arti kombinasi beberapa kata sehingga membentuk makna frase atau kalimat.

Manfaat yang dapat diperoleh adalah dapat membuktikan algoritma Left Corner Parsing untuk implementasi otomatisasi pencarian arti dari kata-kata dalam sebuah kalimat bahasa Inggris yang dilakukan dengan proses scanning dan parsing.

I.4. Pembatasan Masalah

Batasan-batasan masalah yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

- Format input kalimat yang digunakan adalah berbasis file Teks (.txt) atau Rich Text Format (.rtf).
- Penulisan input kata yang tidak dikenal di dalam basisdata akan dikenal sebagai variabel, kecuali yang berawalan huruf besar akan dikenal sebagai nama benda (Proper Noun) .
- Penulisan karakter yang tidak dikenal akan dianggap sebagai error input.

- Perangkat lunak ini hanya bisa dijalankan di atas sistem operasi Windows 95 dan WindowsNT yang memiliki *database engine* untuk mengolah data yang diperlukan perangkat lunak.
- Tidak membahas mengenai solusi kebenaran yang seharusnya dilakukan pada input program yang salah.
- Perangkat lunak yang dibuat ini adalah membuktikan keberhasilan implementasi algoritma *Left Corner parsing* untuk memeriksa makna sebuah kata, bukan membandingkannya dengan algoritma yang lain dengan parameter keberhasilan, efektifitas, atau kecepatan yang empirik.

I.5. Metodologi

Pembuatan tugas akhir ini menggunakan metodologi sebagai berikut:

1. Studi Literatur

Studi literatur dilakukan dengan cara mengumpulkan dan mempelajari segala macam informasi yang berhubungan dengan teknik kompilasi, grammer dan kosakata Bahasa Inggris, algoritma scanner, algoritma Left Corner Parsing, serta metode pemrogramannya.

2. Perancangan Perangkat Lunak

Pada tahap ini dilakukan perancangan sistem perangkat lunak yang akan dibuat menurut persoalan yang ada dan dengan berdasarkan studi literatur yang ada.

3. Implementasi.

Dari perancangan sistem yang dibuat, akan dihasilkan desain objek, diagram alir, dan struktur data yang kemudian diimplementasikan ke dalam bentuk program.

4. Uji Coba dan Revisi Perangkat Lunak

Pada tahap ini dilakukan uji coba dan beberapa revisi, jika terjadi kekurangan dan kesalahan, terhadap perangkat lunak yang telah selesai dibuat, dan diharapkan perangkat lunak tersebut mengalami segala macam uji coba sehingga menghasilkan output yang diharapkan.

5. Pembuatan Naskah

Sebagai tahap akhir proses penelitian adalah pembuatan laporan atau dokumentasi secara lengkap dari semua kegiatan yang telah dilakukan.

I.6. Sistematika Pembahasan

Penulisan buku tugas akhir ini dibagi menjadi 7 bab. Dan penjelasan dari masing-masing bab adalah sebagai berikut :

Bab I : Pendahuluan

Bab ini membahas tentang latar belakang yang menyebabkan timbulnya permasalahan dan batasan masalah dari tugas akhir yang dikerjakan. Selain itu dijelaskan juga mengenai tujuan, metodologi, serta sistematika penulisan dari tugas akhir ini.

Bab II : Pengenalan Tata Bahasa Inggris

Bab ini menerangkan tentang pengenalan tentang Bahasa Inggris, mulai dari sifat karakter yang dikenali, kata, frase dan jenis-jenis kalimatnya.

Bab III : Teknik Kompilasi

Bab ini membahas tentang dasar pengolahan string, scanning dan proses parsing pada umumnya.

Bab IV : Algoritma Left Corner Parsing

Bab ini berisi tentang metode dan teknik pemrogram parsing dengan metode Left Corner.

BAB V : Perancangan dan Implementasi Perangkat Lunak

Bab ini berisi laporan perancangan dan pembuatan program implementasi algoritma Left Corner parsing untuk memeriksa kebenaran grammer Bahasa Inggris dan memeriksa makna yang sesuai. Terdiri dari algoritma scanner untuk memeriksa kebenaran token input, algoritma Left Corner parsing, diagram alir program, perancangan objek, dan algoritma per tahapan proses yang dilakukan. Sekaligus beberapa contoh implementasi prosedure utama perangkat lunak.

Bab VI : Evaluasi Perangkat Lunak

Berisi tentang bahan-bahan evaluasi terhadap implementasi perangkat lunak seperti : tata muka, data-data statis dan pengujian perangkat lunak. Pengujian ini meliputi : metode pengujian, hasil-hasil pengujian, data hasil pengujian, dan analisa pengujian.

Bab VII : Penutup

Berisi kesimpulan mengenai hasil pengujian dan analisa hasil pemeriksaan tata bahasa yang dipakai, jenis dan arti kata yang dihasilkan dari proses *Left Corner parsing.* Tingkat kebenaran proses yang telah dilakukan. Kemudian juga saran-saran untuk pengembangan perangkat lunak lebih lanjut.

BAB II
TATA BAHASA INGGRIS

Seperti pada bahasa umumnya, Bahasa Inggris juga memiliki pola-pola tertentu dalam menyusun sebuah kalimat. Pola-pola ini disebut dengan tata bahasa atau *grammar.*

Secara garis besar semua bahasa hampir memiliki kesamaan dalam tata bahasanya. Hanya beberapa hal kecil yang membedakannya. Misalnya penyusunan kata, model huruf, pengucapan, dan maksud implisitnya. Sebagai contoh, setiap bahasa pasti memiliki beberapa jenis kata seperti : kata kerja, kata sifat, kata benda, dan kata keterangan. Juga pola kalimat, frase, susunan kata, dan tanda baca.

II.1. Karakter yang dikenali

Untuk memeriksa kebenaran penulisan kalimat Bahasa Inggris, maka yang pertama harus dilakukan adalah mengenali karakter yang menyusunnya. Karakter ini merupakan struktur terkecil dari sebuah kalimat. Karakter-karakter ini dalam membentuk susunan kalimat harus menempati posisi yang benar, sehingga bisa dikatakan sebagai kalimat yang benar. Selain itu juga ada karakter yang tidak dikenal di dalam bahasa Inggris, sehingga penempatan karakter jenis ini dimanapun juga akan menyebabkan kesalahan pada penulisan kalimatnya. Kesalahan penyusunan karakter dalam sebuah Bahasa Inggris juga merupakan salah satu penyebab kesalahan penyusunan kalimat.

Karakter dalam bahasa Inggris akan membentuk beberapa macam fungsi dalam sebuah kalimat. Ada beberapa macam hasil bentukan dari susunan berbagai karakter, antara lain :

1. Membentuk kata, angka atau variabel. Sebagai contoh adalah huruf 'a' sampai 'Z', atau gabungan angka dan huruf tersebut akan membentuk kata variabel. Kata variabel adalah jenis kata yang tidak dianggap sebagai kata benda, namun kata benda abstrak.
2. Sebagai tanda baca. Sebagai contoh adalah titik (.), koma (,) dan lain sebagainya. Tanda baca adalah karakter tunggal yang membentuk struktur tersendiri yang diperhitungkan di dalam pembentukan dan analisa sebuah kalimat.
3. Sebagai mata uang, misalnya adalah karakter dolar ($) yang ditempatkan di depan angka.
4. Sebagai persamaan matematika atau yang menyertai bilangan. Sebagai contoh adalah tambah (+), kurang (-) atau prosentase (%).
5. Sebagai karakter tak dikenal, atau nama variabel.

Daftar jenis karakter dan fungsi yang dibentuknya dalam kalimat dijabarkan selengkapnya pada tabel lampiran.

II.2. Jenis-jenis Kata

Kata adalah susunan beberapa huruf yang memiliki nilai makna dalam sebuah kalimat. Di dalam Bahasa Inggris dikenal beberapa macam kata, antara lain adalah :

II.2.1. Verb (kata kerja)

Di dalam kata kerja terdapat empat macam kategori, yaitu :

- *Direct Transitive Verb* adalah kata kerja transitif berobjek tunggal seperti : *spend dan like.*
- *Indirect Transitive Verb* adalah kata kerja transitif berobjek ganda seperti contoh adalah kata : *buy dan make.*
- *Intransitive Verb* adalah kata kerja yang tidak memerlukan objek langsung. Sebagai contoh adalah kata : *bore, dan sleep.*
- *Semi Intransitive Verb* adalah kata kerja yang bisa diikuti objek atau tidak. Sebagai contoh adalah kata : *write, read, eat.*

II.2.2. Noun (kata benda)

Kata benda (noun) memiliki beberapa macam bentuk menurut ketegori yang membedakannya. Antara lain :

Kelompok kata benda berdasarkan bentuk.

Adalah jenis kata benda yang dibedakan berdasarkan sifat atau bentuk benda yang diwakilinya.

1. *Common Noun* (kata benda biasa) : *table, mouse, monitor.*
2. *Proper Noun* (nama benda) : *Surabaya, Anton.*
3. *Abstract Noun* (kata benda tak berbentuk) : planning. Biasanya kata ini terbentuk dari penggabungan kata.
4. *Collective Noun* (kata benda kumpulan). Adalah satu kata yang merupakan kelompok seperti : *classmate*

Kata benda berdasarkan gender.

Adalah jenis kata benda dilihat dari sisi jenis kelamin benda yang diwakilinya.

1. *Masculine*, jenis kelamin laki-laki seperti : *father, grandfather*
2. *Feminine,* perempuan seperti : *girls, mom*
3. *Common Gender*, yang mewakili manusia seperti : *people, grand*
4. *Neuter Gender,* kata benda tak berkelamin seperti : *book, table*

Kata benda berdasarkan kejamakan.

Adalah jenis kata dilihat berdasarkan jumlah benda yang diwakilinya.

1. *Singular Noun* (kata benda tunggal) adalah untuk menyebutkan jumlah benda dalam kalimat tersebut tunggal.
2. *Plural Nouns* (kata benda jamak) adalah untuk menyebutkan bahwa benda dalam kalimat tersebut jamak atau lebih dari satu.

II.2.3. Adjective (kata sifat)

Kata sifat adalah kata yang digunakan untuk menerangkan kata benda. Di dalam Bahasa Inggris, dalam frase kata benda, kata sifat ditempatkan di depan kata benda.

Beberapa macam fungsi kata sifat.

1. *Quality Adjective* (menyatakan kualitas atau deskripsi) seperti : *green grass.*

2. *Quantity Adjective* (menyatakan jumlah), yaitu menyatakan seberapa besar atau seberapa banyak. Jenis kata sifat ini dibedakan menjadi dua jenis, yaitu definite dan indefinite. Definite untuk menyatakan jumlah yang bisa dihitung seperti : *one, three, third*, sedangkan indefinite menyatakan jumlah yang tak terhitung seperti : *some, no, all.*
3. *Possesive Adjective* (menyatakan kepemilikan) seperti : *my, your.* Kata ini akan selalu diikuti oleh frase kata benda. Contoh : *my book, your car.*
4. *Distributive Adjective*, adalah menyatakan bahwa kata yang diterangkan (kata benda) tersebut adalah terdiri dari banyak yang terpisah-pisah dan disebutkan secara tunggal. Sebagai contoh adalah : *each, every.* Pada Distributive Adjective ini, bentuk kata benda yang diterangkan selalu tunggal.
5. *Interogative Adjective*, adalah kata tanya yang digunakan untuk menanyakan/menunjukkan kata sifat. Sebagai contoh adalah : *which book .*
6. *Demonstrative Adjective*, adalah kata sifat yang menunjuk pada kata benda. Sebagai contoh adalah : *that, this, those.*
7. *Indefinite Adjective*, adalah kata sifat yang menunjuk sesuatu, namun tidak jelas diketahui sesuatu tersebut. Contoh :*other boy .*

Perbandingan pada kata sifat

Kata sifat yang menyatakan kualitas (*quality adjective*) juga digunakan untuk membandingkan tingkat kesifatan kata benda yang diterangkan. Ada tiga kategori untuk jenis ini yaitu :

1. *Positive Degree*, adalah untuk kata sifat biasa. Seperti *small book, crazy boy.*
2. *Comparative Degree*, adalah digunakan untuk membandingkan dua buah benda. Ada dua metode yang digunakan yaitu : menambahkan akhiran '–er' pada kata sifatnya atau *more* sebelumnya.
3. *Superlative Degree,* adalah untuk membandingkan lebih dua kata benda. Metodenya dengan menambahkan akhiran '-est' untuk kata sifat tidak berimbuhan dan *most* untuk kata sifat berimbuhan.

Selain dengan metode standar, ada beberapa kata khusus (irregular adjective comparison) yang memiliki bentuk sendiri pada *comparative* maupun *superlative degree.*

II.2.4. Adverb (kata keterangan)

Berbeda dengan kata sifat, kata keterangan ini digunakan untuk menerangkan kata kerja. Adverb juga bisa dihasilkan dari jenis kata lain dengan penambahan akhiran –ly dari kata sifat. Seperti contohnya : *quickly, cleverly*, dan seterusnya.

Namun demikian, ada beberapa bentuk kata keterangan dari kata sifat yang khusus memiliki bentuk tersendiri. Seperti : *very, soon, rather, how* dan seterusnya. Beberapa macam kata keterangan antara lain :

1. *Adverb Of Manner* (keterangan cara). Biasa ditempatkan setelah kata kerja, contoh : *cleverly, hard, fast, loudly.*
2. *Adverb Of Time* (keterangan waktu). Menerangkan waktu pekerjaan : *at the afternoon, in the morning.*
3. *Adverb Of Place* (keterangan tempat). Menerangkan tempat : *there*
4. *Adverb Of Frequancy* (frekwensi kegiatan). Ditempatkan sebelum kata kerja, misalnya : *never, ever, always* dan lain-lain.
5. *Adverb Of Degree* (tingkat kesifatan). Contoh : *very, almost.*
6. *Adverb Of Negation* (berlawanan) Contoh : *not*

II.2.5. Article (kata sandang)

Kata sandang adalah kata yang digunakan untuk menunjuk ke suatu kata benda. Kata ini menempati posisi di depan kata benda atau frase kata benda. Ada tiga macam saja kata sandang dalam Bahasa Inggris yaitu : *a, an* dan *the*. Pemakaian 'a' harus diikuti oleh kata yang huruf depannya konsonan sedangkan 'an' harus diikuti oleh kata yang huruf depannya vokal.

II.2.6. Possessive (kata ganti milik)

Kata ganti milik telah dijabarkan pada item kata sifat kepemilikan. Juga bisa digolongkan pada kelompok kata ganti benda (pronoun posessive) maupun pronoun possessive.

II.2.7. Pronoun (kata ganti orang/benda)

Pronoun adalah kata yang digunakan untuk mewakili suatu benda atau orang. Kata ini akan menunjukkannya tanpa harus dengan namanya. Pronoun ada beberapa macam yaitu :

1. *Personal Pronoun*, adalah kata ganti digunakan untuk mewakili orang. Dibagi menjadi tiga bagian yaitu
   - *First person* (orang pertama) seperti :*I, we*
   - *Second person* (orang kedua) seperti : *you.*
   - *Third person* (orang ketiga) seperti : *he, she, it, they.*

   Selain ketiga jenis tersebut, berdasarkan jumlah nominal kata benda yang diwakilinya dibagi menjadi dua jenis yaitu :
   - *Singular pronoun* (yang menyatakan pronoun tunggal)
   - *Plural pronoun* (menyatakan pronoun banyak/jamak).
2. *Possessive Pronoun*, adalah pronoun yang juga menyatakan kepemilikan. Kata ini ditempatkan di depan frase kata benda. Kelompok kata ini sama dengan possessive adjective, namun perbedaannya terletak pada bentuk penggunaan kata tersebut dalam kalimat. Possessive pronoun tidak diikuti

kata benda sedangkan possessive adjective diikuti bendanya. Sebagai contoh : **yours** is bad.

3. *Reflexive and Emphasizing Pronoun*, digunakan untuk menunjukkan bahwa pelaku (subyek) adalah juga menjadi objek dalam kalimat tersebut. Jadi kata ini akan selalu mengikuti subyeknya, bersebelahan maupun menjadi objek kalimat setelah frase kata kerja. Contohnya seperti : He killed **himself**, I **myself** saw it.
4. *Demostrative Pronoun*, digunakan untuk menunjukkan sesuatu atau seseorang. Kelompok kata ini sama dengan *distributive adjective*, namun dapat dibedakan pada penempatan katanya yaitu, jika berdiri sendiri sebagai pronoun, namun jika diikuti frase kata benda akan menjadi *distributive adjective*. Contoh : **that** is right,
5. *Interrogative Pronoun*, adalah kata tanya yang menunjuk ke suatu benda namun tidak diikuti frase kata benda itu sendiri, sedangkan interogative adjective adalah diikuti oleh frase kata benda. Sebagai contoh adalah : **Which** is yours ? berbeda **Which** pencil is yours.
6. *Relative Pronoun*, digunakan untuk kata sambung (*conjunction*). Biasanya berarti 'yang'. Sebagai contoh : Joko is a good boy **who** does'nt tell a lie.
7. *Indefinite Pronoun*, sabagai penunjuk jumlah yang tidak tertentu, hanya tidak diikuti oleh frase kata benda. Sebagai contoh : First one, than another.
8. *Distributive Pronoun*, adalah kata distributive yang tidak diikuti oleh kata benda. Sebagai contoh : **Each** is three years.

II.2.8. Preposition (kata depan)

Preposition adalah kata yang selalu ditempatkan di depan frase kata benda (*noun*) atau kata ganti benda (*pronoun*). Seperti contohnya adalah : *on, under, before.*

II.2.9. Auxiliary verb (kata kerja bantu)

Auxiliary verb adalah kata kerja yang biasanya ditempatkan di depan kata kerja. Berbeda dengan sekedar kata bantu (TO BE), kata ini memiliki arti tertentu. Sebagai contoh adalah : *can, could,* dan *should.*

II.2.10. To be (kata bantu)

Kata bantu adalah kata pelengkap yang tidak memiliki makna spesifik selain hanya menunjukkan status kalimat yang ada di dalamnya. Seperti contoh : *are, am, was, have* dan sebagainya.

II.2.11. Conjuction (kata penghubung)

Conjunction adalah kata yang digunakan sebagai penggabung pada dua atau lebih kalimat kata atau yang memiliki struktur yang sama atau hampir sama. Contoh : **You and I**. I want to sleep **but** you don't. Terdapat tiga macam conjunction, antara lain :

1. *Coordinative Conjunction* adalah yang biasanya menghubungkan antar frase seperti *but, and, or.*
2. *Subordinate Conjunction* adalah konjungsi tertentu untuk membentuk frase tersendiri seperti *both, between, neither nor, not only also, either or* dan sebagainya.

3. *Correlative Conjunction* adalah kata penghubung yang digunakan untuk menghubungkan anak dan induk kalimat pada kalimat majemuk. Seperti *however, because, until, before* dan sebagainya.

II.2.13. Interogative (kata tanya) .

Kata tanya biasanya ditempatkan pada awal kalimat. Kata tanya yang berada di tengah kalimat bisa menjadi relative pronoun yang menjelaskan benda yang diterangkan. Kata tanya memiliki beberapa macam misal : berdiri sendiri, interogatif pronoun, dan interogatif adjective. Contoh **who** is that, **which** is yours, **which book**.

II.2.14. Interjection (kata seru)

Kata ini biasanya berdiri sebagai sebuah kalimat tunggal yang selalu dipisahkan dengan tanda baca akhir kalimat. Sebagai contoh adalah: **Oh!, Pardon! Hallo**!..dan sebagainya

II.3. Jenis-jenis Frase

Frase adalah sekumpulan kata yang saling berhubungan yang menjabarkan subyek atau kata kerja [8]. Dalam frase terdapat kata utama dan pembantu yang memiliki fungsi untuk menerangkan kata utama tersebut. Dalam bahasa Inggris terdapat beberapa macam frase :

II.3.1. Verb Phrase (frase kata kerja)

Adalah frase yang terdiri dari kata kerja sebagai kata utama ditambah satu atau lebih kata pembantu yang menerangkannya. Pada umumnya, secara semantic, frase kata kerja ini dapat terdiri dari kata kerja transitif atau intransitif yang diikuti oleh frase kata keterangan dan atau kumpulan frase preposisi.

Beberapa modifikasi frase kata kerja antara lain :

- Frase kata kerja yang memiliki obyek langsung (*direct*) dan tak langsug (indirect). *Indirect object* ditempatkan sebelum *direct object*: Contoh : He buys **me** a bottle.
- Frase kata kerja yang menggunakan kata **also** atau **only.** Also dapat selalu ditempatkan sebelum kata kerja. Maka di sini also bisa dianggap sebagai kata keterangan (adverb) Demikian juga dengan penggunaan kata only yang menempati depan kata kerja.

II.3.2. Noun Phrase (frase kata benda).

Frase kata benda adalah dibentuk oleh kata benda dan kata yang menerangkannya. Frase kata benda dapat terbentuk dari kata sifat, kata preposisi atau kata kata benda sendiri. Beberapa modifikasi frase kata benda misalnya :

1. Frase kata benda yang menggunakan kata benda refleksif (*refleksive noun*). Seperti : *Do it by yourself.*
2. Frase kata benda *appositive*, yaitu kata benda yang diikuti oleh frase kata benda yang memperjelas kata sebelumnya. Dibedakan menjadi

dua yaitu restrictive (tanpa koma) dan nonrestrictive (dengan koma). Non appositive biasanya didahului oleh proper noun dan dipisahkan dengan koma. Contoh : *Harry Connick,* the musician, *will come to Champaign.*

3. Frase kata benda yang memiliki komplemen: memodifikasi noun/pronoun dalam kalimat dan ditempatkan setelahnya. Contoh : If you elect **me** *president*, I'll keep the **unions** *satisfied* .
4. Frase kata benda yang menggunakan Relative Pronoun (yang mewakili noun atau pronoun) di dalamnya. Contoh : Where is the girl *who is going* (tidak ada koma); That girl, *who is going to the concert*, has a green dress.
5. Frase kata benda yang berbentuk verbal, yaitu frase kata benda yang dibentuk oleh frase kata kerja. Beberapa macam frase kata benda verbal :
   - *Participle Phrase* (bentuk lampau) contoh : *The girls, frightened by the police car's headlights, quickly came down from the school's roof.*
   - *Gerund Phrase* (bentuk sekarang) contoh : *Making many acquaintances* is *cultivating future friendships*; *The woman* denied *knowing her own husband*
   - *Infinitive Phrases* (kata kerja bentuk pertama) contoh : *To live in Boston eventually is his main goal in life. Quentin Tarentino loves to babble during interviews.*

II.3.3. Prepositional Phrase List (rangkaian frase preposisi)

Adalah jenis frase yang tersusun dari kata preposisi dan kata benda. Seperti frase *The front of the Building.*

Ada banyak frase yang tidak bisa didifinisikan secara tekstual. Frase-frase ini dikombinasikan menurut tata aturan grammar yang berlaku pada Bahasa Inggris. Frase ini akan dibutuhkan pada saat pembentukan Production Rule untuk menjalankan algoritma parsing.

II.4. Imbuhan

Imbuhan pada suatu kata dapat mempengaruhi pembentukan jenis kata selanjutnya selain sebelum mendapat imbuhan. Imbuhan tidak termasuk pembahasan dalam tugas akhir ini karena bentuk katanya dibatasi pada kata yang sudah berimbuhan.

II.5. Kalimat

Beberapa macam kalimat yang bisa dikenali antara lain adalah :

1 Kalimat berita (*statement*), adalah kalimat yang susunannya terdiri dari subyek dan predikat yang diakhiri dengan tanda titik. Diawali dengan frase kata benda yang diikuti oleh kata bantu atau kata kerja.

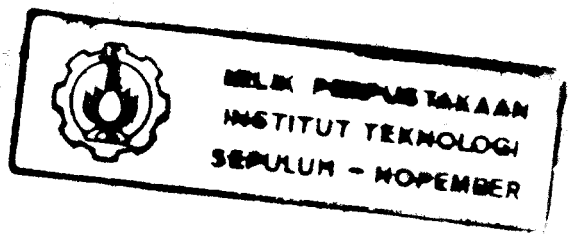
2 Kalimat tanya (*question*). Adalah kalimat yang diawali dengan kata tanya atau kata bantu dan diakhiri dengan tanda tanya.

3 Kalimat perintah/permintaan (*command/desire*). Adalah kalimat yang diawali dengan kata kerja bentuk pertama atau kata 'please' dan diakhiri dengan tanda seru.

4 Kalimat seru (*exclamation*). Adalah kalimat yang diawali dengan kata-kata seru dan diakhiri dengan tanda seru.

5 Kalimat *past tense, present tense* dan *past participle*. Adalah kalimat yang menunjukkan waktu pelaksanaan kegiatan yang dimaksud pada kalimat tersebut. Hal ini ditunjukkan dengan pemakaian kata kerja atau kata bantu bentuk pertama, kedua atau ketiga. Namun dalam pengenalan sintaks hal ini masih diabaikan.

6 Kalimat Kompleks, adalah kalimat yang terdiri dari beberapa anak kalimat di dalamnya yang dihubungkan dengan kata penghubung. Contoh : *Timmy likes grammar class, so he does not like "Magma"; Timmy thinks of grammar when he is at a "Magma" concert*



BAB III

TEKNIK KOMPILASI

Teknik Kompilasi adalah ilmu Pengolahan Bahasa (Language Processing) yang merupakan cabang dari ilmu Artificial Intelligence (AI). Dalam teknik kompilasi terdapat proses-proses untuk memecahkan persoalan dengan melakukan berbagai uji coba dan pemilihan terhadap beberapa alternatif solusi.

III.1. Artificial Intelligence

AI adalah ilmu yang memungkinkan komputer dapat bekerja sebagaimana seorang manusia untuk menyelesaikan pekerjaannya. Permasalahan-permasalahan yang solusinya tidak diketahui manusia secara pasti adalah termasuk dalam ruang linkup AI. Solusi ini diselesaikan oleh AI dengan algoritma, urutan dan selang waktu tertentu, dan akan bekerja menurut pola yang telah ditentukan. Dengan ini, komputer akan dapat memilih sendiri salah satu solusi yang tepat bagi permasalahan tersebut.

Dua karakteristik permasalahan yang bisa diselesaikan oleh sistem AI adalah :

1. Permasalahan tersebut berhubungan atau dapat dihubungkan dengan simbol-simbol informasi seperti karakter, huruf, tanda baca, simbol tertentu atau gambar
2. Permasalahan tersebut dapat diatasi dengan beberapa pilihan solusi. Pilihan solusi inilah yang harus dipilih oleh AI sebagai sebuah solusi yang paling tepat.

Demikian juga dengan menyelesaikan permasalahan kompilasi. Input string kalimat yang diberikan akan diproses oleh *compiler* berdasarkan beberapa alternatif solusi yang disimpan dalam sebuah rule, kemudian harus dipilih salah satunya oleh komputer sebagai sebuah solusi yang paling tepat.

III.2. Natural language Processing

Natural Language Processing (NLP) adalah ilmu yang digunakan untuk mendesain dan membangun sebuah sistem komputer yang dapat menganalisa, mengerti, dan mengenerate fungsi dalam suatu bahasa. Sistem NLP ini akan mengambil input teks sebagai sebuah bahasa, dan melakukan langkah-langkah proses linguistik, dari melalui analisis leksikal (morfologi), sintaks, semantik, pragmatik, *ambiguity*, derivasi makna dan juga *rule of context*.

Bahasa adalah kumpulan dari kalimat yang saling berhubungan satu dengan lainnya, dan memiliki aturan yang baku dalam penyusunannya. Dalam Bahasa Inggris, aturan susunan kata dan kalimat ini bisa tidak terbatas, oleh karena itu sulit untuk membuat aturan yang dinamis yang mencakup semua kemungkinan *grammar* sebuah kalimat. Namun, dalam penelitian ini, ketidakterbatasan ini dibatasi dengan sebuah kumpulan aturan yang masih bisa dimodifikasi.

Pembatasan ini akan dilakukan dengan suatu prosedur keputusan, yang akan mengembalikan nilai *true* jika kalimat tersebut dianggap sah, atau *false* jika tidak. Pengujian terhadap kebenaran kalimat tersebut bisa dilihat/diuji dari makna katanya setelah pemrosesan. Jika sesuai dengan

yang dimaksud oleh pembicara, maka bisa dikatakan proses NLP tersebut benar, dan jika masih ada penyimpangan berarti masih ada kekurangan dalam tahapan analisanya.

Dalam menentukan makna kata ini, ada beberapa tahapan yang harus dilakukan yaitu : analisis sintaks, semantik dan pragmatik.

Analisis sintaks menentukan kebenaran (benar tidaknya) susunan jenis kata berdasarkan struktur grammar input kalimat yang diberikan. Sedangkan semantik akan menentukan arti literal dari hasil proses sintaks tersebut. Dan untuk mengetahui arti kata secara semantik harus diketahui dulu arti masing-masing kata kemudian mengkombinasikannya dengan kata di sekitarnya. Sedangkan pragmatik adalah menentukan maksud yang lebih dalam (yang mungkin ada) pada kalimat tersebut. Misalnya ungkapan-ungkapan atau peribahasa yang sering ada di bahasa manapun.

Sebagai contoh perhatikan percakapan berikut :

Elizabeth : *Did you know that Victor used to smoke just like you ?*
Jack : *No. What happened to him ?*
Elizabeth : *Lung Cancer. He kicked the bucket.*
Jack : *I'm sorry to hear it.*

Perhatikan frase 'kicked the bucket' di atas. Frase di sini tidak bisa diartikan secara per kata, namun dia mewakili satu kata sifat : 'mati'. Untuk memahami maksud kalimat ini, diperlukan pemahaman terhadap sintaks, semantik dan pragmatik. Seluruh kalimat dianalisis dan kemudian dicari frase atau kata yang sebenarnya merupakan kata ganti dari bentuk lain, kata yang memiliki arti *ambiguity* atau makna yang lebih dalam.

Akhirnya, untuk memperoleh makna yang komprehensif, analisis tersebut memang harus ditempatkan di dalam konteks permasalahan yang sedang dihadapi dengan parameter keyakinan, tujuan, kecenderungan dan rencana atau maksud dari pembicara.

Analisis kalimat ini juga bisa berhenti pada salah satu level analisis. Dan untuk mengetahui dengan pasti kesalahan yang mungkin ditemui selama proses parsing, maka harus bisa diketahui setiap detil permasalahan dalam level-level tersebut. Konsekwensinya memang orang harus lebih familiar dengan sintaks, semantik dan pragmatik serta permasalahan yang berhubungan dengan kalimat tersebut.

III.3. Compiler dan Translator

Terminologi *compiler* dan *translator* memiliki pengertian yang berbeda. *Translator* adalah sebuah program yang sekedar mengambil input dari suatu bahasa dan menghasilkan output dalam bahasa yang lain. Sedangkan jika sumber bahasa tersebut adalah sebuah *high level language* seperti Pascal, Deplhi, Visual Basic dan objek bahasanya adalah sebuah *low level language* seperti bahasa assembly atau bahasa mesin, maka translator tersebut itu disebut compiler.

Di dalam *compiler,* menjalankan program dalam high-level language pada dasarnya memiliki dua langkah proses. Pertama program sumber (*source*) harus di-*compile*, yaitu dibentuk menjadi sebuah objek program. Baru kemudian program hasil kompilasi ini akan di-*load* ke memory dan dijalankan.

Khusus tipe *translator* yang mentransformasikan bahasa pemrograman ke bahasa yang lebih sederhana (*intermediate code)*, disebut

dengan *interpreter*. Dan, jika sumber bahasanya adalah bahasa assembly dan kemudian bahasa targetnya adalah bahasa mesin, hal ini disebut dengan *assembler*.

Translator dibutuhkan dalam mengkomunikasikan dua buah bahasa yang berbeda. Jika manusia mengenal karakter dalam pemakaian bahasanya, maka komputer adalah memakai bit '1' dan '0' untuk mewakili bahasanya. Inilah yang harus diterjemahkan oleh compiler. Demikian juga dengan Bahasa Inggris dengan bahasa Indonesia, atau sudut pandang Bahasa Inggris sendiri memerlukan translator untuk memahami kedalaman maksud bahasanya sendiri.

III.4. Lexical Analysis

Analisis leksikal bisa disebut juga dengan analisis morfologi, yaitu aturan susunan kata yang diijinkan dalam sebuah bahasa [8]. Bahasa sendiri dapat dibedakan menjadi tiga kelompok klas : regular, context-free, dan general. Masing-masing memiliki perbedaan dan karakteristik tersendiri, yaitu :

1. Regular Language (RL) dapat digambarkan dengan regular expression atau regular grammar. Bahasa ini dapat dianalisa dan diterima oleh Finite State Automata
2. Context-Free Language (CFL) dapat digambarkan dengan Context-Free Grammar (CFG) atau Recursive Transition-Network (RTN) dan dapat diterima oleh mesin Push Down Automata (PDA).

3. Genearal Language dapat digambarkan dengan Context-Sensitive Grammar atau Augmented-Transition Network dan dapat diterima atau dianalisa oleh Turing Machine

Aturan grammar Bahasa Inggris adalah termasuk dalam CFL, sehingga pemeriksaan terhadap grammar yang digunakan dapat dilakukan dengan sistem PDA. Sistem ini digunakan untuk dapat meng-*handle* proses yang recursive, yang menggunakan stack untuk mengorganisasikan memory, karena bisa menyimpan simbol untuk sementara waktu.

III.4.1. Finite Automata

Dalam mengenali jenis-jenis token berdasarkan input string dibutuhkan beberapa mekanisme tersendiri. Mekanisme ini dapat dibentuk dengan diagram transisi yang disebut dengan Finite State Automation (FA). Sedangkan FA sendiri terbagi menjadi dua yaitu : Nondeterministic Finite Automata (NFA) dan Deteministic Finite Automata (DFA).

FA dapat digambarkan sebagai sebuah *directed graph* yang terdiri dari *node-node* yang dihubungkan dengan garis. Masing-masing *node* ini disebut dengan *state*, sedangkan garis yang memiliki label dan arah panah ini disebut dengan *transition*. Dalam satuan FA ini terdapat dua buah *state khusus* yang disebut dengan *start state* dan *final state*.

NFA dapat dipakai untuk mengenali Regular Expression. Dalam NFA, label garisnya dapat diberi label karakter ϵ (empty) sebagaimana karakter biasa, dan satu state dapat mengeluarkan dua atau lebih karakter yang sama.

NFA masih sulit untuk diimplementasikan dalam sistem komputer. Untuk itu dalam FA ada versi deterministic. DFA ini digunakan untuk mengenali Regular Expression. Disebut deterministic jika FA tersebut memiki dua ciri atau syarat sebagai berikut :

1. Tidak ada transisi ϵ (empty) pada inputnya
2. Pada masing-masing state, jika ada input karakter, maka maksimal terdapat sebuah karakter yang keluar dari state tersebut.

III.4.2. Scanner

Analisis leksikal (morfologi bahasa) akan dilakukan oleh lexical analizer. Lexical analizer adalah interfase antara bahasa sumber dan compiler. *Lexical analizer* membaca sumber bahasa per karakter, dan kemudian dibentuk menjadi unit-unit yang disebut dengan token. Masing-masing token mewakili sebuah urutan karakter yang dikenal oleh compiler sebagai satuan unit entitas seperti : kata biasa, kata bilangan, operator matematika, tanda baca dan lain sebagainya. Sebagai contoh, pada kalimat :

The 23 books cost US$ 5.000,00

ditemukan token huruf awal (The), kata biasa, angka (23), currency (US$) dan nilai angka nominal (5.000,00).

Definisi token-token ini tergantung dari bahasa yang dipakainya. Dalam Bahasa Inggris dikenalkan 17 macam token, antara lain :

1.	COMMON	: huruf kecil semua
2.	FIRST	: kata biasa atau kata benda, karena huruf awalnya besar dan di awal kalimat
3.	NAME	: diawali huruf besar dan tidak di awal kalimat
4.	VARIABLE	: kombinasi karakter tak beraturan, bisa terdiri dari gabungan angka, huruf, huruf kecil dan besar
5.	ABREVIATION	: huruf besar semua
6.	NUMERIC	: angka semua
7.	CURRENCY	: gabungan angka yang membentuk currency
8.	POSSESSIVE	: memiliki tanda apstrok (') di dalamnya
9.	PERCENTAGE	: prosentase (angka + karakter '%')
10.	EQUATION	: gabungan angka, operator dan angka
11.	TIME	: karakter angka dan dipisahkan dengan karakter (':'), dapat menunjukkan tanda waktu
12.	DATE	: karakter-karakter angka dan dipisahkan dengan tanda ('-') yang menunjukkan tanggal
13.	SHORT	: diakhiri dengan titik, namun bukan sebagai akhir kalimat
14.	GROUP	: beberapa kata yang digabung dengan tanda ('-') yang membentuk kata tersendiri
15.	NUMBER	: gabungan huruf dan atau angka dan titik, dan diakhiri dengan karakter titik
16.	OPERATOR	: karakter tunggal operator matematika
17.	TDBACA	: karakter tunggal yang menunjukkan tanda baca dalam sebuah bahasa

Token-token yang dihasilkan ini kemudian dijadikan input ke dalam syntax analizer di bawah kontrol parser.

Dalam menemukan token-token yang dikenal ini, dibentuk program *scanner* yang akan mengambil karakter satu demi satu mulai dari awal string. Setiap karakter yang dilewati akan dianalisa berdasarkan karakter sebelumnya dan kemudian akan dikelompokkan ke dalam beberapa unit token menurut pola yang sudah diatur dalam sebuah rule.

Sebagai contoh karakter '$' yang diikuti oleh angka akan dibentuk sebagai token angka nominal uang. Angka-angka ini pun memiliki pola tersendiri, seperti setiap tiga angka yang dilewati harus ada tanda titik ('.') dan dua angka desimal dibelakangnya yang dipisahkan dengan tanda

koma (,). Jika susunan karakter ini tidak mengikuti pola, maka proses tidak bisa dilanjutkan, atau bisa disebut input error.

Proses scanning ini terbagi menjadi dua proses. Pertama, proses scanning akan menghasilkan beberpa jenis token tingkat dasar. Token-token di tingkat dasar ini kemudian dianalisis lagi untuk menemukan token tingkat berikutnya. Token-token tingkat kedua ini memang tidak terbentuk dari susunan karakter secara langsung, melainkan dari gabungan token-token yang sudah ada. Token-token tersebut adalah :

```
DATE        : token NUMERIC + token OPERATOR ('-')
TIME        : token NUMERIC + token OPERATOR (':')
GROUP       : token NUMERIC + token OPERATOR ('-')
EQUATION    : token NUMERIC + token OPERATOR selain
              pola TIME  dan DATE
```

III.5. Syntax Analysis

Setelah proses analisis leksikal di atas, maka akan dihasilkan unit-unit satuan token. Token-token ini kemudian akan dianalisis lagi untuk mendapatkan informasi lebih lengkap mengenai identitas token tersebut. Informasi token ini akan diproses oleh Syntax Analizer dengan menggunakan notasi yang disebut dengan Context Free Grammar (CFG).

III.5.1. Context Free Grammar

Di dalam sistem CFG dikenal empat macam entitas yaitu : *terminal*, *nonterminal*, *start symbol*, dan *productions*.

Terminal adalah simbol dasar dalam satuan string dalam sebuah bahasa. Dalam CFG sebuah *terminal* tidak memiliki percabangan lagi. Sebagai contoh, dalam Bahasa Inggris, kata 'working' merupakan kata kerja intransitive yang dapat disimbolkan dengan INTR_VERB | ING_VERB

dalam CFG. Simbol '|' bisa diartikan 'atau' karena kata tersebut bisa dianggap Intransitive Verb dan atau Ing Verb dalam sebuah kalimat. INTR_VERB dan ING_VERB ini tidak bisa dicabangkan lagi.

Sedangkan *nonterminal* adalah string simbol yang berisi atau menandakan kumpulan string simbol. String simbol ini bisa berisi *terminal* atau *nonterminal*. Dalam CFG Bahasa Inggris, NP merupakan *nonterminal* karena NP (*Noun Phrase*) adalah simbol untuk kumpulan string yang mewakili frase kata benda. Sebagai contoh :

```
NP → ANP RP          : NP adalah Non Terminal
NP → VERBAL          : det adalah terminal
NP → det ANP         : ANP, RP, VERBAL adalah non
NP → ANP             : terminal walaupun di sebelah kanan
```

Production adalah cara yang dilakukan untuk membentuk gabungan dari terminal dan nonterminal yang ada. Masing-masing production memiliki sebuah nonterminal di sebelah kiri, yang diikuti oleh string simbol terminal dan atau nonterminal di sebelah kanannya. 'NP→ANP RP' di atas adalah contoh sebuah production.

III.5.2. Parser

Proses selanjutnya dalam sistem kompilasi adalah parsing. Parsing adalah program yang mengambil input string yang dianggap sebagai kalimat dan menghasilkan output berupa *parse tree* (pohon parse) berdasarkan rule CFG yang berlaku. Dari parsing inilah kemudian disamping diketahui apakah string tersebut dikenal oleh *syntax rule* yang diberikan kalimat tersebut atau tidak. Jika tidak sesuai, maka bisa disebut

input string tersebut dianggap error. Namun, karena masih mungkin terdapat *ambiguity* dalam sebuah kalimat bahasa Inggris, maka analisis grammar terhadap kalimat string ini bisa dilakukan lebih dari satu kali.

Secara teknis, parsing digunakan untuk memeriksa apakah token-token input sesuai dengan pola yang sudah diatur dalam tata bahasa (*grammar*) yang benar .

Ada beberapa macam algoritma parser yang dapat digunakan untuk menganalisa sintaks sebuah bahasa, antara lain adalah *Shift Reduce, Top Down, Bottom Up*, dan *Left Corner Parser. Bottom Up* dan *Top Down Parser* disini akan dijadikan contoh karena merupakan dasar dari *Left Corner Parsing Algorithm.*

III.5.2.1. Top Down Parser

Algoritma ini memakai konsep penelusuran dari atas (*Top*) dimulai dari rule pertama atau rule yang berisi di kiri *Starting Symbol* (S) dan diikuti derivasi rulenya di sebelah kanan, hasil *production* dari S. Langkah ini kemudian diulang pada rule CFG selama masih ada nonterminal sampai ujung dari tree tercapai (terminal atau kata yang sesuai)

Adapun urutan implementasi algoritma yang digunakan untuk proses ini adalah :

```
1. Inisialisasi Stack = [S]
2. Jika top of stack nonterminal N, maka :
    - Pilih Rule yang sesuai dengan N --> R (R adalah
      terminal atau nonterminal)
    - Hapus N dari stack
    - Tambahkan R pada stack
3. Jika top elemen adalah sebuah terminal P, maka :
    - Temukan kata selanjutnya W dalam kalimat tersebut
```

- Jika ada Rule yang menyebutkan P --> W maka remove terminal dari stack

4. Jika stack = empty, dan tidak ada kata yang diparse lagi, maka stop.
5. Ulangi langkah ke 2.

III.5.2.2. Bottom Up Parser

Bottom Up Parser didasarkan pada *Shift Reduce parser.* Parser ini bekerja dengan memeriksa token-token dari *scanner* kalimat input dan memeriksanya dengan menggunakan aturan grammar yang ada. Ada dua proses utama dalam algoritma ini yaitu 'shift' untuk mengambil token/terminal berikutnya dan 'reduce' untuk menggabungkan terminal (konstituen) yang telah cocok.

Adapun urutan algoritma yang digunakan adalah :

1. Inisialisasi stack = [] (empty)
2. Langkah selanjutnya :
   - Select kata selanjutnya (dimulai dengan kata pertama).
   - Tentukan jenis/kategori kata tersebut dalam leksikal/lexicon
   - Push kategori tersebut kedalam stack
3. Atau lakukan fungsi reduce:
   - Jika kategori tersebut berhubungan dengan kategori sebelah kanan/atasnya maka :
   - Hapus kategori tersebut dari top stack
   - Push kategori tersebut yang baru ke dalam stack
4. Jika tidak ada lagi kata dalam kalimat tersebut, maka :
   - Jika kategori dalam stack = [S], maka selesai sudah.
5. Ulangi langkah ke step 2

BAB IV

ALGORITMA LEFT CORNER PARSING

Pemrosesan kalimat mengkombinasikan kata-kata atau ungkapan dalam sebuah kalimat untuk memperoleh interpretasi yang semestinya. Sebagai contoh beberapa kata keterangan ini berarti mengerjakan sesuatu dengan baik : *quickly, efficiently, effortlessly,* dan *accurately*. Tidak seperti mengatasi masalah matematis atau permainan catur yang memiliki solusi yang pasti, menempatkan kata tersebut sesuai dengan situasi yang ada adalah cukup sulit. Karena selain kompleks, dalam Bahasa Inggris banyak juga ditemukan kata *ambigius* , dimana satu kata bisa memiliki bermacam makna.

Pada bab ini dijelaskan tentang sifat *ambiguity* pada sebuah kalimat dan bagaimana algoritma *Left Corner parsing* yang dapat mengatasi permasalahan tersebut dibanding dengan Bottom Up atau Top Down Parsing. Juga dibahas beberapa alternatif solusi yang dapat dipakai untuk mengatasi permasalahan *recovering* selama terjadi proses parsing, yaitu *backtracking, parallellism,* dan *determinism*.

IV.1. Ambiguity

Ambiguity adalah satu macam kata yang memiliki lebih dari satu macam makna, tergantung pada konteksnya. Sebagai contoh, perhatikan kalimat berikut :

- I robbed the *bank (saya sudah merampok bank)*
- I fished from the *bank (saya memancing setelah dari Bank)*
- I *bank* with Loiyd *bank (saya melakukan bisnis bank dengan Bank Loiyd)*

- *Bank* the plan to turn it *(banting setir kapal itu untuk menurunkannya)*

Pada kalimat (1) dan (2), *bank* berarti sebagai kata benda. Kemudian pada kalimat (3), bank disini sudah sebagai kata kerja walaupun artinya masih berhubungan dengan kata bendanya. Namun, pada kalimat keempat, arti disini sudah lain sama sekali.

Sifat kata inilah yang disebut dengan *ambiguity*. Pada kalimat nomor 2 dan 3, arti di sini sudah bisa dibedakan dengan analisis sintaks, karena *bank* pada kalimat kedua merupakan kata benda, sedangkan pada kalimat ketiga merupakan kata kerja. Namun, pada kalimat ketiga dan keempat, pembedaan arti dari kata bank sudah mencapai tahap analisis pragmatis, karena keduanya merupakan kata kerja, namun artinya sudah berbeda tergantung dengan kondisi yang terjadi.

Dengan algoritma parsing Left Corner, hal ini bisa sedikit teratasi karena algoritma ini mampu memeriksa kata sampai pada analisis semantik. Jadi jenis dan arti kata tidak sekedar dari yang disediakan oleh satu database dimana kata tersebut berada, namun bisa meramalkan jenis dan arti kata tersebut dari database lain yang sesuai dengan tata bahasa yang dipakai pada kalimat tersebut

IV.2. Pengenalan Algoritma Left Corner

Algoritma *Top Down* and *Bottom Up Parsing* adalah dua algoritma parsing yang memiliki jangkauan yang kadang tak terbatas. Karena dalam Top Down Parsing, setiap kata harus diidentifikasi seluruhnya terlebih dahulu, termasuk yang tidak terdapat dalam reserved word (kata kerja, kata

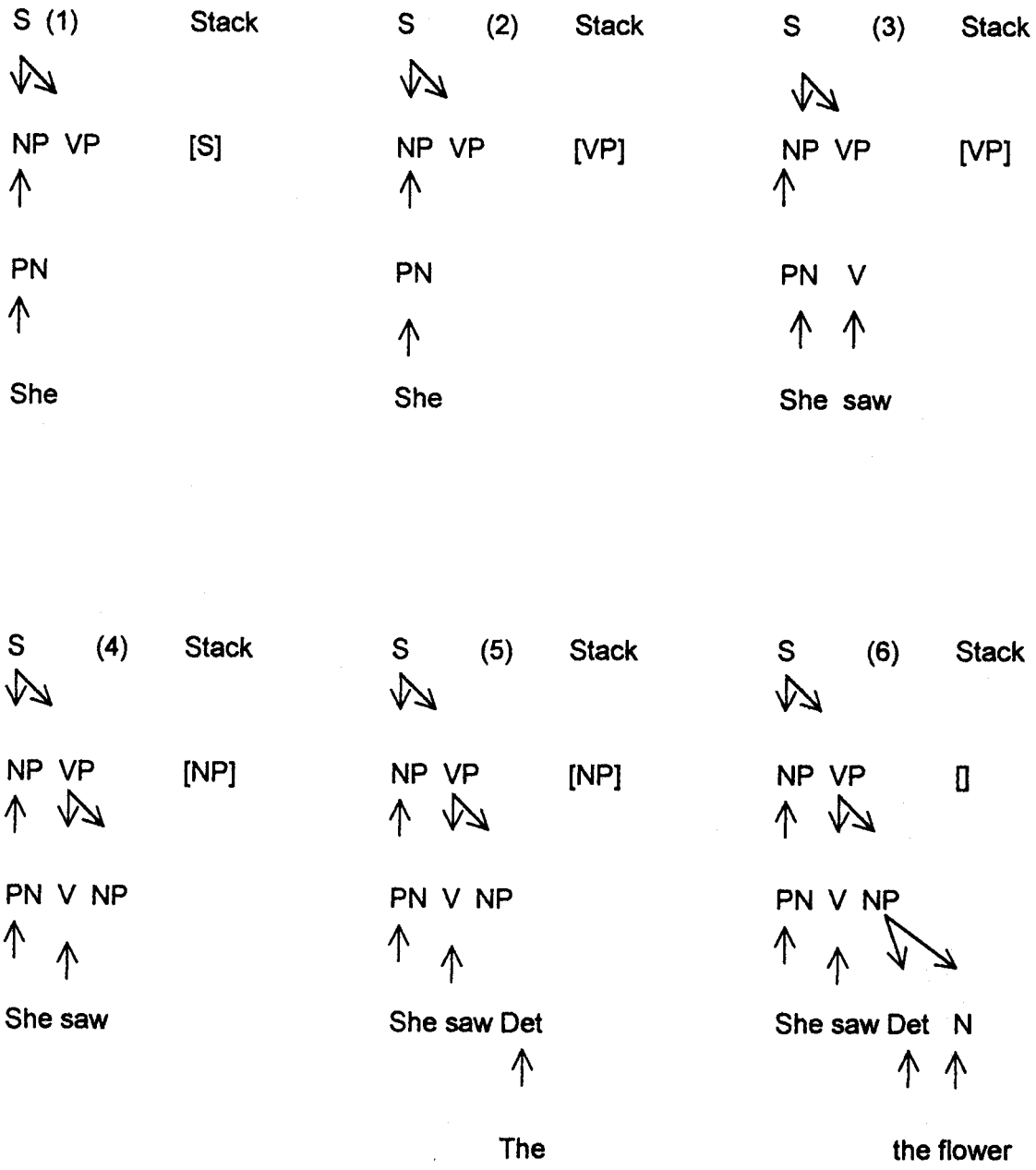
benda dan kata sifat). Sedangkan Bottom Up akan menghabiskan waktu lama dalam proses memilih jenis kata yang bisa lebih dari satu dalam sebuah kata, karena harus memilih salah satu dan mencobanya dalam parsing, sampai bertemu atau dinyatakan error.

Ide dasar dari algoritma *Left Corner Parsing* adalah untuk mengkombinasikan antara algoritma *Top Down* dan *Bottom Up Parsing*. Pusat perhatian dari algoritma ini adalah penggunaan 'sudut kiri' dari frase (simbol paling kiri pada sisi kanan *productions*). Cara ini menggunakan sebuah stack yang hanya berisi satu buah simbol terminal atau nonterminal.

Algoritma ini dapat digambarkan sebagai berikut. Pertama kali dicari hasil *productions* dalam proses Top Down (S), kemudian diprediksikan dengan *production* S --> NP VP. Kemudian dicari sudut kiri dari NP, S dalam stack diganti dengan prediksi kata selanjutnya (sebelah kanannya), yaitu VP. VP dicari Left Cornernya, misalnya V sampai ketemu Verb. Kemudian VP diganti dengan *child* sebelah kanannya yaitu NP. Stack VP diganti dengan NP dan seterusnya.

Proses menemukan VP sampai Verb, atau NP sampai Det disini menggunakan gabungan Top Down dan Bottom Up karena pencocokannya ditelusuri dari jenis kata-kata yang ada dalam kalimat.

Sebagai contoh, perhatikan parsing pada kalimat 'She takes the flower' berikut :



Gambar IV.1 Diagram proses *Left Corner Parsing*

Pada tahap (1), beberapa hal yang dapat diamati adalah : pertama, dibuat asumsi model top-down yang dibentuk sebagai sebuah struktur kalimat, dan S ditempatkan pertama pada stack. Kemudian di dalam bottom up, telah

terdapat nonterminal NP, hasil dari penurunan PN (Proper Noun) dari kata ganti orang 'She' di *Bottom Up stack.*

Dalam tahap (2) dapat dilihat aturan Left Corner : yaitu dengan membuat production S → NP VP, lalu menjadikan NP sebagai sudut terkiri (Left Corner) dari S. NP ini kemudian dihapus dari stack karena bertemu dengan kontestan yang cocok, dan isi stack diganti dengan VP yang akan dicari selanjutnya kemudian.

Pada tahap (3), digambarkan bahwa telah ditemukan sebuah kata kerja di bottom up. Dan langkah (4) juga digunakan Left Corner untuk menghubungkan V dengan VP, dan kemudian memprediksikan NP (sesuai dengan rule VP → V NP). Setelah itu, VP dihapus dari stack, diganti dengan frase NP yang baru. Kemudian diteruskan dengan menemukan sebuah DET, yang merupakan left-corner dari NP dan seterusnya sampai terakhir N bertemu dengan kata flower. Di sini proses Left Corner berakhir dan mengembalikan nilai true, karena stack di *Top Down* $\in$ (empty) dan *Bottom Up* menunjukkan index kata yang terakhir.

IV.3. Reserved Word

Algoritma Left Corner akan membagi jenis kata menjadi empat macam yaitu : reserved word, verb, noun, dan adjective. *Reserved word* adalah kata yang sudah terdaftar dalam tabel kata kunci. Kata kunci di sini digunakan untuk mengenali jenis kata yang seharusnya ada di sekitarnya

menurut aturan grammar yang ada. Dan *verb, noun* dan *adjective* inilah adalah jenis kata yang akan dicari oleh parser. Sebagai contoh :

NP → DET NON

Adalah *production* yang menyatakan bahwa frase kata benda (*Noun Phrase*) dapat dibentuk dari kata sandang (*Determiner*) dan kata benda terhitung (*Numbered Noun*). DET disini sebagai *reserved word*, sedangkan NON adalah kata benda yang harus dicari oleh parser.

Dengan menggunakan strategi *reserved word*, akan dapat dipilih salah satu yang paling tepat dari beberapa jenis kata yang dimiliki oleh suatu kata yang ambigius. Sebagai contoh, kata 'have' berikut akan memiliki beberapa jenis dan makna dalam bentuk kalimat yang berbeda :

I *have to* be a *have*
(saya harus menjadi orang punya=kaya)
I *have* received a flower from a girl
(saya telah menerima sebuah bunga dari seorang gadis)
I don't *have* anything to give.
(saya tak memiliki apapun yang bisa saya berikan)

Pada kalimat (1) have pertama menduduki posisi kata kerja bantu yang digabung dengan *to* membentuk arti 'harus'. Sedangkan kata kedua membentuk kata benda yang berarti 'orang kaya'. Sedangkan kalimat (2) *have* menyatakan kata bantu yang bermakna membentuk kalimat bentuk lampau. Dan pada kalimat (3) *have* menyatakan kata kerja yang bermakna memiliki.

Reserved word dalam kalimat di atas, *(I, to, not)* yang dikombinasikan dengan aturan grammar yang ada akan dapat menentukan jenis kata yang paling tepat untuk kata *have*. Apakah kata tersebut bermakna kata kerja bantu, kata bantu, kata benda atau kata kerja biasa.

IV.4. Metode Recovering dalam Proses Parsing

Sebuah kalimat sangat memungkinkan bisa memiliki lebih dari satu analisis sintaks, mungkin pada frase tunggal, atau pada ungkapan yang berada di balik kalimat tersebut. Dalam contoh di atas, walaupun NP telah dijabarkan dengan menggunakan *production* NP → Det N, yang kondisinya (kebetulan) benar. Tapi hal itu tidak selamanya benar untuk kasus-kasus yang lain. Karena bisa jadi NP tersebut mewakili sebuah ungkapan lain, seperti 'my address' yang merupakan hasil production NP → POSS N, dimana POSS adalah *possessive word*. Supaya memilih sebuah production parsing dapat ditemukan dengan benar, tiga cara pokok yang dapat diambil antara lain :

1. *Backtracking*. Ketika lebih dari satu struktur *production* yang bisa dibuat, maka dipilih salah satu, dan ditandai bahwa keputusan itu sebagai titik pilihan. Jika analisis tersebut gagal, dipilih sebuah struktur yang berbeda, dan dicoba lagi selanjutnya. Diproses terus-menerus sampai sebuah parse ditemukan, atau kemungkinannya habis. Jika sampai habis tidak ditemukan/cocok, maka sintaks atau rule kalimat tersebut tidak benar.
2. *Parallellism* : menelusuri semua kemungkinan secara parallel, seperti ketika lebih dari satu struktur yang bisa dibuat, buat semuanya secara bersama-sama, dan dibatalkan rule lain yang tidak benar.
3. *Determinism* : diawali dengan memberikan *parsing* informasi yang memadai, sehingga bisa diputuskan dengan benar rule yang harus dipilih. Hal ini berarti parser tidak akan membuat beberapa kesalahan, sehingga tidak akan dilakukan *parallelism* atau *backtracking*.

BAB V

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas tentang perancangan sistem Perangkat Lunak (PL) yang dipakai untuk melakukan proses scanning dan parsing terhadap input string sehingga menghasilkan output yang diharapkan. Perancangan ini meliputi desain struktur data, data flow diagram sistem, algoritma, dan flow chart, bahasa pemrograman yang dipakai, implementasi unit - unit program dan diakhiri dengan implementasi prosedur utama.

V.1. Perancangan Sistem

Perancangan sistem yang dibuat dalam PL ini menggunakan sistem pemrograman orientasi obyek (Object Oriented Programming), sehingga dalam perancangan ini dibentuk pula dengan *OOD* (Object Oriented Design). Sistem orientasi obyek ini akan memungkinkan sebuah obyek tidak saja terdiri dari data, seperti sebuah record, namun juga bisa melakukan fungsi dan prosedur.

Penggunaan sistem obyek dalam bahasa pemrograman dapat dibentuk dengan **class**. Class adalah sebuah struktur yang terdiri dari field-field, method, dan property yang disebut dengan komponen atau member (anggota). Field adalah data yang merupakan bagian dari class, sedangkan method adalah prosedur atau fungsi. Sedangkan properti lebih mendekati field, namun lebih didasarkan pada data-data yang menempel dan bisa dimodifikasi.

Obyek secara dinamis akan mengambil sebuah blok memory, yang besarnya tergantung dari class itu sendiri. Alokasi memory ini akan diciptakan

oleh fungsi khusus yang disebut destructor dan constructor yang ada di obyek tersebut. Kemudian juga memiliki sifat visibility yang ditandai dengan private, protected, public, published atau automated [Borland Delphi Help Files].

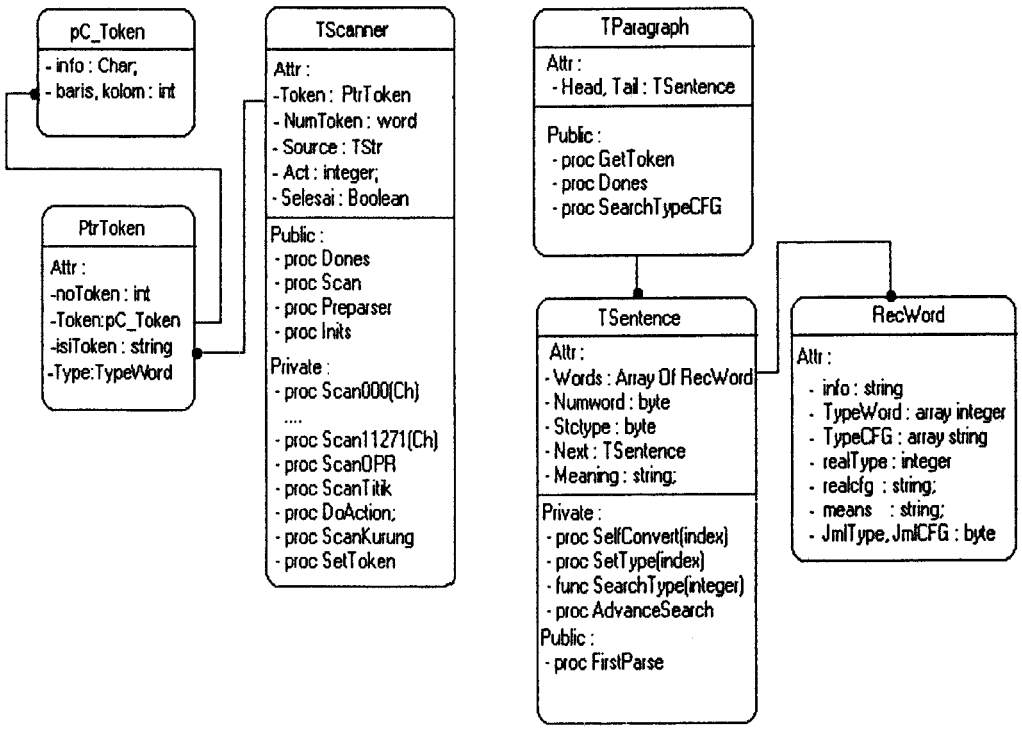
V.2. Perancangan Class

Beberapa class yang dibentuk dari proses-proses yang terdapat dalam tugas akhir ini adalah :

1. Class Tparagraph, digunakan untuk menyimpan obyek paragraf yang terdiri dari obyek kalimat, dan beberapa fungsi yang terkait seperti Get_Token dan Search_Type_CFG.
2. Class Tsentence, digunakan untuk menyimpan obyek kalimat yang terdiri dari unit-unit data kata, dan beberapa fungsi atau prosedur seperti Self_Convert, Advance_Search, dan beberapa fungsi yang mestinya diletakkan dalam class kata (kata berbentuk record) seperti Set_Type_Word.
3. Class Tscanner, digunakan untuk menyimpan fungsi-fungsi dan atribut yang digunakan untuk proses scanning string sampai terbentuk token.
4. Class Tparsers, digunakan untuk menyimpan tungsi-fungsi dan atributnya yang digunakan untuk menjalankan proses *Left Corner Parsing.*

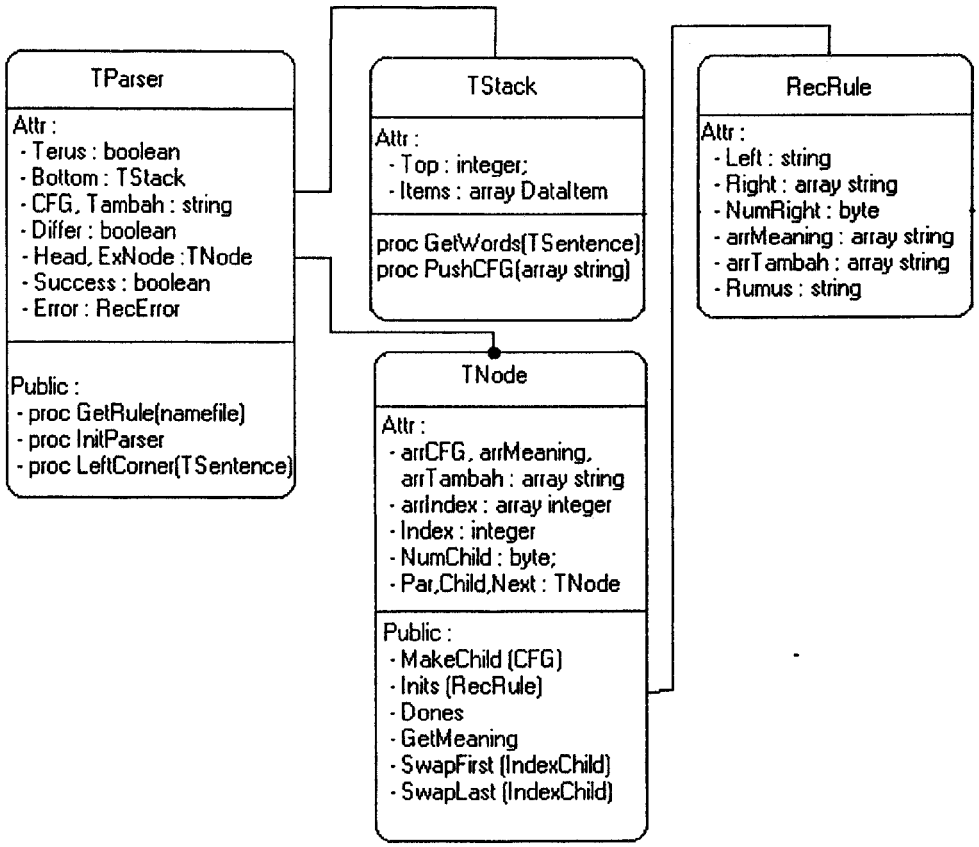
5. Class Tstack, adalah class yang digunakan untuk menyimpan stack pada proses parsing Bottom Up yang nantinya digunakan untuk membantu jalannya proses Left Corner Parsing.
6. Class Tnode, adalah class yang berisi rule-rule dan CFG dan membentuk tree dinamis, yang akan digunakan saat dilakukan proses Left Corner Parsing.

Bentuk dari diagram class Tscanner, TSentence dan TParagraph adalah :



Gambar V.1 Diagram class Scanner

Diagram class TParser



Gambar V.2 Gambar diagram class Parser

V.3. Perancangan data

Data adalah suatu kumpulan karakter dan informasi yang memiliki arti tertentu. Data yang dibutuhkan dalam Perangkat Lunak (PL) ini dibagi menjadi tiga yaitu : perancangan data input, data output dan data proses.

V.3.1. Data input

Data input yang dibutuhkan dalam perangkat lunak ini berupa string kalimat utuh Bahasa Inggris. Tidak bisa menerima frase atau kata tunggal. Kalimat ini dibagi menjadi 5 bagian, yaitu :

1. Kalimat sederhana/berita, yaitu kalimat yang terdiri dari subyek, predikat, obyek dan kata keterangan.
2. Kalimat majemuk, yaitu kalimat yang memiliki anak kalimat di dalamnya.
3. Kalimat perintah, yaitu kalimat yang diawali oleh kata kerja bentuk pertama dan (biasanya) diakhiri dengan tanda seru.
4. Kalimat pertanyaan, yaitu kalimat yang diakhiri dengan tanda tanya.
5. Kalimat seru, yaitu kalimat yang terdiri dari kata-kata seru (keterkejutan).

V.3.2. Data output

Data output dalam PL ini akan membentuk obyek paragraf (class Tparagraph) yang akan menyimpan beberapa (satu atau lebih) kalimat (class TSentence) hasil proses PL sesuai dengan data input yang diberikan. Obyek Tsentence juga menyimpan array kata yang sudah memiliki atribut tertentu hasil dari proses parsing. Beberapa fungsi penting dari proses *Left Corner Algorithm* akan disimpan di dalam atribut obyek TSentence.

Dua hasil proses penting tersebut antara lain adalah :

V.3.2.1. Penentuan jenis dan arti kata.

Jenis dan arti kata ini bisa diperiksa sebelum dan sesudah melakukan proses Left Corner Parsing. Pada menu PL dilakukan oleh

berbentuk sebuah tabel token. Dalam tabel tersebut akan terlihat perbedaan jenis kata sebelum proses parsing dan sesudah proses parsing. Dalam beberapa kata dan kasus tertentu, jenis kata hasil definisi awal (proses scanning), akan berbeda dengan jenis setelah proses akhir (setelah parsing).

Kemudian untuk atribut jenis dan arti kata ini akan disimpan dalam array record *words* yang berada dalam obyek TSentence.

V.3.2.2. Penentuan makna kalimat.

Penentuan makna kalimat ini dilakukan oleh *Left Corner Parsing* secara *recursive* dan kemudian akan disimpan kembali dalam atribut 'means' dalam obyek TSentence. Hasil output arti kalimat ini akan menampilkan arti per kata dan per kalimat.

Sebelum dilakukan proses *Left Corner Parsing*, terlebih dahulu dilakukan pemeriksaan sintaks. Sehingga, jika pada proses ini sudah ditemukan kesalahan, maka tidak akan dilakukan proses *Left Corner Parsing*.

V.3.3. Data proses

Data proses digunakan untuk membantu jalannya proses-proses yang dilakukan oleh perangkat lunak. Data-data yang dibutuhkan yang dibagi menurut prosesnya antara lain :

V.3.3.1. Jenis-jenis token

Jenis-jenis token dibutuhkan pada saat proses scanning. Masing-masing karakter yang dikelompokkan menurut entitasnya

diidentifikasi dalam bentuk token-token. Jenis-jenis token ini disimpan dalam bentuk bilangan integer 0 sampai dengan 20 yang dikodekan dengan konstanta agar mudah dibaca.

V.3.3.2. Jenis-jenis kata

Dari proses tokenisasi, maka dihasilkan entitas-entitas yang salah satunya berjenis token kata. Token kata ini kemudian diproses ulang untuk diidentifikasi kembali sehingga menjadi jenis-jenis kata. Jenis kata ini disimpan dalam record kata dalam bentuk bilangan integer yang bisa dikodekan secara biner sampai tigabelas karakter.

Satu bentuk kata dalam sebuah bahasa bisa memiliki lebih dari satu macam jenis. Oleh karena itu harus tipe kata ini harus disimpan dalam bentuk array. Adapun struktur data yang menyimpan kata adalah :

```
Words = record
  info    : string; menyimpan string kata
  Tyword  : Array of integer; kode integer
  TyCFG   : Array Of String;  kode string (CFG)
End;
```

Ketiga belas karakter biner yang merupakan kombinasi kode jenis kata ini memiliki arti sebagai berikut :

XXXX	*empat karakter I menandai jenis kata utama*
XXX	*tiga karakter II menandai sub jenis kata utama*
XXX	*tiga karakter III menandai sub dari sub jenis*
XXX	*tiga karakter terakhir menandai jenis kata dari kata utama yang tidak memiliki hubungan dengan kedua kelompok karakter sebelumnya*

Sebagai contoh kata 'being' memiliki jenis kata : 716. Jika angka ini dikodekan menjadi bilangan biner, maka akan menjadi : 0001 011 001 100 dimana :

0001	*merupakan kode verb (kata kerja)*
011	*kode transitive verb (kata kerja berobjek)*
001	*kode direct transitive verb (kata kerja berobjek langsung)*
100	*kode kata kerja bentuk ing*

Jenis kata ini disimpan dalam database dalam bentuk file DATAWORD.DBF. File ini memiliki lima field yaitu :

STR_CODE	*jenis kata bentuk string (4 karakter)*
NT_CODE	*jenis kata dalam bilangan integer*
BIN_CODE	*jenis kata dalam bentuk string biner*
REMARK	*penjelasan jenis kata tersebut*

V.3.3.3. Jenis-jenis kalimat

Jenis kalimat dalam tugas akhir ini dibedakan menjadi 5 (lima) macam yaitu :

1. Statement	*diakhiri tanda titik*
2. Question	*diakhiri tanda tanya*
3. Desire	*diakhiri tanda seru atau diawali kata kerja bentuk I*
4. Exclamation	*diakhiri tanda seru atau diawali kata seru*
5. Clause	*diawali frase preposisi*

Data-data jenis kalimat ini disimpan dalam file *dbSTC.dbf* yang akan digunakan untuk pengenalan jenis kalimat pada string input. Jenis kalimat ini bisa diketahui pada saat inisialisasi parsing, yang bisa diperiksa berdasarkan kata pertama dan tanda baca terakhir dari token-token yang dihasilkan. Dengan cara seperti ini, maka akan bisa langsung

digunakan CFG yang bersesuaian dengan jenis kalimat tersebut, sehingga bisa menghemat waktu proses parsing.

V.3.3.4. Basis data leksikal (Lexicon Database)

Lexicon database dalam PL ini akan dibagi menjadi empat bagian yaitu: *reserved word* atau leksikal pokok (default lexicon), kata kerja (verb lexicon), kata benda (noun lexicon) dan kata sifat (adjective lexicon). Data-data leksikal ini disimpan dalam file **dbDefault.dbf, dbVerb.dbf, dbNoun.dbf**, dan **dbAdj.dbf**. Keempat data leksikal ini memiliki struktur data yang sama yaitu :

```
INFO      : menyimpan informasi kata
TYPEWORD  : menyimpan jenis kata dalam integer
MEANING   : menyimpan arti kata dalam bahasa
            Indonesia
```

Ketika menjalankan proses, maka data leksikal ini akan ditransfer ke table *dbDefa* yang akan digunakan untuk pencarian kata ketika proses parsing berlangsung. Misalnya, ketika pertama proses parsing dibutuhkan DbDefa secara default menunjuk ke DbDefault.DBF, namun ketika membutuhkan kata benda, maka dbDefa akan di-*switch* ke dbNoun.DBF, dan seterusnya.

Leksikal Pokok ini bisa disebut juga dengan *Reserved Word*. Artinya, sebelum melakukan proses parsing, maka semua kata input diidentifikasi terlebih dahulu dengan kata yang ada di daftar leksikal pokok. Kemudian, setelah melakukan proses parsing left corner, maka dapat ditentukan lagi, apakah penentuan dari leksikal pokok tersebut

sesuai dengan grammar atau tidak. Kalau tidak, maka dicari alternatif lain di dalam kata kerja, kata benda atau kata sifat.

V.3.3.5. Context Free Grammar

CFG adalah sekelompok *rule* yang digunakan untuk memeriksa kebenaran tata bahasa terhadap kalimat yang diinputkan. CFG ini disimpan dengan kode ciri empat karakter sebagai pembeda dengan CFG yang lain, kecuali yang terpaksa tidak bisa dijadikan empat karakter seperti kata kerja ing (ING_VERB). Sebagai contoh :

```
DIRE_TRAN_VERB  Direct Transitive Verb
INDI_TRAN_VERB  Indirect Transitive Verb
PROP_NOUN       Proper Noun
```

Dalam perangkat lunak ini dibuat *rule* yang bertingkat yang memungkinkan satu buah posisi kata memiliki lebih dari satu syarat (di sini masih dibatasi dua). Sebagai contoh :

```
I am buying a souvenir for my friend.
```

kata *buying* di atas memiliki dua syarat : (1) Harus berbentuk kata kerja ing karena mengikuti *tobe* bentuk pertama, (2) Harus merupakan kata kerja *Direct Transitif Verb* karena diikuti oleh frase kata benda 'a souvenir'. Jadi bentuk CFG-nya adalah :

```
S1 → NP tobe ing_VERB
S2 → NP tobe VP
VP → DIRE_TRAN_VERB NP AVL PPL
```

Jika satu kata tersebut hanya memiliki satu syarat, maka kedua rule harus berisi type CFG yang sama. Misalnya, rule pertama berisi PREP, maka rule kedua juga berisi PREP. Dan, kedua rule tersebut menempati posisi yang sejajar.

Dalam PL ini, rule pertama disimpan dalam file CFG.dat dan rule kedua disimpan dalam file Tambah.dat dalam bentuk file teks. Format penyimpananya adalah seperti sebuah production rule biasa yaitu :

```
S>NP VP
VP>TRAN_VERB NP AVL PPL
... dan seterusnya.
```

S dan VP adalah nonterminal dan tanda '>' adalah pemisah antara nonterminal dan *production*-nya. Sedangkan sebagai pemisah antara CFG hasil production adalah tanda spasi. Kedua file ini akan di-load bersama dalam sebuah array rule dengan menggunakan metode scanning, yang akan memeriksa posisi karakter berdasarkan tanda '>' dan spasi sehingga bisa dibedakan antara *nonterminal*, *terminal* dan *production*. Kemudian hasil parsing ini akan disimpan dalam array record dengan struktur data seperti ini:

```
RecRule = record
  Left : string           Non Terminal
  Right : ArrTypeCfg      CFG Utama
  NumRight : byte         jumlah CFG
  arrTambah : arrTypeCfg  CFG tambahan
  arrMeans : ArrTypeCfg   arti kata tiap CFG
  rumus : string          rumus arti dari CFG
end;
```

V.3.3.6. Pesan Kesalahan (Error message)

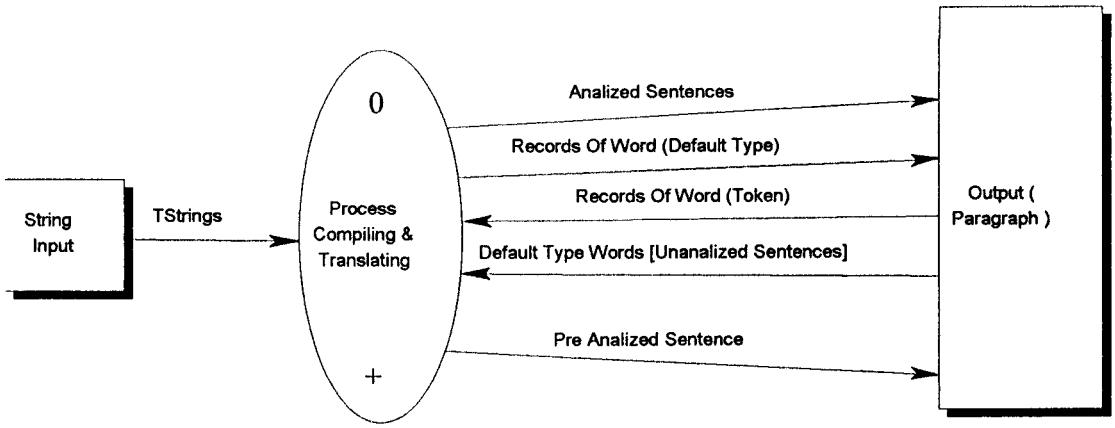
Jenis pesan-pesan kesalahan (error) disimpan dalam file tersendiri yaitu : msg.err. Pesan-pesan error ini akan diload ke dalam memory dalam array record yang berbentuk:

```
RecError = Record
   Bar   : integer ; baris karakter
   Kol   : Integer ; kolom karakter
   Kar   : Char    ; jenis karakter
   Data  : String  ; informasi error
   Jenis : byte    ; nomor informasi error
end;
```

V.4. Perancangan Proses

Dalam perancangan ini ditunjukkan alur proses yang dilakukan untuk mendapatkan output Tsentence dari input string yang diberikan. Kedalaman proses dilakukan sampai pada level 2. Pembuatan perancangan proses ini dilakukan dengan menggunakan Data Flow Diagram yang dibagi menjadi beberapa level, antara lain :

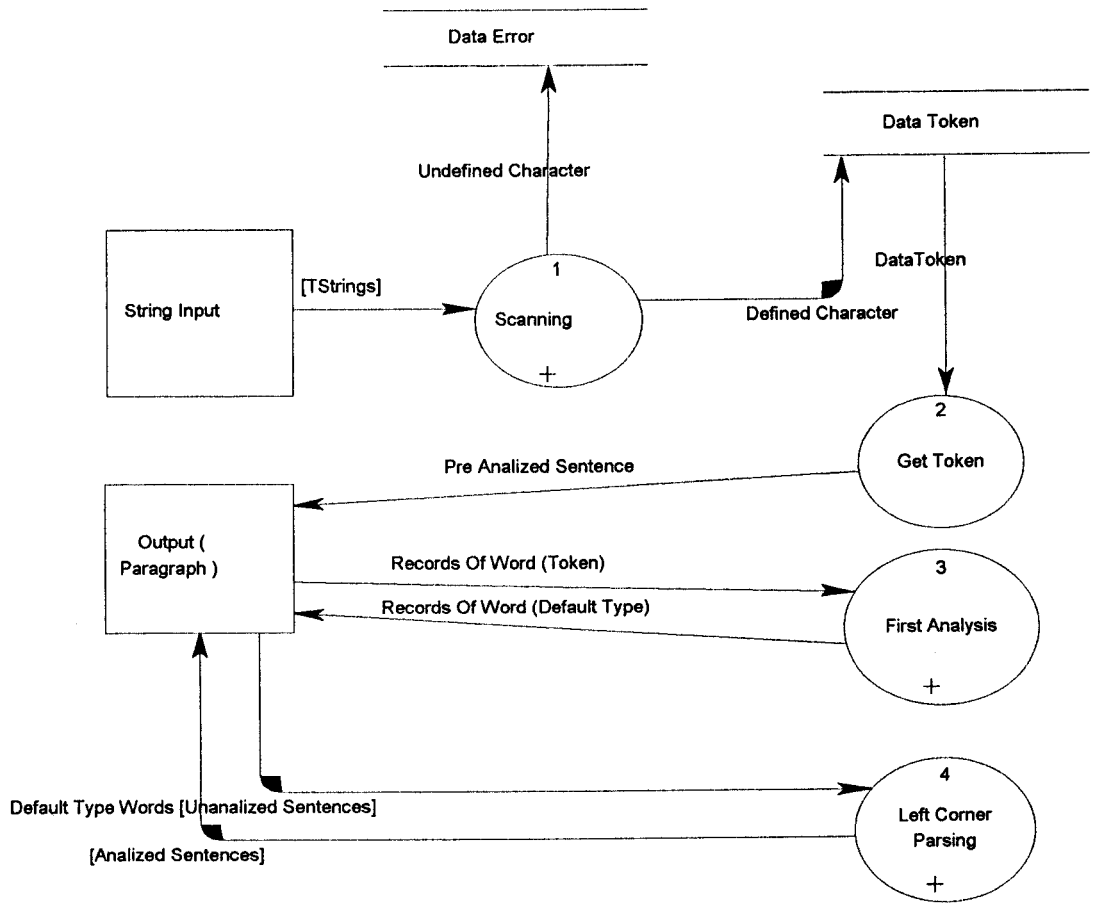
V.4.1. DFD Level 0



Gambar V.3 DFD level 0

Pada level ini, akan dapat dilihat gambaran proses global dimana input yang berupa string akan diolah oleh proses compiling dan translating sehingga menjadi sebuah output obyek paragraph yang sudah dikenali masing-masing kata penyusunnya. Banyaknya lalu lintas data yang menghubungkan obyek paragraf dan proses compiling dan translating tersebut menggambarkan bahwa proses pembentukan obyek paragraf ini dilakukan berulang-ulang.

V.4.2. DFD Level 1



Gambar V.4 DFD level 1

Dakam DFD level 1 ini terlihat beberapa proses global yang membentuk kesatuan proses compiling dan translating. Tiga proses utama yang ada di dalamnya adalah Scanning, GetToken, FirstAnalysis, dan penerapan Left Corner Parsing Algorithm.

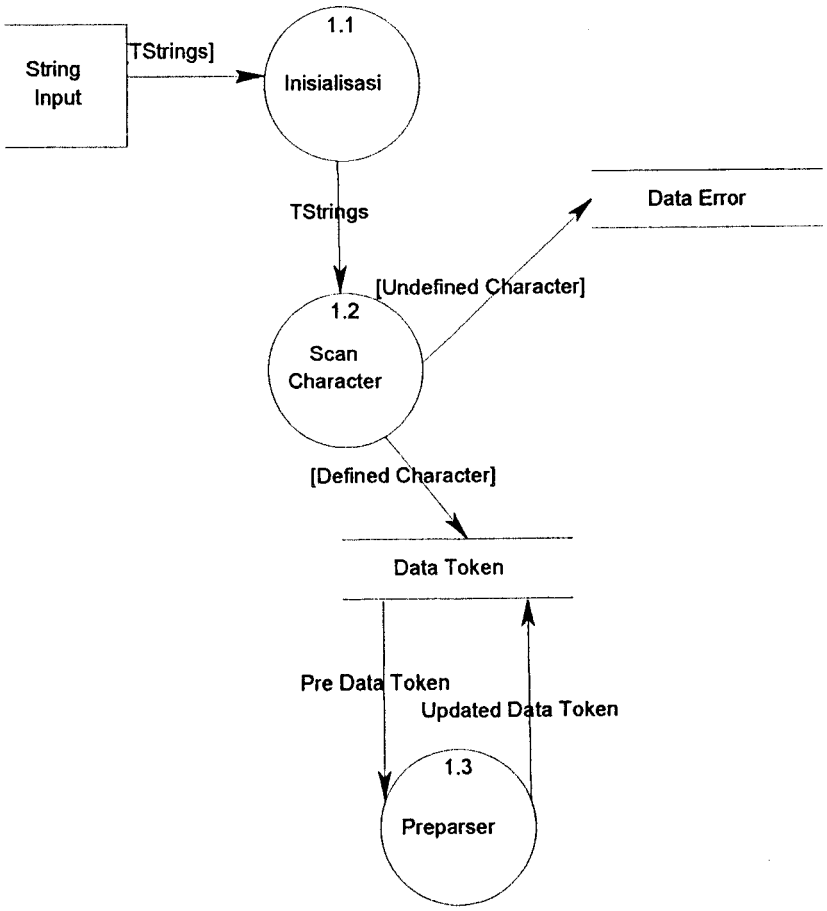
Proses scanning, adalah menganalisis string sehingga membentuk entitas-entitas token. Sedangkan GetToken akan mentransfer token-token

tersebut ke dalam obyek paragraph yang terdiri dari kalimat-kalimat di dalamnya, namun jenis katanya masih dalam bentuk token.

First Analysis akan mendifinisikan jenis-jenis kata berdasarkan basis data leksikal pokok (*default lexicon database*) yang telah disediakan. Jenis ini meliputi jenis kata yang bertipe integer dan string, dan dalam satu kata yang sama bisa memiliki jenis lebih dari satu macam. Jika kata-kata tersebut tidak ditemukan dalam Default Lexicon, maka kata tersebut didefinisikan sebagai Unknown Word yang berkode string 'UNKN' dan berkode integer 0.

Setelah proses-proses di atas dilalui, barulah dilakukan tahap parsing dengan metode Left Corner. Proses ini akan mencari, memilih dan menentukan jenis kata-kata yang masih bertipe 0 tersebut dari semua database *lexicon* yang disediakan berdasarkan struktur grammar yang dikenal dalam Bahasa Inggris. Yang dalam PL ini disimpan dalam rule CFG. Sehingga bisa memeriksa kebenaran sintaks yang dipakai, dan juga menentukan jenis dan arti kata-kata yang menyusunnya.

V.4.3. DFD Level 2



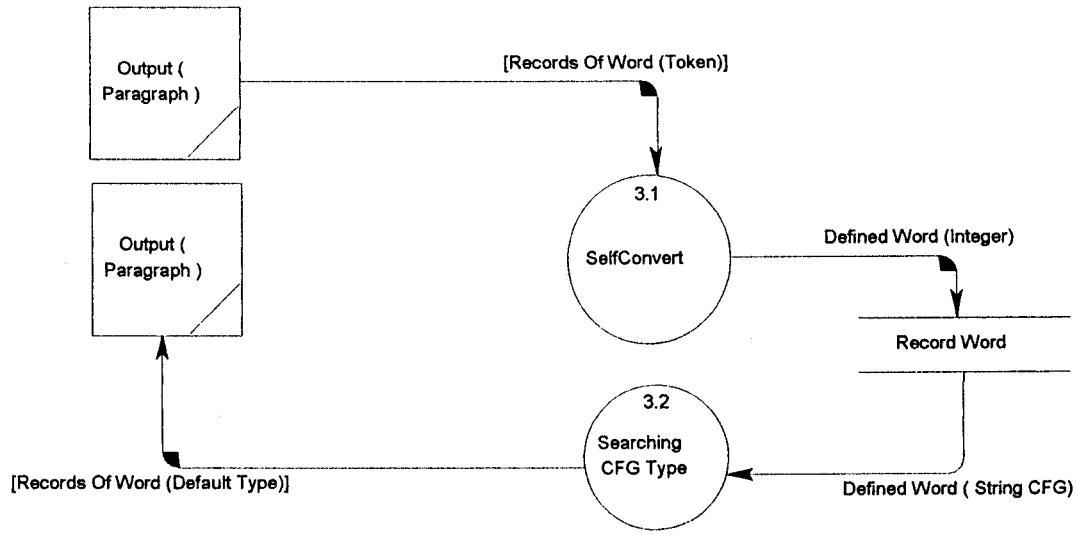
Gambar V.5 DFD level 2 Proses Nomor 1

Dalam diagram di atas, dapat terlihat bahwa proses scanning tersebut terbagi menjadi tiga proses utama yaitu Inisitalisasi, Scan Character dan Preparser.

Proses Inisialisasi akan dilakukan jika akan melakukan proses scanning terhadap input string yang baru, walau secara langsung tidak memproses string tersebut. Kemudian, dalam proses scanning selanjutnya

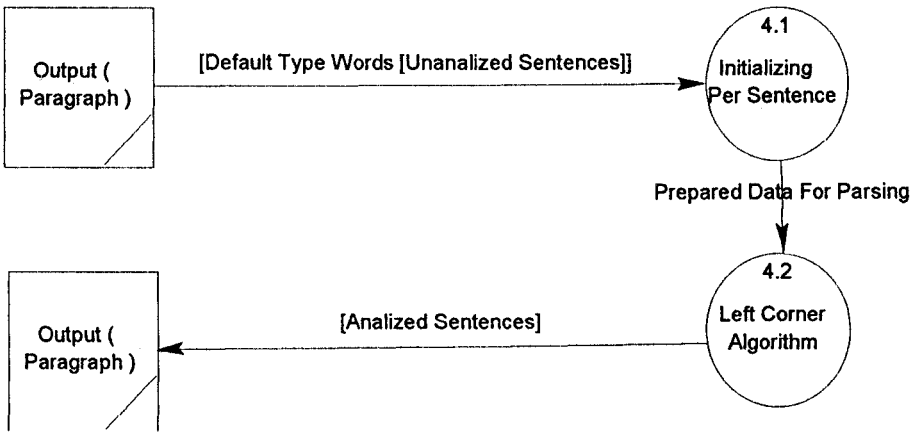
string input sudah dipisah-pisahkan sesuai dengan standard pemisahnya (*delimiter*), sehingga membentuk unit-unit token yang disimpan dalam bentuk list data token. Jika dalam scanning tersebut ditemukan kesalahan, proses akan langsung berhenti dan error disimpan ke dalam data error.

Data token hasil proses scanning ini masih diproses lagi oleh preparser untuk mendapatkan token-token baru hasil dari penggabungan beberapa token, seperti token tanggal, waktu, persamaan atau kata singkatan, yang kemudian dimasukkan lagi ke data token.



Gambar V.6 DFD level 2 Proses Nomor 3

First Analysis ini terbagi menjadi dua buah proses yaitu Self Convert dan Searching CFG Types. Self Convert akan mengubah kode-kode token menjadi kode-kode jenis kata dalam bentuk kode integer. Sedangkan Searching CFG akan menentukan bentuk kode string dari kode integer tersebut yang akan digunakan dalam proses parsing nantinya.



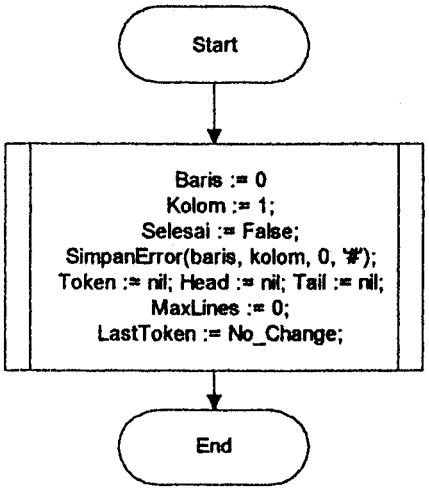
Gambar V.7 DFD Level 2 Proses Nomor 4

Setelah semua kata telah didifinisikan tipe CFG-nya (yang diketahui maupun yang tidak), maka bisa dilakukan proses parsing Left Corner. Proses ini didahului dengan proses inisialisasi, baru kemudian menjalankan proses parsing Left Corner.

V.5. Perancangan Algoritma

Perancangan algoritma disini akan digambarkan dalam bentuk *flow chart,* yang merupakan penjabaran dari DFD dalam sub bab sebelumnya. Yaitu, penjabaran dari proses-proses level terdalam yang ada dalam lingkaran proses DFD tersebut.

Proses pertama kali yang dilakukan adalah inisialisasi scanning. Inisialisasi pada proses scanning ini dilakukan untuk menandai bahwa proses scanning belum dilakukan. Hal ini dengan mengeset baris, kolom, error, token, dan seterusnya dengan posisi awal scanning.



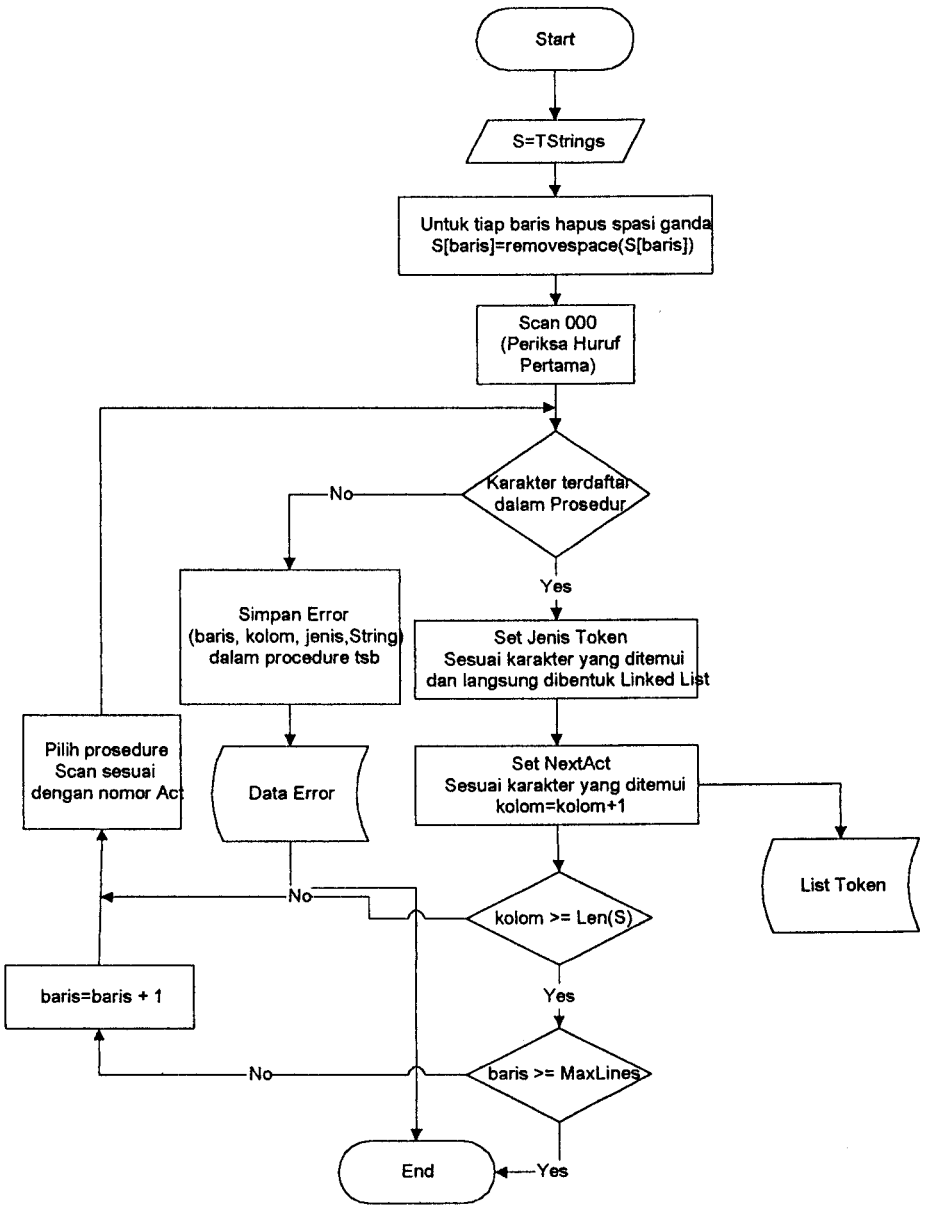
Gambar V.8 Proses 1.1 Inisialisasi Scanning

Kemudian, barulah dilakukan proses parsing, dengan syarat input sudah tersedia (tidak empty) yang bertipe TStrings. Input ini kemudian diproses lebih lanjut dengan menghilangkan spasi ganda. Hal ini digunakan agar mempermudah proses pemisahan token dalam proses scanning.

Setelah input siap, maka dilakukan proses scanning yang sebenarnya. Proses scanning ini terdiri dari banyak modul yang memiliki sifat

sejenis, yaitu memiliki satu set karakter dan nomor *Action* yang unik (unique). Masing-masing karakter ini digunakan untuk memeriksa jenis token yang terbentuk, sedangkan Action digunakan untuk menunjuk prosedur Scan selanjutnya.

Bentuk diagram alurnya dapat dilihat di bawah ini.

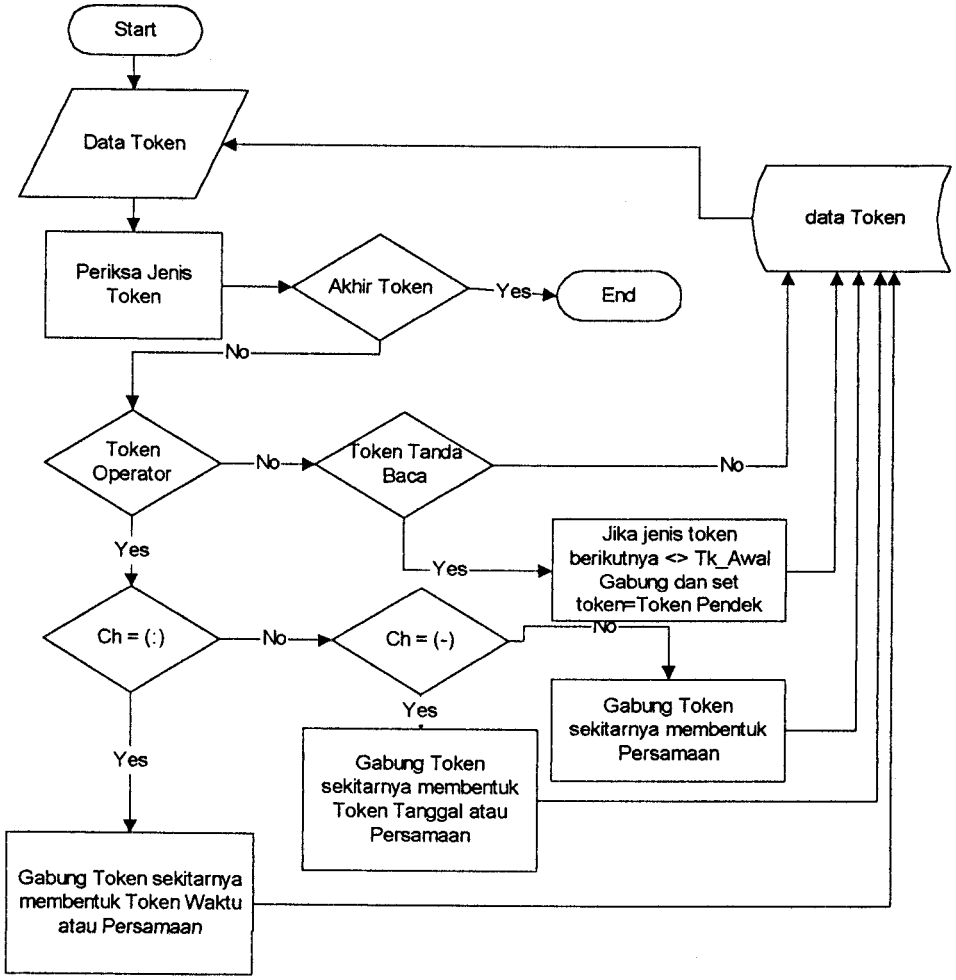


Gambar V.9 Proses 1.2 Scanning String

Prosedur scan ini akan dimulai dari prosedur pertama. Kemudian, karakter yang di scan akan dibandingkan dengan set karakter yang terdaftar dalam prosedur tersebut. Jika tidak ditemukan, maka dianggap error dan langsung keluar prosedur. Jika karakter tersebut ditemukan, maka akan menuju ke prosedur yang ditunjuk oleh nomor Action dan membentuk jenis token yang sesuai. Token-token ini akan otomatis berganti dan membentuk token baru jika bertemu dengan karakter yang membentuk jenis token yang berbeda dari sebelumnya.

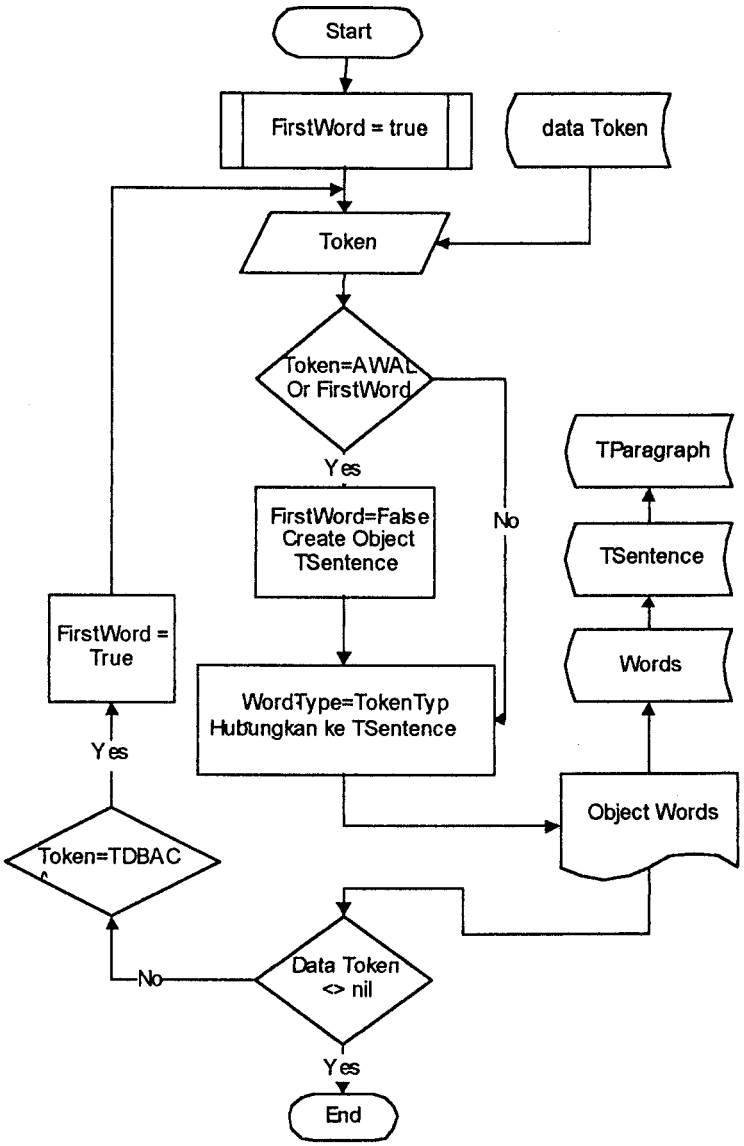
Setelah terbentuk daftar token (list of token), maka dilakukan analisis lagi untuk menggabungkan satu atau lebih token menjadi sebuah token baru. Misalnya adalah token tanggal, waktu, persamaan dan singkatan. Proses ini bisa disebut juga dengan preparsing, atau awal proses parsing.

Bentuk diagram alurnya dapat dilihat di bawah ini.



Gambar V.10 Proses 1.3 Preparsing

Setelah proses pembentukan token selesai, maka *list* token ini dimasukkan ke dalam obyek Tsentence, yaitu dengan prosedur GetToken. Diagram alurnya dapat dilihat di bawah ini.

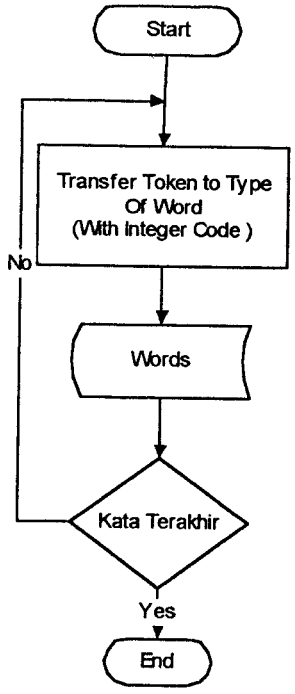


Gambar V.11 Proses 2 Get Token

Di sini, setiap kali ditemukan token yang menandakan kata awal kalimat (huruf depan besar dan didahului titik atau awal paragraf), maka dibentuk obyek kalimat (Tsentence) baru.

Tipe kata dalam prosedur ini masih berupa tipe token, untuk itu proses selanjutnya adalah mentransfer jenis-jenis token ini menjadi jenis-jenis kata.

Seperti token angka menjadi *Quantitive Adjective*; token variabel, nama, singkatan dan persamaan menjadi *Proper Noun*, token jam dan tanggal menjadi *Adverb Of Time* dan token biasa akan dicari dalam database *Default Lexicon*. Dan, jika tidak ditemukan akan dikenal sebagai *Unknown Word* atau bertipe 0.

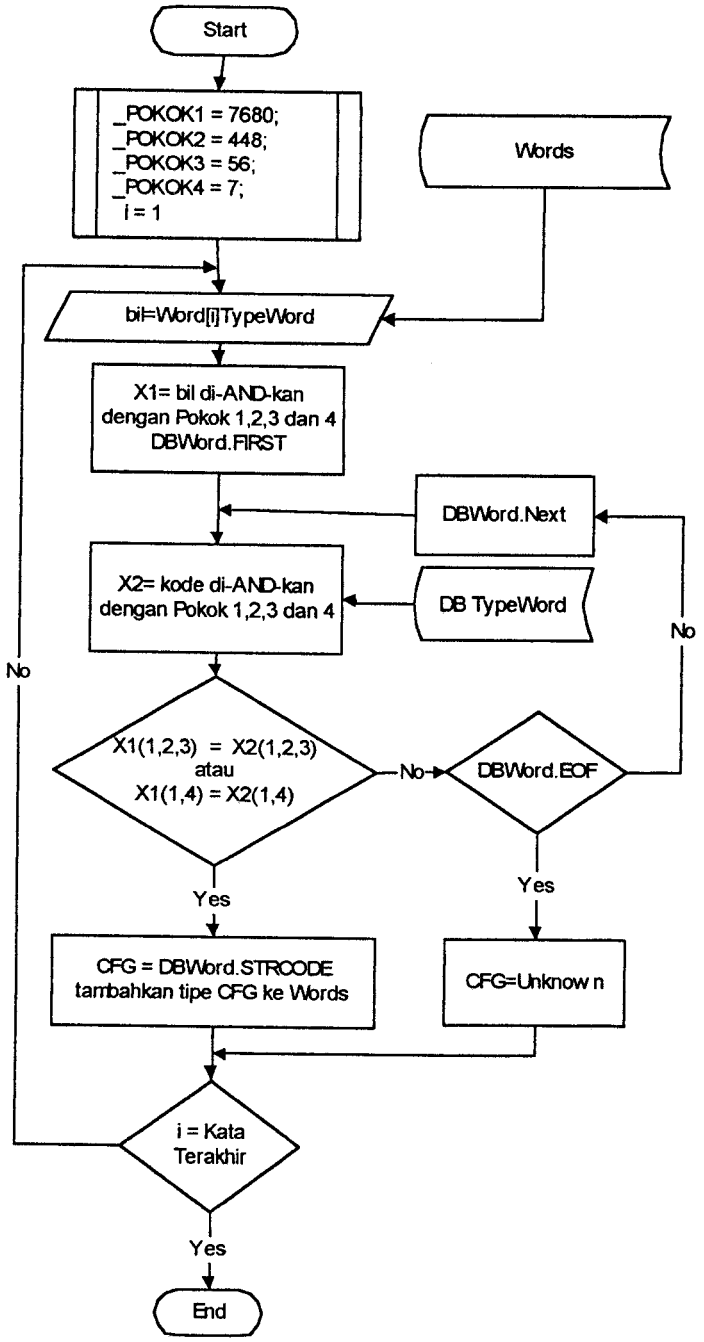


Gambar V.12 Proses 3.1 SelfConvert

Tipe kata ini masih berupa kode integer, sehingga belum bisa dikenali jika dilakukan proses parsing yang menggunakan kode CFG. Untuk itu dilakukan proses pengenalan tipe yang berkode string dari kode integer. Proses ini dinamakan dengan prosedur *SearchNameType* (integer), yang

akan mencari kode pstring dari database (DBWord.DBF) yang berisi kode integer, kode string dan kode biner (tidak digunakan dalam perangkat lunak).

Kode string ini kemudian disimpan dalam bentuk array karena bisa jadi satu kode integer memiliki lebih dari satu interpretasi kode string. Misalnya kata *given* yang memiliki kode integer 715, adalah termasuk *Direct Transitive Verb* dan *Third Verb*.

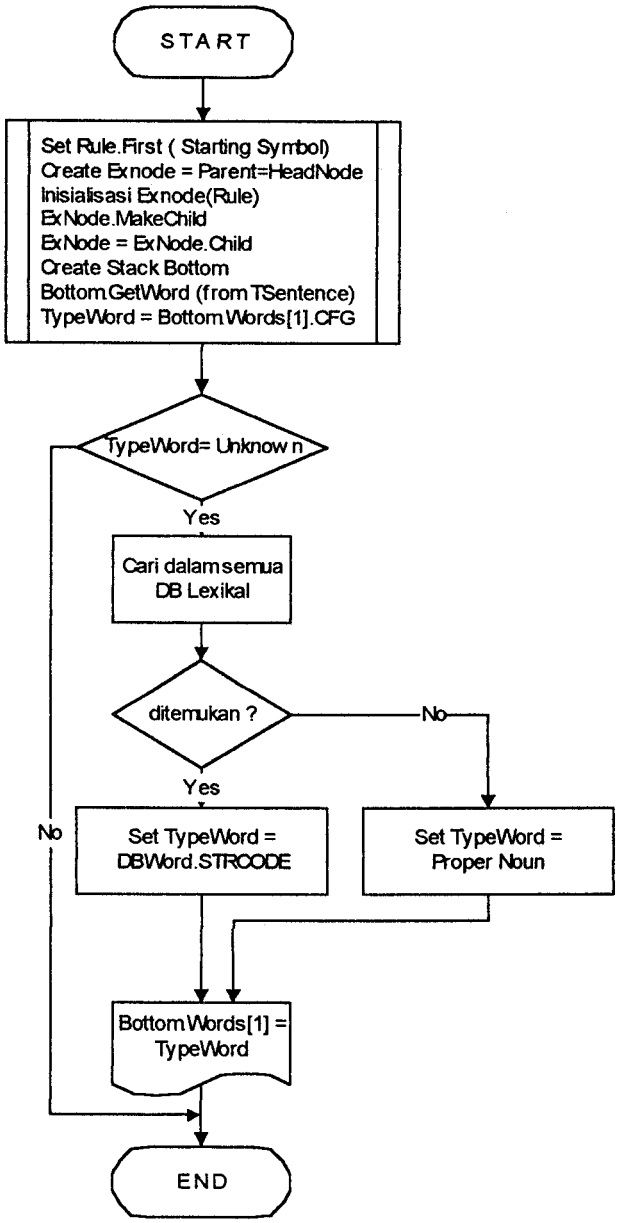


Gambar V.13 Proses 3.2 Searching CFG

Proses ini dilakukan dengan membandingkan kode integer dari data yang ada di obyek Kalimat dengan kode kata yang tersimpan dalam database yang berisi nama-nama dan kode jenis kata (DBWord). Kode ini terbagi menjadi dua kelompok yang bisa dikombinasikan pada sebuah kata. Misalnya jenis kata benda menurut jumlah (plural dan singular) dan jenis yang lain (Abstract, Proper Noun, Gender Noun dan sebagainya).

Kelompok pertama disimpan dalam kelompok biner I,II, dan III. Sedangkan kelompok kedua adalah biner I dan IV. Setelah melewati proses ini, akan terbentuk obyek kalimat yang sudah terdifinisi jenis-jenis kata yang berkode integer maupun string CFG, baru kemudian bisa dilakukan proses parsing Left Corner.

Proses ini diawali dengan inisialisasi proses seperti digambarkan dalam Flowchart di bawah ini.

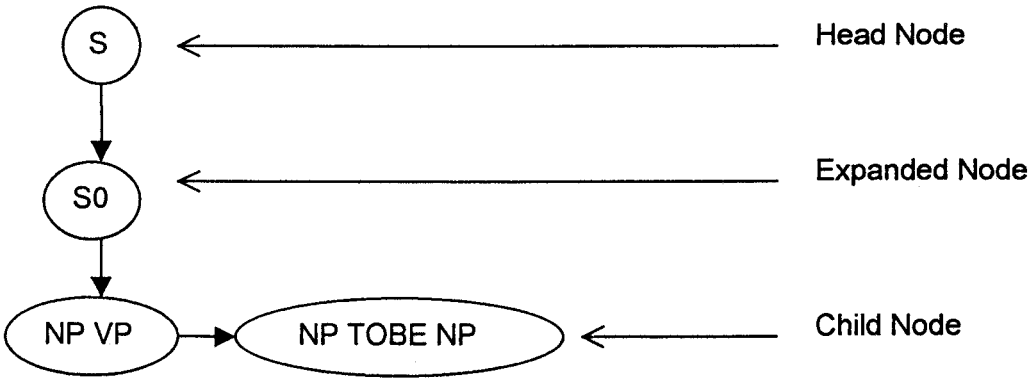


Gambar V.14 Proses 4.1 Inisialisasi Parsing

Langkah pertama yang dilakukan adalah menempatkan data kata ke dalam stack Bottom Up, membuat (*creating*) Node yang akan digunakan untuk proses *production rule*, mengeset Node dengan aturan CFG dan

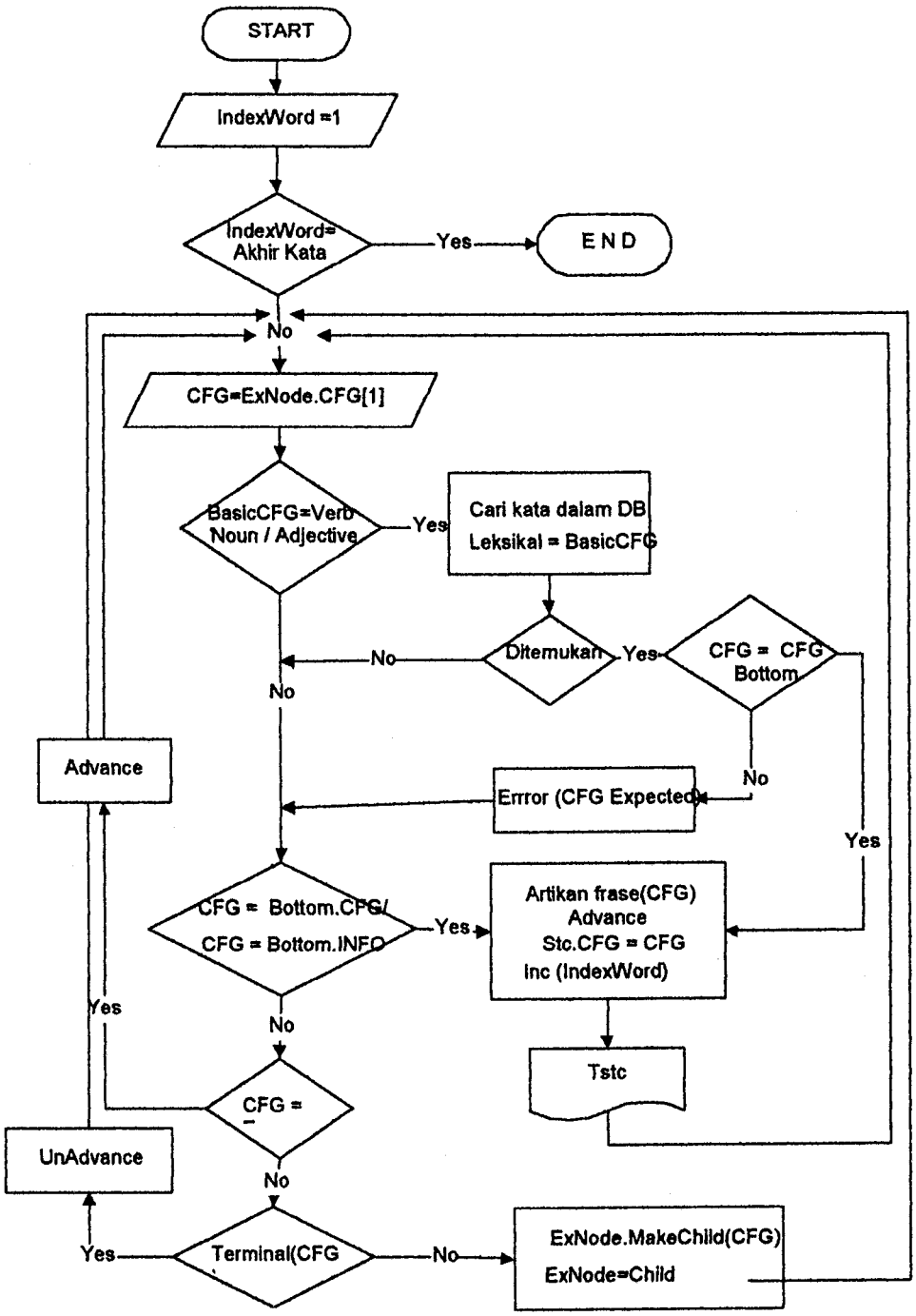
menunjuk *Starting Symbol*, membuat production pertama kali (*MakeChild*) dan langsung menuju *terminal* pertama.

Node dalam proses algoritma *Left Corner Parsing* ini dibagi menjadi tiga yaitu *ExNode, HeadNode dan Child*. HeadNode akan menunjuk Node yang pertama dibuat (pada proses inisialisasi), ExNode (*Expanded Node*) akan mewakili Node yang sekarang aktif, dan Child akan menunjuk kepada Node pertama yang diproduksi oleh *Nonterminal* yang ada di ExNode. Jadi dapat membentuk *Tree Of Node* yang dapat digambarkan seperti diagram di bawah ini, yang pembentukannya didasarkan pada aturan yang ada di CFG dan grammar yang digunakan.



Gambar V.15 Bentuk Tree of Node

Proses Left Corner pertama akan mengambil kode CFG dari Exnode. CFG ini dicek, apakah termasuk kepada tiga macam jenis kata : *Verb, Noun* dan *Adjective*. Jika ya, maka kata tersebut dicari dalam leksikal yang bersangkutan. Jika ditemukan dan cocok, maka akan dilakukan proses Advance yang akan mencari kata selanjutnya.



Gambar V.16 Proses 4.2 Left Corner Parsing Algorithm

V.6. Bahasa Pemrograman

Berikut ini adalah penjelasan mengenai bahasa pemrograman yang dipakai untuk pembuatan PL disertai faktor kelebihan dan faktor kekurangannya.

V.6.1. Jenis Bahasa Pemrograman

Bahasa pemrograman yang dipakai untuk menjalankan algoritma Left Corner Parsing dalam tugas akhir ini adalah Borland Delphi 3.0. Borland Delphi adalah bahasa pemrograman visual yang sudah cukup memiliki komponen-komponen yang diperlukan.

V.6.2. Kelebihan Jenis Bahasa Pemrograman

Adapun kelebihan - kelebihan yang dipunyai oleh Borland Delphi 3.0 antara lain :

- Menyediakan fasilitas untuk pengelolaan basis data dengan *data control* dan *data access*. Data control berarti user dapat mengelola basis data dengan bantuan kontrol - kontrol atau komponen yang disediakan oleh Borland Delphi secara visual. Data access berarti user dapat mengelola basis data dengan bantuan pemrograman basis data yang disediakan juga oleh Borland Delphi.
- Sudah mengenal sistem orientasi objek, sehingga mendukung sistem Object Oriented Design (OOD) yang telah dirancang, Dimana, dapat membuat class-class obyek sehingga dapat menghemat memory

karena setiap kali tidak diperlukan, sebuah obyek dapat di-destroy dari memory.

- Pemakaian sintaks yang mudah dimengerti, sehingga mudah dalam membaca maupun mempelajarinya.

V.6.3. Kekurangan Bahasa Pemrograman

Adapun kekurangan - kekurangan yang dipunyai oleh Delphi 3.0 adalah kurang bisa fleksibel untuk dikembangkan ke basis pemrograman lain, misalnya dalam bentuk basis web yang saat ini banyak dikembangkan.

V.7. Implementasi Prosedur Utama

Berikut ini adalah prosedur - prosedur utama yang dipakai untuk proses scanning dan parsing dalam PL ini :

V.7.1. Prosedur Scan

Berikut ini diberikan salah satu contoh prosedur scan, yang dapat dijadikan representasi prosedure scan lainnya. Prosedur ini digunakan untuk mengenali jenis-jenis token dari karakter yang diseleksinya.

```
TScanner.Scan000(Ch : Char; par : byte);
{ Scan pertama kali pada awal paragraf }

 Case Ch Of
     UPPERCASE : SetToken(TK_AWAL); Action = 11;
     Angka     : SetToken(TK_ANGKA); Action = 12;
     '$'  :  SetToken(TK_CURRE);  Action  =  13;  FinishCurr  =
 False;
     TAB, SPASI      : SetToken(GANTI_TK); Action = 1;
```

```
'-'              : SetToken(TK_ANGKA); Action = 15;
PGanda,PTunggal: SetToken(GANTI_TK); Action =16;
                 Status_PETIK = PETIK
'('      : begin  SetToken(GANTI_TK); Action =16;
                 Status_PETIK = KR_BUKA
Enter, GtBaris :
  Begin
     If Ch ← Tanda_GtBaris Then
        inc(baris);
        kolom = 0;
     SetToken(Ch, GANTI_TK ); Act = 1;
     Status_TTK = 0;
  End
Lowercase :
  Begin
     if par in [TTK_VAR, TTK_DUA] Then
         begin SetToken(Ch, TK_BIASA ); Act = 114; end;
     if par in [TANPA_TTK, TTK_KATA] Then
         SimpanError(baris, kolom, 1, Ch );
   End
Else  SimpanError(baris,kolom,1, Ch);

{ cek karakter selanjutnya }
kolom = kolom + 1 ;
DoAction;
```


Keterangan :

- Action digunakan untuk menuju prosedur scan selanjutnya
- Prosedure SetToken akan mengeset jenis token yang sekarang aktif menjadi seperti dalam kurung
- Status-status seperti FinishCurr, Status_PETIK, Status_TTK adalah untuk menandai awal dan akhir dari sebuah tanda baca, akhir kalimat atau token. FinishCurr yang bernilai true digunakan jika menemukan karakter angka, maka tidak dijadikan token angka, namun Currency. Status petik akan digunakan jika ditemukan kata atau frase yang berada dalam kurung, petik tunggal atau ganda. Dan status titik akan digunakan jika terjadi peralihan paragraf atau paragraf baru.

Jika ternyata karakter yang diseleksi tidak terdapat dalam daftar, maka dilakukan prosedur Simpan Error dengan parameter baris, kolom, nomor error yang akan menunjuk index dalam tabel error, dan karakter yang error tersebut.

Tahap terakhir adalah melakukan prosedure DoAction yang akan memilih prosedur Scan Selanjutnya.

V.7.2. Prosedure Do_Action

Prosedur ini digunakan untuk menunjuk prosedur Scan selanjutnya :

```
TScanner.DoAction;

  If baris >= MaxLines Then Selesai = True;
  if kolom > 1 Then CBef = Ts[baris,kolom-1];
  If Not Selesai Then
  Case Action Of
    1 : Scan000(Ts[baris,kolom],STATUS_TTK);//status titik
    11 : Scan011(Ts[baris,kolom]); // Scan huruf besar pertama
    12 : Scan012(Ts[baris,kolom]); // Scan Angka
    13 : Scan013(Ts[baris,kolom]); // Tanda tanda dollar
    15 : Scan015(Ts[baris,kolom]); // Tanda negatif

   …. Dan seterusnya sampai dengan …

 11271  :  Scan11271  (Ts[baris,kolom]);  //    Scan  setelah
possessive
End;
```

Keterangan :

- Action digunakan untuk menunjuk prosedur scan
- Variabel selesai menandai akhir input string, sehingga tidak perlu dilakukan scan lagi
- Ts adalah variabel penyimpan input yang bertipe TStrings (dalam Delphi) yang dapat menyimpan rangkaian string. Baris dan kolom adalah variabel yang menunjuk sebuah karakter di dalamnya.

- CBef (Character Before) digunakan untuk menandai karakter sebelumnya. Jika terjadi kesalahan input, maka karakter ini disimpan pula dalam data error, sehingga dapat diketahui posisi karakter yang salah dan karakter yang mendahuluinya.

V.7.3 Prrocedure Searching_CFG

Prosedure ini digunakan untuk mencari jneis-jenis CFG bertipe string dari sebuah nilai integer.

```
function SearchNameType(bil : integer;
                        var Ncfg : byte ):arrTypeCFG;
CONST _POKOK1 = 7680;  // 1111 000 000 000
      _POKOK2 = 448;   // 0000 111 000 000
      _POKOK3 = 56;    // 0000 000 111 000
      _POKOK4 = 7;     // 0000 000 000 111


  found = false;
  n = nCfg;  { jumlah CFG yang sudah ada }

  { untuk contoh bil biner 0011 100 011 001 }

  BilFirst = bil and _POKOK1;  // diambil 0011 000 000 000
  BilSecond = bil and _POKOK2; // diambil 0000 100 000 000
  BilThird = bil and _POKOK3;  // diambil 0000 000 011 000
  BilFourth = bil and _POKOK4; // diambil 0000 000 000 001

  { ambil kode dari basis data jenis kata }

  while (not dbWord.EOF) do
  begin
    KODEWORD = dbWord.fieldValues['INT_CODE'];
    CdFirst = KODEWORD and _POKOK1;
    CdSecond = KODEWORD and _POKOK2;
    CdThird = KODEWORD and _POKOK3;
    CdFourth = KODEWORD and _POKOK4;

{bandingkan biner pertama (I,II dan III ) dan kedua ( I dan
IV )}

if ((CdSecond = BilSecond) and (CdFirst = BilFirst) and
(CdThird = BilThird)) or ((CdPokok = BilPokok) and (CdFourth
= BilFourth) and (BilFourth <> 0)) then
      begin
```

V.7.4. Prosedur algoritma *Left Corner parsing*

Prosedure ini merupakan prosedure parsing utama untuk memeriksa susunan frase dan memilih jenis kata yang tersusun dalam TSentence.

```
procedure LeftCorner(Var Stc : TSentence);

If Terus then
   begin
   Set terus = false
   Set nextNode = false

   {ambil CFG berdasarkan Index CFG dari ExNode yang aktif }
   {misalnya ExNode tersebut berisi 'NP VP' dan Index = 2 }
   {maka CFG = VP }

   CFG = ExNode.Arrcfg[Exnode.IndexCFG]

   {Index kata dalam TStc juga dimasukkan ke ExNode agar }
   {ketika terjadi kesalahan, langsung ditunjuk index kata}
   {yang sudah benar sebelumnya dari TStc }

   ExNode.arrIndex[ExNode.IndexCFG] = IndexWord

   {CFG diseleksi berdasarkan dua tingkat, yang harus}
   {benar kedua-duanya; disimpan dalam variabel IsDiff}

   if not isDiff Then CFGTambah =
   ExNode.ArrTambah[Exnode.IndexCFG]

   Basic_CFG = Copy(CFG,Length(CFG)-3,4)
   if ((Basic_CFG = 'VERB'/'NOUN'/'ADJE')) then
         SwitchDBDefault(DB Leksikal yang sesuai dgn Basic_CFG)
   else
         TypeError = 'Non Expected CFG'
         if kata di Bottom Up ada di DBDefault then
         begin
         Set found = true

        {looping ini digunakan jika kata memiliki lebih dari }
        { satu morfologi dalam sebuah database leksikal }

        While found Do
        Begin

         { Set CFG di Bottom Up dengan CFG baru dari DBDefault}

         TmpType = DbDefault.FieldValues['TYPEWORD']
         Bottom.Item[Bottom.Top].ArrCfg :=
         SearchNameType(TmpType,n)
         If CFG dan CFGTambah cocok dengan di Bottom Up Then
          Begin
            { CFG di TStc di update, termasuk artinya }
```

```
            Stc.Words[IndexWord].Cfg = CFG
            Stc.Words[IndexWord].realMean = DBDefault['MEANS']

            {ExNode kemudian juga mengenerate arti frase
            {ditransfer ke Stc membentuk arti kalimat, lalu }
            {menjalankan prosedure Advance yang akan menuju }
            {kata selanjutnya, yang di dalamnya juga mengeset }
            {Terus = True sehingga proses parsing dilanjutkan }

            ExNode.GetMeans;
            Stc.means := ExNode.means;
            Advance
            NextNode = true

          end  { if cocok }
          else { jika ditemukan tapi tidak cocok }
            TypeError = ' UnExpected CFG '
        end
      end
    end


{ jika Basic CFG tidak termasuk ketiga leksikal di atas }

if Not NextNode Then
begin
   if CFG = kata, atau CFG/CFGTambah = CFG di Bottom Up then

   { lakukan seperti prosedure Advance Di Atas }

   else if (CFG = 'E' {Empty}) Then Advance

   { Jika CFG menunjuk Empty, maka langsung Advance tanpa }
   { mengeset CFG dari kata di TStc }

   else if isTerminal(CFG) then UnAdvance

   {Jika CFG menunjuk Terminal dan tidak cocok, maka }
   {dilakukan proses UnAdvance}
   {yang akan menuju parent dari ExNode, dan mengembalikan }
   {indexword ke posisi sebelum kesalahan, disimpan di
   ArrIndex }

   else
   begin

      {Jika CFG tsb Non Terminal, maka bisa diextrac lagi}
      {dengan prosedure MakeChild berdasarkan CFG sekarang }

      if CFG dan CFGTambah tidak sama then isDiff = True
      ExNode.MakeChild

      {kemudian, terus di set ke true, sehingga bisa }
      {proses parsing lagi, dan ExNode di set ke Child }
```

```
            Set terus = true
            ExNode = ExNode.Child
          end
        end

    {Error dalam proses parsing belum tentu menghentikan proses}
    {cukup disimpan dalam array, sehingga ketika proses parsing}
    {berhenti, bisa dilihat kesalahan apa yang pernah terjadi}

    if (Error.Top > 0) Then Error.Item[Error.Top] = TypeError

  end { Terus }
```

Keterangan :

- Variabel terus digunakan untuk memberikan status apakah proses parsing bisa dilanjutkan atau tidak. Sedangkan Success digunakan untuk memberikan status keberhasilan proses parsing untuk sebuah kalimat
- ExNode adalah variabel bertipe TNode yang sedang aktif
- NextNode untuk menandai apakah CFG yang aktif sekarang termasuk di dalam ketiga leksikal *verb, noun* dan *adjective,* dan ditemukan dalam database leksikalnya. Jika tidak maka dilakukan proses-proses yang lain.
- Prosedur atau fungsi Advance, Unadvance, MakeChild, dan SearchNameType yang berada dalam prosedur parsing Left Corner sengaja tidak ditampakkan.

BAB VI

UJI COBA DAN EVALUASI PERANGKAT LUNAK

Pada bab ini akan dibahas tentang uji coba dan evaluasi terhadap performansi PL yang dibangun berdasarkan rancangan yang telah dibuat. Pembahasan dalam bab ini meliputi bentuk tata muka, data statis yang digunakan, proses scanning dan parsing serta data output yang dihasilkan dengan beberapa contoh studi kasus.

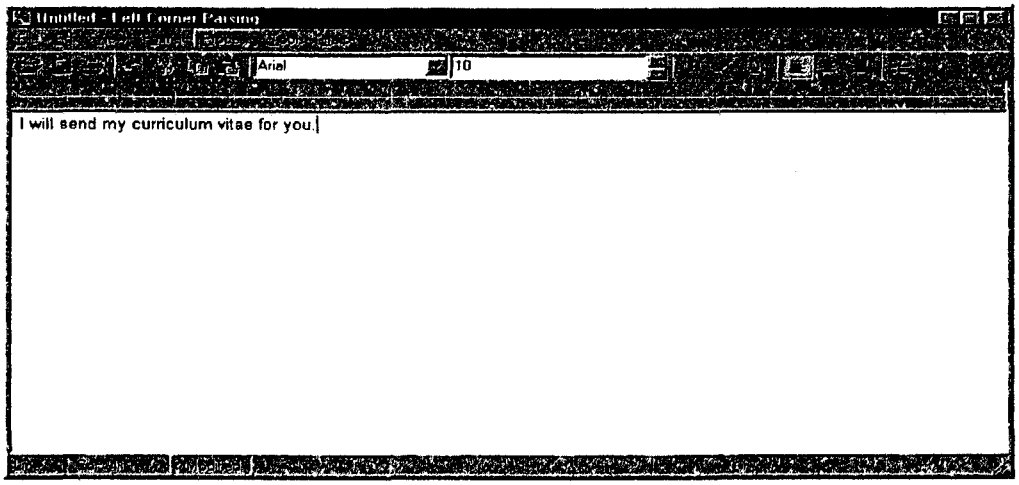
Metodologi pengujian yang dilakukan adalah dengan cara mengambil data-data kalimat sampel dari buku-buku yang memiliki kalimat Bahasa Inggris dengan pola tata bahasa yang baku. Kata-kata yang menyusun contoh kalimat ini harus terlebih dahulu sudah ada (dimasukkan) ke dalam basis data leksikal, baru dapat diproses oleh parser.

VI.1. Tata Muka (User Interface)

Tata muka perangkat lunak ini dipisahkan menjadi dua program utama. Program pertama adalah editor untuk menempatkan data input dan sekaligus menjalankan proses parsing, sedangkan yang kedua sebagai *Data Entry* untuk basis data pendukung PL ini.

Editor yang dipakai untuk menempatkan data input adalah komponen yang telah dimiliki oleh Borland Delphi 3.0 yaitu TRichEdit yang telah mendukung beberapa operasi *word processor* seperti cut, copy, paste, font setting, dan sebagainya. Sehingga untuk mendukung perangkat lunak ini cukup menambahkan beberapa menu seperti :

- View – Token List : melihat data token dan tipe kata
- View – Result : melihat hasil arti kata sintaktik dan semantik
- View – Check Type : memeriksa jenis kata dari suatu integer
- Process – Compile : memproses input string membentuk token
- Process – Left Corner Parsing : memproses input sampai analisis secara sintaktik maupun semantik
- Tool – Database : memanggil program *database editing*
- Tool – Options : memilih proses parsing yang *viewable* atau tidak



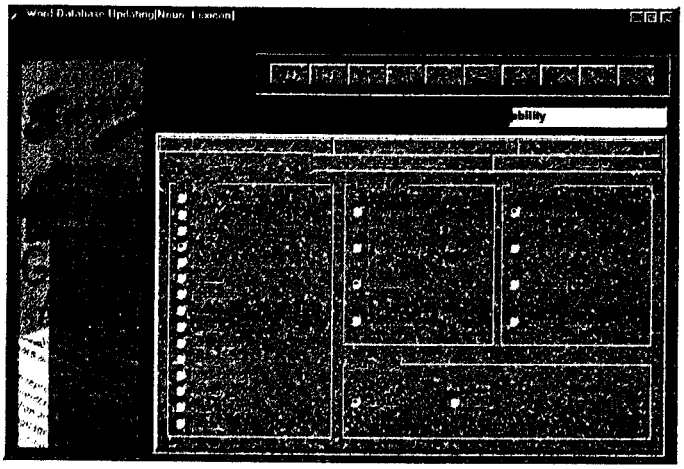
Gambar VI.1 Bentuk tampilan program editor

Sedangkan untuk program *Data Entry* dibuat dengan beberapa feature seperti :

- Mengubah rule CFG, CFG tambahan dan rumus semantik
- Menambah, mengurangi atau mengubah data leksikal
- Menambah, mengurangi atau mengubah data-data statis seperti jenis kata, kalimat dan possessive

- Memeriksa kombinasi biner berdasarkan jenis-jenis kata yang telah didifinisikan

Menu-menu ditampilkan dalam bentuk pop-up menu yang bisa muncul ketika kita menekan tombol kiri mouse pada titik merah pada form entry seperti di bawah ini.



Gambar VI.2 Bentuk tampilan program *updating database*

VI.2. Data Statis

Data statis yang dimaksud disini adalah data-data yang digunakan untuk mengoperasikan program perangkat lunak utama (editor). Data-data ini dapat ditambah, dikurangi atau diubah sesuai dengan kebutuhan sistem.

Data-data tersebut meliputi antara lain : leksikal pokok (reserved word), leksikal kata kerja, leksikal kata benda, leksikal kata sifat, aturan CFG, jenis-jenis kalimat, jenis-jenis kata dan derivasinya, jenis kata bertanda petik tunggal (possessive word), dan jenis-jenis kata singkatan

Perangkat lunak ini sangat tergantung pada validitas data yang dimasukkan dalam program Data Entry ini. Oleh karena itu pengguna harus betul-betul mengerti bagaimana mengoperasikannya dan sekaligus memahami tipe-tipe sebuah kata. Sebagai contoh, kata 'taken' adalah *Direct Transitive Verb* dan *Third Form Verb*.

VI.3. Proses Scanning dan Parsing

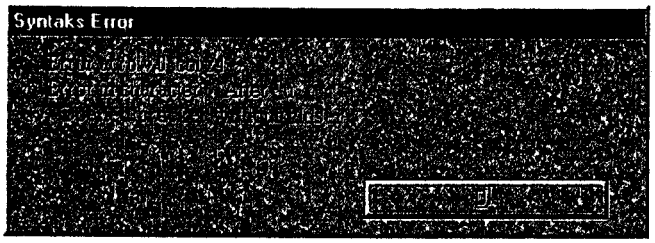
Proses scanning ataupun parsing selalu diawali dengan proses inisialisasi yang digunakan untuk mempersiapkan data-data statis yang dibutuhkan sistem. Proses ini meliputi : *creating object*, *activating lexicon databases, loading database error, loading CFG*, dan *setting* status pada prosedur *scanning* and *parsing* dalam posisi awal.

Setelah input teks dimasukkan dalam text editor, proses pertama yang dilakukan adalah inisialisasi variabel-variabel yang akan melakukan beberapa proses seperti : create object pada class scanner dan parser dan mengeset Rule CFG ke parser.

Proses scanning akan menganalisis string input yang dimasukkan ke Editor, membentuk rangkaian token dan membentuknya menjadi sebuah struktur obyek paragraf, yang bisa berisi satu atau lebih kalimat di dalamnya. Proses scanning ini akan berhenti dan menandakan error jika ditemukan kesalahan pada input tersebut.

Sebagai contoh, jika dimasukkan kalimat tanpa titik, huruf awal kecil, penempatan tanda baca petik tunggal yang tidak tepat, kurung buka dan tutup yang tidak seimbang akan menyebabkan proses scanning berhenti, sehingga tidak bisa dilanjutkan pada proses parsing. Sebagai

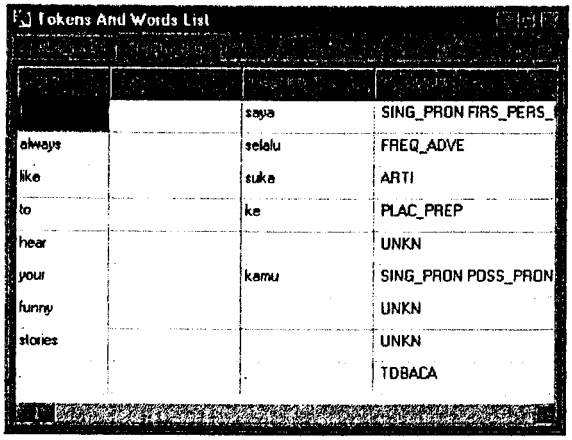
contoh kalimat : 'I am very happy to see (you.', akan menyebabkan program mengeluarkan pesan :



Gambar VI.3 Error Message Scanning

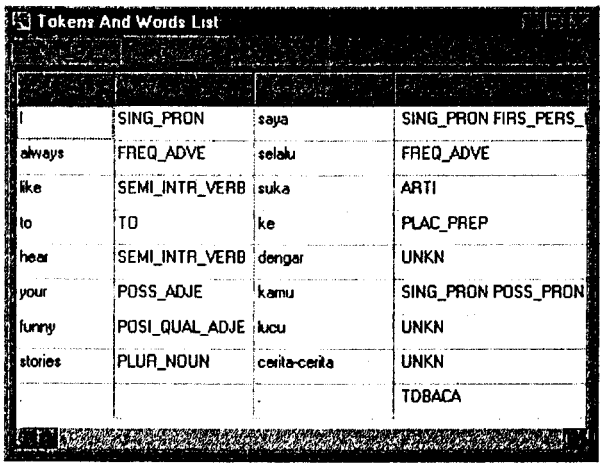
Jika proses scanning ini berhasil, akan dihasilkan rangkaian kata yang sudah tergabung dalam obyek kalimat. Sudah terdifinisi kata-kata yang termasuk dalam reserved word (leksikal default) dan yang tidak (*unknown type*). Hasil dari obyek kalimat ini bisa dilihat dalam menu view – list token. Dalam form tersebut bisa ditampakkan kata-kata yang sudah terdaftar dalam reserved word dan yang belum, juga daftar dan jenis-jenis token yang dibentuk.

Sebagai contoh kalimat 'I always like to hear your funny stories', setelah proses scanning akan muncul tabel token dan jenis kata seperti di bawah ini.



Gambar VI.4 Data jenis kata setelah proses scanning

Jika sudah dilakukan proses parsing, jenis-jenis kata yang lain (yang bertipe 'UNKN' maupun yang sudah terdifinisi namun bisa jadi salah) akan muncul seperti tabel di bawah ini.



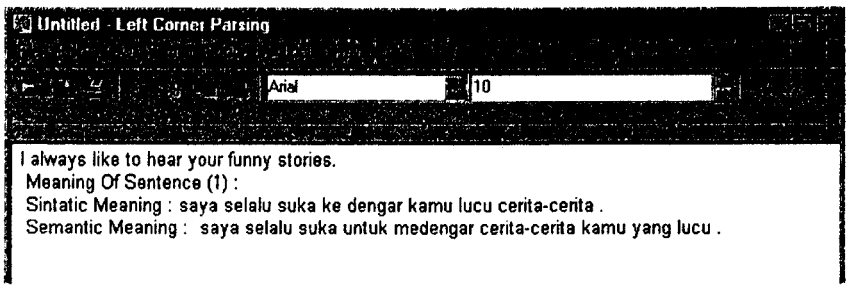
Gambar VI.5 Data jenis kata setelah proses parsing

Walau terkadang bertipe sama, seperti kata 'always' di atas, antara *type* dan *prediction type* di atas berbeda asal. *Prediction Type* adalah tipe-tipe kata yang telah dihasilkan dari proses scanning, sedangkan *Type* adalah hasil dari proses *Left Corner Parsing.* Arti

(*meaning*) dari kata tersebut adalah didasarkan pada analisis sintaksis, sehingga belum bisa disebut arti kata secara semantik.

Bersama dengan proses parsing Left Corner yang digunakan, dalam perangkat lunak ini juga melakukan penterjemahan secara semantik walau masih pada tahap sederhana. Proses ini dilakukan secara rekursif dengan sistem penterjemahan per frase yang rumusnya dibentuk dan ditempatkan menurut *production rule* dalam CFG.

Setiap kali melakukan parsing per frase, proses penterjemahan ini dilakukan. Sehingga, dalam sebuah kalimat, arti yang ditimbulkan adalah berdasarkan arti per frase, bukan per kata. Sebagai contoh, kalimat *I always like to hear your funny stories* akan memiliki arti 'saya selalu suka untuk medengar cerita-cerita kamu yang lucu'. Hasil ini bisa diketahui dalam form Editor yang keluar jika ditekan menu View – Result.



Gambar VI.6 Gambar Editor hasil terjemahan kalimat
dengan analisis secara sintaktik dan semantik

Dalam contoh di atas, kalimat tersebut terbagi menjadi beberapa frase menurut production rule yang dihasilkan kalimat tersebut. Proses penterjemahannya dapat dilihat dalam tabel di bawah ini :

Production Rule	Rumus	Sentence/Phrase	Meaning
S → NP_1 VP_1	1+2	I always like to hear your funny stories	Saya selalu suka untuk medengar cerita-cerita kamu yang lucu
NP_1 → PRON	1	I	Saya
VP_1 → ADVE TV NP_2	1+2+3	always like to hear your funny stories	selalu suka untuk medengar cerita-cerita kamu yang lucu
NP_2 → VERBAL	1	to hear your funny stories	untuk medengar cerita-cerita kamu yang lucu
VERBAL → TO VP_2	untuk+me-2	to hear your funny stories	ke medengar cerita-cerita kamu yang lucu
VP_2 → TV NP_3	1+2	hear your funny stories	dengar cerita-cerita kamu yang lucu
NP_3 → POSS NT	2+1	your funny stories	cerita-cerita kamu yang lucu
NT → ADJE N	2+yang+1	funny stories	cerita-cerita yang lucu

Tabel VI.1 Proses rekursif pada algoritma Left Corner untuk menterjemahkan frase

VI.4. Analisis dan Evaluasi

Sesuai dengan jenis-jenis kalimat yang sudah didefinisikan dalam bab Perancangan dan Implementasi PL, maka dalam bab ini akan dilakukan beberapa contoh kalimat[1] tersebut dan analisis hasil proses scanning dan parsing oleh PL. Jenis-jenis kalimat tersebut antara lain :

1. Statement (kalimat berita)
   - There are two boys in the class
   - A boy was killed by a tiger.
2. Question (kalimat tanya)
   - How old are you?
   - Where does he live?
3. Desires (kalimat harapan atau perintah)
   - May you be happy
   - Lend me some money
   - Stand Up!
4. Exclamation (kalimat seru)
   - How stupid!
   - What a pity!
   - Pardon!

Contoh-contoh kalimat yang diberikan di atas akan menghasilkan output sebagai berikut :

[1] Referensi nomor 1 hal. 3-4

Contoh kalimat #1 :

Kalimat : 'There are two boys in the class.'

Tabel kata setelah dilakukan proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
There	unknown	single noun	-
are	first tobe	first tobe	adalah
two	unknown	quantitive adjective	-
boys	unknown	plural noun	-
in	preposition	di dalam	di dalam
the	article	itu	itu
class	unknown	-	-

Tabel VI.2 Hasil output kalimat 'There are two boys in the class.'

Arti kalimat :

Sintaktik : -

Semantik : -

Pada kalimat ini struktur pola kalimat yang didahului dengan auxiliary belum teratasi dengan aturan CFG yang ada, sehingga tidak bisa mengeluarkan output jenis kalimat dan arti selengkapnya setelah proses parsing. hal ini karena proses parsing berhenti pada kata kedua, dimana CFG belum bisa mengenali kalimat yang terdapat frase kata benda ('two boys') yang diikuti oleh frase kata katerangan ('in the class').

Contoh kalimat #2 :

Kalimat : 'A boy was killed by a tiger'

Tabel kata setelah dilakukan proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
A	article	article	sebuah
boy	unknown	single noun	laki-laki
was	second tobe	second tobe	adalah
killed	unknown	intransitive verb	dibunuh
by	preposition	preposition	oleh
a	article	article	sebuah
tiger	unknown	single noun	macan

Tabel VI.3 Hasil output contoh kalimat 'A boy was killed by a tiger.'

Arti kalimat :

Sintaktik : Sebuah laki-laki adalah dibunuh oleh sebuah macan

Semantik : Sebuah laki-laki adalah medibunuh oleh sebuah macan

Pada contoh kalimat di atas aturan CFG yang ada sudah mampu mengenali pola kalimatnya sehingga dapat mendifinisikan semua jenis katanya. sedangkan pada arti semantik mengalami kesalahan output karena pola penterjemahan frase yang masih kurang sempurna pada jenis kalimat tersebut.

Contoh kalimat #3 :

Kalimat : 'How old are you?'

Tabel kata setelah proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
How	interogative	'How'	Bagaimana
old	unknown	positive quality adjective	tua
are	first tobe	first tobe	adalah
you	single pronoun & second personal pronoun	single pronoun	kamu

Tabel VI.4 Hasil output kalimat 'How old are you?'

Arti kalimat :

Sintaktik : Bagaimana tua adalah kamu ?

Semantik : Betapa tua kamu ? (sudah benar)

Contoh kalimat #4 :

Kalimat : 'Where does he live?'

Tabel kata setelah proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
Where	conjunction	conjunction	dimana
does	first tobe	first tobe	-
he	single pronoun, third pers. pron	single pronoun	dia
live	unknown	intransitive verb	tinggal

Tabel VI.5 Hasil output kalimat 'Where does he live?'

Arti kalimat :

Sintaktik : Dimana – dia tinggal ?

Semantik : Dimana dia tinggal ?

Contoh kalimat #5 :

Kalimat : 'May you be happy?'

Khusus untuk kalimat ini parser belum bisa mengidentifikasi pola kalimatnya menurut aturan dalam CFG yang diberikan, yaitu jenis kalimat yang diawali auxiliary dan bukan kalimat pertanyaan. Sehingga belum bisa mengeluarkan hasil yang diharapkan.

Contoh kalimat #6 :

Kalimat : 'Lend me some money'

Tabel kata setelah proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
Lend	unknown	indirect transitive verb	pinjam
me	Single Noun, First Personal Pronoun	single noun	saya
some	plural noun, indefinitive quantity adjective	indefinitive quantity adjective	beberapa
money	unknown	plural noun	uang

Tabel VI.6 Hasil output kalimat 'Lend me some money !'

Arti kalimat :

Sintaktik : Pinjam saya beberapa uang !

Semantik : Pinjami saya beberapa uang !

Pada contoh kalimat din atas, parser berhasil mengenali pola kalimatnya sehingga dapat mengeluarkan output sebagaimana mestinya.

Contoh kalimat #7 :

Kalimat : 'Stand Up!'

Tabel kata setelah proses scanning dan parsing :

Kata	Jenis kata setelah kompilasi	Jenis kata setelah parsing	Arti
Stand	unknown	intransitive verb	berdiri
up	preposition	preposition	atas

Tabel VI.7 Hasil output kalimat 'Stand up!'

Arti kalimat :

Sintaktik : Berdiri atas !

Semantik : Berdiri atas !

Walaupun contoh kalimat di atas berhasil dianalisis oleh scanner dan parser, namun hal ini sebenarnya tidak sesuai dengan hasil output yang diharapkan. Artinya, penelusuran kalimat tersebut kebetulan benar karena aturan CFG yang dibuat sebenarnya belum bisa mengenali kalimat majemuk, dimana kombinasi dua atau lebih kata memiliki satu arti.

Pada contoh kalimat jenis *exclamation* di atas, hanya kata 'pardon' yang bisa dikenali oleh CFG karena bukan merupakan kalimat majemuk, sehingga masih mengikuti pola tata bahasa yang semestinya. Sedangkan 'What a pity!' dan 'How stupid!' tidak mengikuti pola yang semestinya, sehingga (dalam tugas akhir ini) tidak dikenali oleh aturan CFG.

BAB VII

PENUTUP

Setelah dilakukan uji coba dan evaluasi terhadap perangkat lunak penggunaan algoritma *Left Corner Parsing*, pada bab ini dibahas tentang kesimpulan yang dapat diambil dari perangkat lunak tersebut dan saran - saran yang berguna untuk pengembangan sistem lebih luas lagi.

VII.1. Kesimpulan

Dari perancangan dan implementasi serta evaluasi performansi perangkat lunak penggunaan algoritma Left Corner Parsing untuk memeriksa kebenaran tata bahasa dan makna dalam Bahasa Inggris, dapat diambil beberapa kesimpulan yaitu :

1. Algoritma *Left Corner (LC) parser* akan dapat membedakan arti dari dua atau lebih kata yang morfologinya sama, namun memiliki maksud (arti) lain dalam sebuah kalimat Bahasa Inggris, dengan syarat tidak di awal kalimat.
2. Dapat membangkitkan makna sebuah kalimat dengan pola rekursif menurut susunan frasenya, sehingga arti yang dihasilkannya lebih mendekati bahasa natural manusia.
3. Keberhasilan *LC parser* sangat tergantung oleh oleh aturan CFG yang dibuat dan tingkat pemahaman tata bahasa yang dipakai oleh pengguna
4. Kalimat yang dianalisis oleh *LC parser* tidak termasuk kata kiasan atau majemuk yang analisisnya tidak empiris, melainkan menggunakan intuisi
5. Penerapan *LC parsing* dapat memiliki lebih dari satu rule CFG

6. Pendekatan analisis pada jenis kalimat akan dapat membantu kecepatan proses *LC parsing*, karena langsung bisa menunjuk rule yang sesuai di awal proses *parsing*
7. Dapat mengenali jenis kata sampai dengan 3 tingkat dan 2 kombinasi sehingga bisa mengenali jenis kata secara lebih detail

VII.2. Saran-saran

Pembuatan perangkat lunak untuk implementasi algoritma *Left Corner parsing* untuk memeriksa kebenaran tata bahasa dan makna kata ini tentunya tidak terlepas dari kesalahan dan kekuarangan. Oleh karenanya, berikut ini adalah saran-saran yang bisa digunakan untuk pengembangan tugas akhir ini lebih jauh lagi di masa mendatang sehingga dapat menjadi perangkat lunak yang lebih sempurna.

1. Perlu aturan CFG yang lebih lengkap lagi (sementara ini 2 tingkat) untuk memeriksa jenis kata yang lebih kompleks, sehingga dapat memasukkan berbagai macam kombinasi jenis kata dalam sebuah kalimat.
2. Perlu pemahaman yang lebih sempurna mengenai tata bahasa yang dipakai, sehingga aturan CFG dibentuk dapat mengatasi berbagai kemungkinan sintaks pada kalimat yang diinputkan.
3. Perlu ketelitian dalam memberikan jenis kata dalam basis data leksikal, sehingga cocok dengan CFG yang telah dibuat.
4. Analisis imbuhan, CFG yang lebih lengkap, dan pola penterjemahan frase perlu dikembangkan lebih lanjut lagi agar kemampuan PL ini bisa lebih optimal tidak hanya untuk analisa tata bahasa, namun juga analisis terjemahannya

DAFTAR PUSTAKA

[1] Milon Nandy, "A Practical English Grammar With Exercise", Penerbit Composite Study Aid Publication, Singapura, 1994

[2] M. W. Croker,"Mechanism For Sentence Processing", Thessis, Center Of Cognitive Science, University Of Edinburg, 1994

[3] Christopher F. Chabris, "Artificial Intelligence And Turbo Pascal", Penerbit Multiscience Press.Inc, Illinois

[4] Dra. B.M.G. Endang Sri Wulandari, "A Reading Program, For First Year Non-English Department University Students In Indonesia", Penerbit Kanisius, Jogyakarta, 1997

[5] Achmad Marzuq M, "Pintar Kosakata Bahasa Inggris", Penerbit Indah, Surabaya, 1995

[6] Michael A.Covington, "Natural Language Processing For Prolog Programmers", Prentice Hall, New Jersey, 1994

[7] Alfred V. Aho, "Principles Of Compiler Design", Addison-Wesley Publishing Company, 1997

[8] Perangkat Lunak "WordNet Online Lexical Database", Princeton University Cognitive Science Laboratory, alamat : www.cogsci.princeton.edu/~wn/, 1998

[9] Engish Grammar Documentation, http://www.english.uiuc.edu/cws/wworkshop/grammarmenu.htm

LAMPIRAN

Context Free Grammar :

```
S>S0
S0>CONJL S1 SCONJ
S0>CONJL S2 SCONJ
S0>CONJL S3 SCONJ
S0>CONJL S4 SCONJ
S1>NPL TOBE STOBE
S1>NPL AVVL VP
S1>NPL AVVL AUXI AVVL VP
S2>INTR TOBE NP AVL PPL
S2>AUXI NPL AVVL VP
S2>TOBE NPL AVVL VP
S2>TOBE NPL QUAL_ADJE
S2>TOBE NPL NP PPL
S2>How QUAL_ADJE TOBE NP
S2>How QUAL_ADJE TOBE NP VP
S2>Which TOBE POSS_PRON
S2>Which NP TOBE NPL VP
S2>transitive_verb INTR
S3>DONL SQ
SQ>VP
SQ>AUXI NPL VP
SQ>BE QUAL_ADJE
DONL>DO NOT
DONL>E
S4>CL , S
CL>SUBO_CONJ
CL>ADVE
CL>PPL
CL>ADJE
CL>NP PPL
SCONJ>CONJ SCONJ S
SCONJ>ADVE
SCONJ>e
CONJL>CONJ
CONJL>e
STOBE>PPL
STOBE>QUAL_ADJE TO VP PPL
STOBE>NP NPL
STOBE>QUAL_ADJE
STOBE>AVVL VP
STOBE>AVVL VP
STOBE>been VP
TOBE>first_tobe
TOBE>second_tobe
TOBE>third_tobe
AVVL>FREQ_ADVE
AVVL>DEGR_ADVE
AVVL>NEGA_ADVE
AVVL>MANN_ADVE
AVVL>e
NP>among NP NPL
NP>between NP AND NP
NP>ANP RP
NP>VERBAL
NP>FRON ANP
NP>BACK ANP
NP>subordinate_conjunction S0
VP>direct_transitive_verb NP AVL PPL
VP>indirect_transitive_verb NP NPL
AVL PPL
VP>intransitive_verb AVL PPL
VP>semi_intransitive_verb NPL AVL PPL
VP>adjective_verb QUAL_ADJE PPL
VP>link_verb TO VP
VP>VPASIF
VPASIF>direct_transitive_verb AVL PPL
VPASIF>indirect_transitive_verb NPL
AVL PPL
AUXI>first_auxiliary
AUXI>second_auxiliary
AUXI>third_auxiliary
INTR>INTR_SING
INTR>INTR_ADJE
INTR>INTR_PRON
PRON>PERS_PRON
PRON>single_pronoun
PRON>plural_pronoun
PRON>reflective_pronoun
PRON>relative_pronoun
PRON>indefinitive_pronoun
PRON>distributive_pronoun
PERS_PRON>first_personal_pronoun
PERS_PRON>second_personal_pronoun
PERS_PRON>third_personal_pronoun
ADJE>QUAN_ADJE
ADJE>possessive_adjective
ADJE>distributive_adjective
ADJE>QUAL_ADJE
QUAN_ADJE>definitive q. adjective
QUAN_ADJE>indefinitive q. adjective
QUAL_ADJE>AVVL positive_degree
QUAL_ADJE>comparative_degree THAN
QUAL_ADJE>superlative_degree
COMP_QUAL_ADJE>more positive_degree
SUPE_QUAL_ADJE>most positive_degree
PPL>PP PPL
PPL>e
RP>REPRL VPASIF
RP>E
REPRL>RELA_PRON
```

```
REPRL>e
VERBAL>TO VP
VERBAL>VP
NPL>NP NPE
NPE>CONJ NPL
NPE>e
CONJ>,
CONJ>coordinative_conj
CONJ>correlative_conj
CONJ>subordinate_conj
AVL>e
ADVE>adverb_of_manner
ADVE>adverb_of_time
ADVE>adverb_of_place
FRON>FRON_ARTI
BACK>BACK_ARTI
BACK>POSS_ADJE
POSS_ADJE>both NT CONJ
POSS_ADJE>berween NP and
ANP>demonstrative_adj ANPL
ANP>ADJEL NT VPL
ANPL>ANP
ANPL>e
VPL>VPASIF
VPL>e
```

```
ADJEL>ADJE ADJEL
ADJEL>e
NT>NON NTL
NT>PRON RPL
NT>proper_noun NTL
RPL>reflective_pronoun
RPL>e
NTL>NT
NTL>e
PP>PREP NP PPL
PREP>time_preposition
PREP>direct_preposition
PREP>place_preposition
PREP>reason_preposition
PREP>agent_preposition
NON>single_noun
NON>plural_noun
```

Ket : penulisan terminal ditulis dengan lengkap sesuai dengan jenis katanya, walaupun format sebenarnya adalah huruf besar semua dan memakai batas empat huruf

Jenis-jenis kata

Binnary Code	Integer	Type	CFG Code
0000000000000	0	Unknown word	UNKN
0000000010010	18	Tanda Baca	TDBACA
0001000000000	512	Verb	VERB
0001000000001	513	First Verb	FIRS_VERB
0001000000010	514	Second Verb	SECO_VERB
0001000000011	515	Third Verb	THIR_VERB
0001000000100	516	Verb Ing	ING_VERB
0001001000000	576	Intransitive Verb	INTR_VERB
0001010000000	640	Semi Intransitive Ver	SEMI_INTR_VERB
0001011000000	704	Transitive Verb	TRAN_VERB
0001011001000	712	Direct Transitive Ver	DIRE_TRAN_VERB
0001011010000	720	Indirect Transitive V	INDI_TRAN_VERB
0001100000000	768	Linking Verb	LINK_VERB
0001101000000	789	Lingking Adjective	ADJE_VERB
0010000000000	1024	Article	ARTI
0010001000000	1088	Front Article	FRON_ARTI
0010010000000	1152	Back Article	BACK_ARTI
0011000000000	1536	Noun	NOUN
0011000000001	1537	Singular	SING_NOUN
0011000000010	1538	Plural	PLUR_NOUN
0011000001000	1544	Neuter Noun	NEUT_GEND_NOUN
0011000010000	1552	Feminime Noun	FEMI_GEND_NOUN
0011000011000	1560	Masculine Noun	MASC_GEND_NOUN
0011000100000	1568	Common Gender	COMM_GEND_NOUN
0011001000000	1600	Common Noun	COMM_NOUN
0011010000000	1664	Proper Noun	PROP_NOUN
0011011000000	1728	Abstract Noun	ABST_NOUN
0011100000000	1792	Collective Noun	COLT_NOUN
0100000000000	2048	Adjective	ADJE
0100001000000	2112	Quality Adjective	QUAL_ADJE
0100001001000	2120	Positif Degree	POSI_QUAL_ADJE
0100001010000	2128	Comparative Degree	COMP_QUAL_ADJE
0100001011000	2136	Superlative Degree	SUPE_QUAL_ADJE
0100010000000	2176	Quantity Adjective	QUAN_ADJE
0100010001000	2184	Definitive Quantity	DEFI_QUAN_ADJE
0100010010000	2192	Indefinitive Quantity	INDE_QUAN_ADJE
0100011000000	2240	Possessive Adjective	POSS_ADJE
0100100000000	2304	Distributive Adjectiv	DIST_ADJE
0100101000000	2368	Demonstrative Adjecti	DEMO_ADJE
0101000000000	2560	Pronoun	PRON
0101000000001	2561	Singular Pronoun	SING_PRON
0101000000010	2562	Plural Pronoun	PLUR_NOUN
0101001000000	2624	Personal Pronoun (PP)	PERS_PRON
0101001001000	2632	PP First Person	FIRS_PERS_PRON
0101001010000	2640	PP Second Person	SECO_PERS_PRON
0101001011000	2648	PP Third Person	THIR_PERS_PRON
0101010000000	2688	Possessive Pronoun	POSS_PRON
0101011000000	2752	Reflexive Pronoun	REFL_PRON

Jenis-jenis kata

0101100000000	2816	Demonstrative Pronoun	DEMO_PRON
0101101000000	2880	Relative Pronoun	RELA_PRON
0101110000000	2944	Indefinite Pronoun	INDE_PRON
0101111000000	3008	Distributive Pronoun	DIST_PRON
0110000000000	3072	Preposition	PREP
0110001000000	3136	Preposition Of Place	PLAC_PREP
0110010000000	3200	Preposition Of Direct	DIRE_PREP
0110011000000	3264	Preposition Of Time	TIME_PREP
0110100000000	3328	Preposition Of Agent	AGEN_PREP
0110101000000	3392	Preposition Of Reason	REAS_PREP
0111000000000	3584	Auxiliary	AUXI
0111001000000	3648	First Auxiliary	FIRS_AUXI
0111010000000	3712	Second Auxiliary	SECO_AUXI
0111011000000	3776	Third Auxiliary	THIR_AUXI
1000000000000	4096	TO BE	TOBE
1000001000000	4160	First TOBE	FIRS_TOBE
1000010000000	4224	Second TOBE	SECO_TOBE
1000011000000	4288	Third TOBE	THIR_TOBE
1001000000000	4608	Conjunction	CONJ
1001001000000	4672	Coordinate Conjunct	COOR_CONJ
1001010000000	4736	Correlative Conjuncti	CORR_CONJ
1001011000000	4800	Subordinate Conjuncti	SUBO_CONJ
1010000000000	5120	Interogative	INTR
1010001000000	5184	Interogative Single	NONE_INTR
1010010000000	5248	Interogative Pronoun	PRON_INTR
1010011000000	5312	Interogative Adjectiv	ADJE_INTR
1011000000000	5632	Intersection	INTE
1100000000000	6144	Adverb	ADVE
1100001000000	6208	Adverb Of Manner	MANN_ADVE
1100010000000	6272	Adverb Of Time	TIME_ADVE
1100011000000	6336	Adverb Of Place	PLAC_ADVE
1100100000000	6400	Adverb Of Frequency	FREQ_ADVE
1100101000000	6464	Adverb Of Degree	DEGR_ADVE
1100110000000	6528	Adverb Of Negation	NEGA_ADVE
1101000000000	6656	Helping Word	HELP

Analisa Jenis Token

No	Jenis	Macam	Fungsi	Contoh
1	Tanda baca	1. Titik(.)	• akhir kalimat • singkatan • Penomoran Bab • Pemomoran biasa • Nilai mata uang dollar	• – • (Spec.) • Fig. (6.4), 2. • 5. Teacher • $ 500,000.00
		2. koma(,)	• Akhir sub kalimat • Pemisah token • Pemisah (conjunction) • Nilai Uang ribuan	• – • (P,P) • stone, land • $ 500,000.00
		3. titik dua (:)	• pendetail • jam/waktu	• like : • 05:48 GMT
		4. titik koma (;)	• pemisah sub/satu kalimat	• I hungry;
		5. tanda seru (!)	• akhir kalimat	• Be Quite !
		6. tanda tanya (?)	• akhir kalimat	• Why ?
		7. kurung (())	• mengurung kata/frase	• but (Budi)
		8. Garis miring (/),	• pemisah kata sejenis	• go/walkout
		9. Tanda Dan (&)	• kata sambung	• I & you
		10. Petik (')	• pemilik	• Budi's book
			• pemisah aux	• I'm, aren't
		11. Petik dua (")	• kata penting	• like "mami"
			• kalimat tertentu (harus 2)	• "I didn't .."
		12. tanda hubung (-)	• ganti baris • compound word (majemuk)	• I make so-mething • Boy-friend
			• tanggal	• 05-02-1999
		13. Spasi	• pemisah token	• B = 4X
			• pemisah kata	• Mary and I
		14. Kurung [] 15. Kurung {}	• Fungsi Pustaka kt. Tertentu • Sementara samakan ()	• .. [1,199]
2.	Operator Matematika	1. tambah (+) 2. kurang (-) 3. kali (x)	• Operator (tengah) • Bil Negatif | operator • Operator (Nama/Var)	• A + B = G • -4 | 4 – 5 • Z x D
		4. bagi (/) 5. bagi (:)	• Times = X • pembagi (opr)	• Mem X 23 • 5/5
		6. sama dengan(=) 7. lebih besar (>) 8. lebih kecil (<) 9. tidak sama dengan (≠)	• jam • pembagi / pembanding • operator • idem • idem	• 05:40 • 5:2 • R=5 ; 4+C=3 • 45 + 4 > M • dst
		10. kurung () 11. Kurung []	• idem • harus menutup persamaan	

Analisa Jenis Token

		12. Kurung {}	• samakan dengan kurung	
3	Angka	'0'..'9'	• Variabel / Nama • Angka • Jam • Nomor • Tanggal	• V9 make result • 45 times • 3:45/04:45:45 • Fig. 4. | No. 5.5 • 05-02-1999
4	Huruf	1. Kecil 'a'..'z', 2. Besar 'A'..'Z"	• Kata biasa • Penomoran • Singkatan jika diikuti besar (kt) semua • Awal kalimat • Variabel / Nama • Penomoran • Proper Noun (Name)	• want • 3.4b ; 1.4C. • ACM • Make • MKas9 • 3.4b ; 1.4C. • Malinda; Figure 2
5	Karakter aneh	1. Add (@) 2. Tak hingga (~) 3. Dollar ($) 4. Crash (#) 5. Bintang (*) 6. Persen (%) 7. Underscor (_) 8. Determinan (|) 9. Dll (not in keyb)	• Nama Email • d • Standart uang Dollar • d • d • Persen (follow Numeric) • d • Besar variabel bilangan • d	• d • d • $500.00 • d • d • 300 % • d • | A |

Token yang khusus dikenal dalam Bahasa Inggris :

Jenis	Jenis / ciri	Keterangan & Contoh
1. Kata awal kalimat	• huruf depan besar	• What is this ?
2. Kata biasa	• huruf kecil semua	• What is this ?
3. Kata singkatan	• huruf besar semua	• ICW (Indonesian Corruption Watch)
4. Kata penyingkat	• huruf besar diakhiri titik	• Fig. 24 | fig. 34
5. Nama (Proper)	• ditengah tapi awal huruf besar	• Stallone Ramboo
6. Penomoran	• angka diikuti huruf dan titik	• 1.1b ; 1b. ; c.
7. Variabel / Name	• jika huruf diikuti angka atau sebaliknya (tak teratur) tanpa titik	• 5V ; R3 ; R34
8. Angka	• terdiri karakter angka semua	• 90 ; 678
9. Tanggal	• tiga susunan angka dipisah – tahun minimal dua angka	• 12-03-1999
10. Jam	• Dua atau tiga angka dipisah : huruf awal bisa satu atau dua	• 12:34:34 ; 2:34
11. Tanda baca	• Akhir kalimat	• I remember your advice.
12. Nilai uang	• Diawali dollar, Rp atau lainnya lalu angka.	• US $ 100.00.