BAB II
LANDASAN TEORI

2.1 Sintaksis Bahasa Indonesia

Sintaksis merupakan cabang ilmu bahasa yang membicarakan penataan dan pengaturan kata-kata ke dalam satuan yang lebih besar, yang disebut satuan-satuan sintaksis, yakni kata, frase, klausa, kalimat, dan wacana.

2.1.1 Fungsi Sintaksis

Fungsi sintaksis adalah semacam kotak-kotak dalam struktur sintaksis yang kedalamnya diisikan kategori-kategori tertentu. Kotak-kotak itu bernama subjek (S), predikat (P), objek (O), komplemen (Kom), dan keterangan (Ket). Namun di dalam praktik berbahasa urutannya tidak sama. Secara formal fungsi S dan P harus selalu adadalam setiap klausa karena keduanya saling berkaitan dalam hal ini bisa dikatakan, bahwa S adalah bagian klausa yang menandai apa yang dinyatakan oleh pembicaraan mengenai S.

Objek adalah bagian dari verba yang menjadi predikat dalam klausa. Kehadirannya sangat ditentukan oleh ketransitifan verba tersebut. Artinya jika verba bersifat transitif maka objek akan muncul, tetapi kalau verbanya bersifat tak transitif maka objek tidak akan ada. Terdapat dua macam objek yaitu objek afektif dan objek efektif. Objek afektif adalah objek yang bukan merupakan hasil perbuatan predikat. Sebaliknya objek efektif adalah objek yang merupakan hasil perbuatan predikat.

Komplemen (komp) atau pelengkap adalah bagian dari P verba yang menjadikan P itu menjadi pelengkap. Kedudukannya mirip dengan O, hanya perbedanya jika O keberadaannya ditentukan oleh sifat verbanya yang transitif. Sedangkan komp keberadaannya bukan ditentukan oleh faktor ketransitifan, melainkan oleh faktor keharusan untuk melengkapi P. Unsur S, P, O dan komplemen merupakan inti klausa, sedangkan unsur keterangan merupakan bagian luar inti klausa. Hal tersebut karena kedudukan keterangan di dalam klausa lebih fleksibel, artinya dapat berada pada awal klausa maupun pada akhir klausa [6].

2.1.2 Kategori Sintaksis

Kategori sintaksis merupakan jenis atau tipe kata atau frase yang menjadi pengisi fungsi-fungsi sintaksis. Kategori sintaksis berkenaan dengan istilah nomina (N), verba (V), ajektifa (A), adverbia (Adv), numeralia (Num), preposisi (Prep), konjungsi (Konj), dan pronominal (Pron). Dalam hal ini N, V, dan A merupakan kategori utama, sedangkan yang lain merupakan kategori tambahan. Pengisi fungsi sintaksis dapat berupa kata dapat pula berupa frase, sehingga di samping ada kata nomina ada pula frase nomina (FN), di samping kata verba ada pula frase verba (FV), dan di samping ada kata ajektifa ada pula frase ajektifa (FA). Selain itu di samping ada kata berkategori adverbial ada pula frase adverbia (FAdv), di samping kata berkategori numeralia ada pula frase numeral (FNum), dan di samping kata berkategori preposisi ada pula frase preposisional (FProp).

Secara formal kategori N atau FN mengisi fungsi S dan atau O pada klausa verba. Bisa juga mengisi fungsi P pada klausa nominal. Kategori V atau FV secara formal mengisi fungsi P pada klausa verba, dan kategori A atau FA mengisi fungsi P pada klausa ajektifa [6]. Maka, secara formal pengisi fungsi-fungsi sintaksis dapat dilihat pada Tabel 2.1 berikut:

Tabel 2.1 Pengisi Fungsi-Fungsi Sintaksis

S	P	O
N	N	N
FN	FN	FN
	V	
	FV	
	A	
	FA	

2.1.3 Adverbia

Adverbia adalah kategori yang mendampingi nomina, verba, dan ajektifa dalam pembentukan frase atau dalam pembentukan sebuah klausa. Sebagai pendamping kelas terbuka, adverbia dengan kategori yang didampinginya membentuk sebuah frase untuk mengisi salah satu fungsi sintaksis. Kategori yang didampingi tergantung dari makna *inheren* yang dimiliki oleh adverbia [6]. Adverbia yang menyatakan makna antara lain:

1. Sangkalan

   Adverbia sangkalan adalah adverbia yang menyatakan ‘ingkar’ atau ‘menyangkal’ akan kategori yang didampinginya. Yang termasuk adverbia sangkalan adalah kata-kata bukan, tidak, tanpa, dan tiada.

2. Penjumlahan (kuantitas)

   Adverbia penjumlahan adalah adverbia yang menyatakan ‘banyak’ atau ‘kuantitas’ terhadap kategori yang didampingi. Yang termasuk adverbia penjumlahan adalah kata-kata banyak, sedikit, beberapa, semua, seluruh, sejumlah, separuh, setengah, kira-kira, sekitar, dan kurang lebih.

3. Pembatasan

   Adverbia pembatasan adalah adverbia yang menyatakan ‘batas dari suatu hal’. Yang termasuk adverbia ini adalah kata-kata hanya, Cuma, saja, dan belaka.

4. Derajat

   Adverbia derajat atau kualitas adalah adverbia yang menyatakan tingkat mutu keadaan atau kegiatan. Yang termasuk adverbia ini adalah sangat, amat, sekali, paling, lebih, cukup, kurang, agak, hampir, rada, maha, nian, dan terlalu.

5. Kala (waktu)

   Adverbia kala adalah adverbia yang menyatakan waktu tindakan dilakukan. Yang termasuk adverbia ini adalah kata-kata sudah, telah, mau, sedang, lagi, tengah, akan, bakal, dan hendak.

6. Keselesaian

   Adverbia keselesaian adalah yang menyatakan tindakan atau perbuatan apakah sudah selesai, belum selesai, atau sedang dilakukan. Yang termasuk adverbia ini adalah kata-kata belum, baru, mulai, sedang, lagi, tengah, masih, sudah, telah, sempat, dan pernah. Semuaya berposisi di sebelah kiri verba atau ajektifa (yang mengisi fungsi predikat).

7. Kepastian

   Adverbia kepastian adalah adverbia yang menyatakan tindakan atau keadaan yang pasti terjadi maupun yang diragukan kejadiannya. Adverbia kepastian adalah kata-kata pasti, memang, tentu, agaknya, dan rupanya.

8. Menyungguhkan

Adverbia menyungguhkan adalah adverbia yang menyatakan ‘kesungguhan’ atau ‘menguatkan’. Yang termasuk adverbia ini adalah kata-kata sesungguhnya, sebenarnya, sebetulnya, dan memang.

9. Keharusan

   Adverbia keharusan adalah adverbia yang menyatakan ‘keharusan’ atau ketidakharusan dilakukan dilakukannya sesuatu’. Yang termasuk adverbia keharusan adalah kata-kata harus, wajib, mesti, boleh, dan jangan.

10. Keinginan

    Adverbia keinginan adalah adverbia yang menyatakan ‘keinginan’. Yang termasuk adverbia keinginan adalah kata-kata ingin, mau, hendak, suka, dan segan.

11. Frekuensi

    Adverbia frekuensi adalah adverbia yang menyatakan ‘berapa kali suatu tindakan atau perbuatan dilakukan atau terjadi’. Yang termasuk adverbia frekuensi adalah kata-kata sekali, sesekali, sekali-kali, sekali-sekali, jarang, kadang-kadang, sering (seringkali), acap (acapkali), biasa, selalu, dan senantiasa.

12. Penambahan

    Adverbia penambahan adalah adverbia yang menyatakan penambahan terhadap kategori yang didampingi. Yang termasuk adverbia penambahan adalah kata-kata pula, juga, dan jua.

13. Kesanggupan

    Adverbia kesanggupan adalah adverbia yang digunakan untuk menyatakan ‘kesanggupan’. Yang termasuk adverbia kesanggupan adalah kata-kata sanggup, dapat, dan bisa.

14. Harapan

    Adverbia harapan adalah adverbia yang menyatakan ‘harapan akan terjadinya sesuatu tindakan, hal, atau keadaan. Yang termasuk adverbia harapan adalah kata-kata moga-moga, semoga, mudah-mudahan, hendaknya, sepatutnya, sebaiknya, seyogianya, seharusnya, dan sepantasnya.

2.1.4 Konjungsi

Konjungsi adalah kategori yang menghubungkan kata dengan kata, klausa dengan klausa, atau kalimat dengan kalimat, bisa juga antara paragraf dengan paragraf. Ditinjau dari kedudukan konstituen yang dihubungkan dibedakan adanya konjungsi koordinatif dan konjungsi subordinatif.

2.1.4.1 Konjungsi Koordinatif

Konjungsi koordinatif adalah konjungsi yang menghubungkan dua buah konstituen yang kedudukannya sederajat. Konjungsi koordinatif dibedakan atas konjungsi yang menyatakan:

1. Penjumlahan, yaitu konjungsi dan, dengan, dan serta.
2. Pemilihan, yaitu konjungsi atau
3. Pertentangan, yaitu konjungsi tetapi, namun, sedangkan, dan sebaliknya.
4. Pembetulan, yaitu konjungsi melainkan, dan hanya.
5. Penegasan, yaitu konjungsi bahkan, malah, lagipula, apalagi, dan jangankan.
6. Pembatasan, yaitu konjungsi kecuali, dan hanya.
7. Pengurutan, yaitu konjungsi lalu, kemudian, dan selanjutnya.
8. Penyamaan, yaitu konjungsi yakni, bahwa, adalah, dan ialah.
9. Penyimpulan, yaitu konjungsi jadi, karena itu, oleh sebab itu, maka, maka itu, dengan demikian, dan dengan begitu.

2.1.4.2 Konjungsi Subordinatif

Konjungsi subordinatif adalah konjungsi yang menghubungkan dua buah konstituen yang kedudukannya tidak sederajat. Ada konstituen atasan dan konstituen bawahan, konjungsi subordinatif dibedakan atas konjungsi yang menyatakan:

1. Penyebaban, yaitu konjungsi sebab, dan karena.
2. Persyaratan, yaitu konjungsi kalau, jika, jikalau, bila, apabila, bilamana, dan asal.
3. Tujuan, yaitu konjungsi agar, dan supaya.
4. Penyungguhan, yaitu konjungsi meskipun, biarpun, walaupun, sungguhpun, dan sekalipun.

5. Kesewaktuan, yaitu konjungsi ketika, tatkala, sewaktu, sebelum, sesudah, dan sehabis.
6. Pengakibatan, yaitu konjungsi sampai, hingga, dan sehingga.
7. Perbandingan, yaitu konjungsi seperti, sebagai, dan laksana.

2.1.5 Preposisi

Preposisi adalah kategori yang terletak di sebelah kiri nomina sehingga terbentuk sebuah frase eksosentrik untuk mengisi fungsi keterangan dalam sebuah klausa atau kalimat. Preposisi dapat dibedakan atas preposisi yang menyatakan:

1. Tempat berada

   Preposisi tempat berada menyatakan tempat terjadinya peristiwa, tindakan, atau keadaan terjadi. Yang termasuk preposisi tempat berada adalah kata-kata di, pada, dalam, dan antara.
2. Tempat asal

   Preposisi tempat asal adalah preposisi yang menyatakan tempat berasalnya nomina yang mengikuti. Yang termasuk preposisi tempat asal adalah kata-kata dari. Penggunaannya adalah diletakkan di sebelah kiri nomina yang menyatakan tempat.
3. Tempat tujuan

   Preposisi tempat tujuan adalah preposisi yang menyatakan tempat yang dituju dari perbuatan atau tindakan yang dilakukan. Ada dua preposisi tempat tujuan, yaitu preposisi ke dan kepada.
4. Asal bahan

   Preposisi asal bahan adalah preposisi yang menyatakan asal bahan pembuat sesuatu. Yang termasuk preposisi asal bahan adalah kata-kata dari, yang diletakkan di sebelah kiri nomina yang menyatakan bahan pembuat sesuatu. Sementara subjeknya merupakan barang jadian atau buatan.
5. Asal waktu

   Preposisi asal waktu adalah preposisi yang menyatakan waktu mulai suatu kejadian, peristiwa, atau tindakan. Yang termasuk preposisi asal waktu adalah kata dari, sejak.
6. Waktu tertentu

Preposisi waktu tertentu adalah preposisi yang menyatakan awal dan akhir dari suatu kejadian, peristiwa, atau tindakan. Yang termasuk preposisi waktu tertentu adalah kata dari disertai dengan kata sampai.

7. Tempat tertentu

   Preposisi tempat tertentu adalah preposisi yang menyatakan awal tempat kejadian hingga akhir tempat kejadian. Preposisi tempat tertentu berupa kata dari yang disertai dengan kata sampai. Dalam hal tersebut kata dari dapat diganti dengan kata sejak, dan kata sampai diganti dengan kata hingga.

8. Perbandingan

   Preposisi perbandingan adalah preposisi yang menyatakan perbandingan antara dua tindakan atau dua hal. Preposisi perbandingan ini adalah kata daripada.

9. Pelaku

   Preposisi pelaku adalah preposisi yang menyatakan pelaku perbuatan atau tindakan yang disebutkan dalam predikat klausa. Preposisi pelaku ini adalah kata oleh.

10. Alat

    Preposisi alat adalah preposisi yang menyatakan alat untuk atau dalam melakukan perbuatan atau tindakan yang dinyatakan oleh predikat klausa yang bersangkutan. Preposisi alat yang ada adalah kata dengan, dan berkat.

11. Hal

    Preposisi hal adalah preposisi yang menyatakan hal yang akan disebutkan dalam predikat klausanya. Preposisi hal yang ada adalah perihal, tentang, dan mengenai. Ketiganya dapat saling menggantikan. Penggunaannya adalah dengan meletakkannya di sebelah kiri nomina atau frase nomina yang mengikutinya.

12. Pembatasan

    Preposisi pembatasan adalah preposisi yang menyatakan batas akhir dari suatu tindakan, tempat, atau waktu yang disebutkan dalam predikat klausanya. Preposisi pembatasan ini adalah kata sampai, dan hingga. Secara umum keduanya bisa saling menggantikan.

13. Tujuan

Preposisi tujuan adalah preposisi yang menyatakan tujuan atau maksud dari perbuatan atau tindakannya yang disebutkan dalam predikat klausanya. Preposisi tujuan ini adalah kata agar dan supaya yang secara umum dapat saling menggantikan.penggunaannya adalah dengan cara meletakkan di sebelah kiri kata atau frase berkategori ajektifa atau verba keadaan.

2.1.6 Penyusunan Frase

Frase adalah satuan sintaksis yang tersusun dari dua buah kata atau lebih, yang di dalam klausa menduduki fungsi-fungsi sintaksis. Dilihat dari kedudukan kedua unsurnya, dibedakan adanya frase koordinatif dan frase subordinatif. Frase koordinatif yaitu yang kedudukan kedua unsurnya sederajat, sedangkan frase subordinatif yaitu yang kedudukan kedua unsurnya tidak sederajat. Ada yang berkedudukan sebagai unsur atasan yang disebut inti frase dan ada yang berkedudukan sebagai bawahan yang disebut sebagai tambahan penjelas frase [6]. Frase dibagi menjadi beberapa kelompok:

1. Frase nomina koordinatif (FNK)
2. Frase nomina subordinatif (FNS)
3. Frase verba koordinatif (FVK)
4. Frase verba subordinatif (FVS)
5. Frase ajektifa koordinatif (FAK)
6. Frase ajektifa subordinatif (FAS)
7. Frase preposisional (Fprep)

2.1.6.1 Penyusunan Frase Nomina

Frase nomina (FN) adalah frase yang dapat mengisi fungsi subjek atau objek didalam klausa. Menurut strukturnya dapat dibedakan adanya frase nomina koordinatif (FNK) dan frase nomina subordinatif (FNS).

1. Penyusunan Frase Nomina Koordinatif (FNK)

   FNK dapat disusun dari:

   a. Dua buah kata berkategori nomina yang merupakan pasangan dari antonim relasional. Contoh: ayah ibu, pembeli penjual, guru murid, pembicara pendengar, penatar petatar, siang malam.

b. Dua buah kata berkategori nomina yang merupakan anggota dari suatu medan makna. Contoh: sawah ladang, ayam itik, kampung halaman, cabe bawang, semen pasir, tikar bantal.

2. Penyusunan Frase Nomina Subordinatif (FNS)

Frase nomina subordinatif dapat disusun dari nomina + nomina (N + N), nomina + verba (N + V), nomina + ajektifa (N + A), adverbia + nomina (Adv + N), nomina + adverbia (N + Adv), nomina + numeralia (N + Num), numeralia + nomina (Num + N), dan nomina + demonstratifa (N + Dem).

a. FNS yang berstruktur N + N

Beberapa contoh FNS yang berstruktur N + N dapat dilihat pada tabel 2.2.

Tabel 2.2 Contoh Frase Nomina (N + N)

Makna Gramatikal	Contoh Frase
milik	tongkat kakek
bagian	tengah semester
asal bahan	soto ayam
asal tempat	putri solo
hasil	motor cina
campuran	kopi susu
jenis	bunga anggrek
gender	sapi jantan
model	rumah eropa
seperti	akar rambut
menggunakan, memakai	kapal layar
peruntukan	obat mata
ada di	kapal laut
wadah	kaleng susu
letak	laci atas
dilengkapi	truk gandeng
sasaran	penulisan buku
pelaku	pukulan Mohamad ali
alat	lempar cakram

b. FNS yang berstruktur N + V

Beberapa contoh FNS yang berstruktur N + V dapat dilihat pada tabel 2.3.

Tabel 2.3 Contoh Frase Nomina (N + V)

Makna Gramatikal	Contoh Frase
tempat	kolam renang
kegunaan	mobil derek
yang di	roti bakar
yang biasa melakukan	tukang pukul

c. FNS yang berstruktur N + A

Beberapa contoh FNS yang berstruktur N + A dapat dilihat pada tabel 2.4.

Tabel 2.4 Contoh Frase Nomina (N + A)

Makna Gramatikal	Contoh Frase
keadaan	ban kempes
derajat	perwira menengah
rasa, bau	minyak wangi
bentuk	gedung bundar

d. FNS yang berstruktur Adv + N

Beberapa contoh FNS yang berstruktur Adv + N dapat dilihat pada tabel 2.5.

Tabel 2.5 Contoh Frase Nomina (Adv + N)

Makna Gramatikal	Contoh Frase
ingkar	bukan saya
kuantitas	banyak uang
batas	cuma meja

e. FNS yang berstruktur N + Adv

Sejauh ini FNS yang berstruktur N + Adv hanya bermakna gramatikal ‘pembatasan’. Dalam hal ini hanya ada sebuah adverbia pembatasan yaitu ‘saja’. Contoh: air saja (taka da yang lain), uang saja (bukan benda lain), dia saja (orang lain tidak ada), garam saja (tanpa bumbu lain), pensil saja (tidak pakai pena).

f. FNS berstruktur Num + N

Beberapa contoh FNS yang berstruktur Num + N dapat dilihat pada tabel 2.6.

Tabel 2.6 Contoh Frase Nomina (Num + N)

Makna Gramatikal	Contoh Frase
banyaknya	tiga orang India
himpunan	dua gelas air

g. FNS berstruktur N + Num

Sejauh ini FNS yang berstruktur N + Num memiliki makna gramatikal ‘tingkat’, dapat disusun apabila N-nya memiliki komponen makna (+ terhitung) dan numeralianya memiliki komponen makna (+ tingkat). Contoh: anak kelima, simpangan kedua, rumah kelima, kursi ketiga, juara kedua.

h. FNS yang berstruktur N + Dem

Sejauh ini FNS yang berstruktur N + Dem memiliki makna gramatikal ‘penentu’, dapat disusun apabila N-nya memiliki komponen makna (+ benda umum) dan unsur kedua berkategori pronominal demonstratifa (ini, itu). Contoh: anak itu, pegawai ini, universitas itu, topi ini, mereka itu.

2.1.6.2 Penyusunan Frase Verba

Frase verba adalah frase yang mengisi atau menduduki fungsi predikat pada sebuah klausa. Dilihat dari kedudukan di antara kedua usur pembentuknya dibedakan adanya frase verba koordinatif (FVK) dan frase subordinatif (FVS).

1. Penyusunan Frase Verba Koordinatif (FVK)

Frase verba koordinatif dapat disusun dari:

a. Dua buah kata berkategori verba yang merupakan anggota dari antonim relasional, dan memiliki makna gramatikal ‘menggabungkan’ sehingga di antara keduanya dapat disisipkan kata ‘dan’. Contoh: tambah kurang, jual beli, pulang pergi, mundur maju, naik turun.

b. Dua buah kata berkategori verba yang merupakan anggota dari satu medan makna dan memiliki makna gramatikal ‘menggabungkan’ sehingga di antara kedua unsurnya dapat disisipkan kata ‘dan’. Contoh: makan minum, usap raba, peluk cium, makan pakai, dengar lihat.

2. Penysunan Frase Verba Subordinatif (FVS)

Frase verba subordinatif dapat disusun dari adverbia + verba (Adv + V), verba + adverbia (V + Adv), verba + nomina (V + N), dan verba + ajektifa (V + A).

a. FVS yang berstruktur Adv + V

Beberapa contoh FVS yang berstruktur Adv + V dapat dilihat pada tabel 2.7.

Table 2.7 Contoh Frase Verba (Adv + V)

Makna Gramatikal	Contoh Frase
Ingkar	tidak membayar
Frekuensi	jarang mandi
Kuantitas	banyak menulis
waktu (kala)	sedang belajar
keinginan	mau mandi
keselesaian	sudah hadir
keharusan	mesti berobat
Kepastian	pasti dibayar
pembatasan	hanya diam

b. FVS yang berstruktur V + Adv

Beberapa contoh FVS yang berstruktur V + Adv dapat dilihat pada table 2.8.

Table 2.8 Contoh Frase Verba (V + Adv)

Makna Gramatikal	Contoh Frase
Berulang	tidur lagi
ikut serta	naik pula

c. FVS yang berstruktur V + N

FVS yang berstruktur V + N memiliki makna gramatikal ‘alat’, dapat disusun apabila unsur pertama berkategori verba yang memiliki komponen makna (+ tindakan) atau (+ perbuatan), sedangkan unsur kedua berkategori nomina yang memiliki komponen makna (+ alat). Contoh: terjun paying, lempar cakram, lari gawang, lompat galah, uji materi.

d. FVS yang berstruktur V + A

FVS yang berstruktur V + A memiliki makna gramatikal ‘keadaan’ atau ‘sifat’ dapat disusun apabila unsur pertama berkategori verba yang memiliki komponen makna (+ tindakan) atau (+ perbuatan), sedangkan unsur kedua berkategori ajektifa yang memiliki komponen makna (+ keadaan) atau (+ sifat). Contoh: lompat jauh, loncat indah, terjun bebas, jalan cepat, membaca nyaring.

2.1.6.3 Penyusunan Frase Ajektifa

Frase ajektifa adalah frase yang mengisi atau menduduki fungsi predikat dalam sebuah klausa ajektifa. Dilihat dari kedudukan kedua unsurnya dibedakan adanya frase ajektifa koordinatif (FAK) dan frase ajektifa subordinatif (FAS).

1. Penyusunan Frase Ajektifa Koordinatif (FAK)
   a. Dua buah kata berkategori ajektifa yang merupakan anggota dari antonim relasional dan memiliki makna gramatikal ‘pilihan’, sehingga di antara kedua dapat disisipkan kata 'ata‘'. Contoh: baik buruk, tua muda, jauh dekat, lama baru. Untuk bentuk kata yang tidak mempunyai pasangan antonym, digunakan rumus: tidaknya. Contoh: laku tidaknya, habis tidaknya, benar tidaknya, suka tidaknya, kering tidaknya.
   b. Dua buah kata berkategori ajektifa yang merupakan anggota dari pasangan bersinonim, dan memiliki makna gramatikal ‘sangat’. Contoh: tua renta, terang benderang, cantik molek, muda belia, segar bugar.
   c. Dua buah kata berkategori ajektifa yang maknanya sejalan tidak bertentangan dan memiliki makna gramatikal ‘himpunan’ sehingga di antara keduanya dapat disisipkan kata ‘dan’. Contoh: bulat panjang, gemuk pendek, tinggi kurus, kecil mungil, lurus mulus.

d. Dua buah kata berkategori ajektifa yang maknanya tidak sejalan (bertentangan) dan memiliki makna ‘berkebalikan’ sehingga di antara kedua unsurnya harusnya disisipkan kata ‘tetapi’. Contoh: murah tetapi bagus, kecil tetapi mungil, besar tetapi jelek, repot tetapi menyenangkan, sedih tetapi senang.

2. Penyusunan Frase Ajektifa Subordinatif (FAS)

Frase ajektifa subordinatif disusun dengan struktur ajektifa + nomina (A + N), ajektifa + ajektifa (A + A), ajektifa + verba (A + V), adverbia + ajektifa (Adv + A), dan ajektifa + adverbia (A + Adv). Aturannya sebagai berikut:

a. FAS yang berstruktur A + N dan memiliki makna gramatikal ‘seperti’ apabila unsur pertama berkategori ajektifa dan memiliki komponen makna (+ makna) dan unsur kedua berkategori nomina dan memiliki komponen makna (+ perbandingan), sehingga di antara kedua unsurnya dapat disisipkan kata ‘seperti warna’. Contoh: merah darah, kuning emas, hijau daun, biru langit, kuning gading.

b. FAS yang berstruktur A + A dan memiliki makna gramatikal ‘jenis warna’ dapat disusun dari:

   1. Unsur pertama berkategori ajektifa dan berkomponen makna (+ warna) dan unsur kedua berkategori ajektifa dan berkomponen makna (+ cahaya). Contoh: merah terang, biru gelap, putih kelabu, coklat tua, hijau muda.
   2. Unsur pertama berkategori ajektifa dan memiliki komponen makna (+ warna), sedangkan unsur kedua berkategori ajektifa dan berkomponen makna (+ warna) dan (+ benda). Contoh: putih kebiru-biruan, kuning kehijau-hijauan, merah kebiru-biruan, coklat kehitam-hitaman, biru kecoklat-coklatan.

c. FAS yang berstruktur A + V dan bermakna gramatikal ‘untuk’ dapat disusun apabila unsur pertama berkategori ajektifa dan memiliki komponen makna (+ sikap batin), sedangkan unsur kedua berkategori verba dan memiliki komponen makna (+ tindakan) atau (+ kejadian).

Contoh: berani datang, takut pulang, malu bertanya, siap berjuang, berani mati.

d. FAS yang berstruktur Adv + A dan memiliki makna gramatikal ‘ingkar’ dapat disusun apabila unsur pertama berkategori adverbia yang berkomponen makna (+ ingkar) dan unsur kedua berkategori ajektifa dan berkomponen makna (+ keadaan) atau (+ sikap batin). Contoh: tidak malas, tidak takut, tidak nakal, tidak bodoh.
e. FAS yang berstruktur Adv + A dan bermakna gramatikal ‘derajat’ dapat disusun bila unsur pertama berkategori adverbia dan berkomponen makna (+ derajat) atau (+ tingkat), sedangkan unsur kedua berkategori ajektifa dan berkomponen makna (+ keadaan) atau (+ sifat). Contoh: sangat indah, kurang bagus, lebih buruk, cukup baik, lebih pandai.
f. FAS yang berstruktur A + Adv dan bermakna gramatikal ‘sangat’ atau ‘tingkat superlatif’ dapat disusun apabila unsur pertama berkategori ajektifa dan bermakna gramatikal (+ keadaan), sedangkan kedua berkategori adverbia dan berkomponen makna (+ paling) dalam bentuk kata ‘sekali’. Contoh: indah sekali, bagus sekali, merah sekali.

2.1.6.4 Penyusunan Frase Preposisional

Frase preposisional adalah frase yang berfungsi sebagai pengisi fungsi keterangan di dalam sebuah klausa. Frase preposisional ini bukan frase koordinatif maupun frase subordinatif, melainkan frase eksosentrik. Jadi, di dalam frase ini tidak ada unsur inti dan unsur tambahan. Kedua unsurnya merupakan satu kesatuan yang utuh.

Frase preposisional tersusun dari kata berkategori preposisi dan kata atau frase berkategori nomina. Beberapa contoh: di pasar, ke dalam kamar, dari rumah sakit, dengan pensil alis, oleh ayah tiri.

2.1.7 Penyusunan Klausa

Klausa adalah satuan sintaksis yang bersifat prediktif. Artinya di dalam satuan atau konstruksi itu terdapat sebuah predikat, bila di dalam satuan tidak terdapat predikat, maka satuan itu bukan sebuah klausa. Kedudukan predikat ini sangat

penting, sebab jenis dan kategori dari predikat itulah yang menentukan hadirnya fungsi subjek (S), fungsi objek (O), fungsi pelengkap, dan sebagainya [6]. Dalam analisis fungsional klausa dianalisis berdasarkan fungsi unsur-unsurnya menjadi S, P, O, PEL, KET, dan dalam analisis kategorial telah dijelaskan bahwa pengisi fungsi S terdiri dari N, fungsi P terdiri dari N, V, Num, FP, fungsi O terdiri dari N, fungsi PEL terdiri dari N, V, Num, dan fungsi KET terdiri dari FP, N [11].

Berdasarkan kategori yang mengisi fungsi P dapat dibedakan menjadi:

1. Klausa verba
2. Klausa nomina
3. Klausa ajektifa
4. Klausa preposisional
5. Klausa numerial

2.1.7.1 Penyusunan Klausa Verba

Secara semantik ada tiga buah jenis verba, yaitu verba tindakan, verba kejadian, dan verba keadaan.

1. Klausa Verba Tindakan

   Klausa verba tindakan dibagi menjadi beberapa kategri:

   a. Klausa Verba Tindakan Bersasaran Tak Berpelengkap

      Klausa tindakan bersasaran tak berpelengkap dapat disusun dari sebuah verba berkomponen makna (+ tindakan) dan (+ sasaran), sehingga klausanya memiliki fungsi sintaksis S, P, dan O. dalam hal ini komponen makna V yang mengisi fungsi P harus sejalan dengan komponen makna yang dimiliki fungsi S dan fungsi O. Contoh dapat dilihat pada table 2.9.

Table 2.9 Contoh Klausa Verba Tindakan Bersasaran Tak Berpelengkap

kucing itu	makan	dendeng
S	P	O
(+ makhluk)	(+ makhluk)	-
-	(+ makanan)	(+ makanan)

   b. Klausa Verba Tindakan Bersasaran Berpelengkap

Klausa tindakan bersasaran berpelengkap dapat disusun dari sebuah verba berkomponen makna (+ tindakan), (+ sasaran), dan (+ pelengkap), sehingga klausanya memiliki fungsi S, P, O, dan Pel. Dalam hal ini tentu saja komponen makna yang dimiliki P harus sejalan dengan komponen makna yang dimiliki fungsi-fungsi lain. Contoh dapat dilihat pada table 2.10.

Table 2.10 Contoh Klausa Verba Tindakan Bersasaran Berpelengkap

Saya	membukakan	ayah	pintu
S	P	Pel	O
(+ manusia)	(+ manusia)	-	-
-	(+ pembuka)	(+ dibukakan)	-
-	(+ bukaan)	-	(+ bukaan)

c. Klausa Verba Tindakan Tak Bersasaran

Klausa verba tindakan tidak bersasaran dapat disusun dari sebuah verba yang memiliki komponen makna (+ tindakan) dan (- sasaran), sehingga klausanya hanya memiliki fungsi S dan fungsi P. Dalam hal ini tentu saja komponen makna yang dimiliki P harus sejalan dengan komponen makna yang dimiliki S. Contoh: Kapal itu berlayar ke Makasar, Mereka sedang berlibur di Bali, Kami berjalan kaki dari rumah ke sekolah.

2. Klausa Verba Kejadian

Klausa verba kejadian dapat disusun dari predikat verba yang memiliki komponen makna (+ kejadian). Dalam hal fungsi sintaksis yang wajib hadir adalah fungsi S dan fungsi P. Fungsi S berupa nomina yang mengalami kejadian seperti disebutkan oleh predikat. Contoh dapat dilihat pada tabel 2.11.

Table 2.11 Contoh Klausa Verba Kejadian

Tanggul Sungai Bengawan Solo	jebol
S	P
yang mengalami kejadian	kejadian

3. Klausa Verba Keadaan

Klausa Verba keadaan dapat disusun dari predikat verba yang memiliki komponen makna (+ keadaan). Dalam hal ini fungsi sintaksis yang muncul hanyalah fungsi S dan fungsi P. Fungsi S berupa nomina yang mengalami

keadaan seperti yang disebutkan oleh predikat. Contoh dapat dilihat pada tabel 2.12.

Tabel 2.12 Contoh Klausa Keadaan

Pintu kamarnya	terbuka
S	P
yang mengalami	keadaan

2.1.7.2 Penyusunan Klausa Nomina

Klausa nomina hanya memiliki fungsi wajib S dan P. Klausa nomina ini dapat disusun dari fungsi S yang berupa kata atau frase berkategori nomina dan P yang berupa kata atau frase berkategori nomina. Klausa nomina, antara lain, dapat disusun jika:

1. Nomina yang mengisi fungsi S merupakan jenis (spesifik) dari nomina pengsisi fungsi P (generik). Contoh dapat dilihat pada tabel 2.13.

Tabel 2.13 Contoh Pertama Penyusunan Klausa Nomina

Mobil itu	kendaraan darat
S	P

2. Nomina yang mengisi fungsi S mempunyai nama pada nomina pengisi fungsi P. Contoh dapat dilihat pada tabel 2.14.

Tabel 2.14 Contoh Kedua Penyusunan Klausa Nomina

Kucing itu	si manis
S	P

3. Nomina pengisi fungsi P adalah profesi (jabatan, pekerjaan) bagi nomina pengisi fungsi S. Contoh dapat dilihat pada table 2.15.

Tabel 2.15 Contoh Ketiga Penyusunan Klausa Nomina

Temanku	pengacara di sana
S	P

4. Nomina pengisi fungsi P adalah relasi bagi nomina pengisi fungsi S. Contoh dapat dilihat pada table 2.16.

Tabel 2.16 Contoh Keempat Penyusunan Klausa Nomina

Pemuda itu	menantu Pak Camat
S	P

5. Nomina pengisi fungsi S mempunyai ciri atau sifat khas yang disebutkan oleh nomina pengisi fungsi S. Contoh dapat dilihat pada table 2.17.

Tabel 2.17 Contoh Kelima Penyusunan Klausa Nomina

Gajah	binatang berkelompok
S	P

2.1.7.3 Penyusunan Klausa Ajektifa

Klausa ajektifa memiliki fungsi wajib S dan P. Klausa ajektifa dapat disusun dari fungsi S yang berkategori N dan fungsi P yang berkategori A. Klausa ajektifa ini dapat disusun jika:

1. Fungsi yang berkategori ajektifa memiliki komponen makna (+ keadaan fisik). Contoh dapat dilihat pada tabel 2.18.

Tabel 2.18 Contoh Pertama Klausa Ajektifa

Mobil pejabat itu	sangat mewah
S	P

2. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ sifat batin). Contoh dapat dilihat pada tabel 2.19.

Tabel 2.19 Contoh Kedua Klausa Ajektifa

Mereka	riang gembira
S	P

3. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ perasaan batin). Contoh dapat dilihat pada tabel 2.20.

Tabel 2.20 Contoh Ketiga Klausa Ajektifa

Beliau	marah kepada kamu
S	P

2.1.7.4 Penyusunan Klausa Preposisional

Klausa preposisional adalah klausa yang fungsi P nya diisi oleh frase preposisional. Contoh dapat dilihat pada tabel 2.21.

Tabel 2.21 Contoh Pertama Klausa Preposisional

Uangnya	di bank
S	P

Klausa preposisional ini lazim digunakan dalam bahasa ragam lisan dan ragam bahasa nonformal. Dalam ragam formal fungsi P akan diisi oleh sebuah verba dan frase preposisinya menjadi fungsi keterangan. Contoh dapat dilihat pada tabel 2.22.

Tabel 2.22 Contoh Kedua Klausa Preposisional

Uangnya	disimpan	di bank
S	P	Ket

2.1.7.5 Penyusunan Klausa Numeral

Klausa numeral adalah klausa yang fungsi P nya diisi oleh frase numeral. Contoh dapat dilihat pada tabel 2.23.

Tabel 2.23 Contoh Klausa Numeral

Uangnya	Seratus ribu rupiah
S	P

2.1.8 Penyusunan Kalimat

Kalimat adalah satuan sintaksis yang disusun dari konstituen dasar, yang biasanya berupa klausa, dilengkapi dengan konjungsi bila diperlukan, serta disertai dengan intonasi *final*. Intonasi *final* merupakan syarat penting dalam pembentukan sebuah kalimat dapat berupa intonasi deklaratif (ang dalam bahasa ragam tulis diberi tanda titik), intonasi interogatif (yang dalam bahasa ragam tulis diberi tanda tanya), intonasi imperatif (yang dalam bahasa ragam tulis diberi tanda seru), dan intonasi interjektif (yang dalam bahasa ragam tulis diberi tanda seru). Tanpa intonasi *final* ini sebuah klausa tidak akan menjadi sebuah kalimat.

2.1.8.1 Kalimat Sederhana

Kalimat sederhana adalah kalimat yang dibentuk dari sebuah klausa dasar atau klausa sederhana, yakni klausa yang fungsi-fungsi sintaksisnya hanya diisi oleh sebuah kata atau sebuag frase sederhana. Contoh: Anaknya pegawai negeri, Kakek tidur di kamar depan, Adik membaca komik.

Semua kalimat sederhana yang dibentuk dari klausa dasar ini ini adalah kalimat deklaratif positif. Lalu, berdasarkan kategori klausanya dapat disusun kalimat dasar atau kalimat sederhana yang berkategori seperti berikut:

1. Kalimat Verba Monotransitif

   Kalimat verba monotransitif adalah kalimat yang predikatnya berupa verba yang memiliki komponen makna (+ tindakan) dan (+ sasaran). Contoh dapat dilihat pada tabel 2.24.

Tabel 2.24 Contoh Kalimat Verba Monotransitif

Kami	makan	ketoprak	di sana
S	P	O	Ket

2. Kalimat Verba Bitransitif

   Kalimat verba bitransitif adalah kalimat yang predikatnya berupa verba yang memiliki komponen makna (+ tindakan), (+ sasaran), dan (+ pelengkap). Contoh dapat dilihat pada tabel 2.25.

Tabel 2.25 Contoh Kalimat Verba Bitransitif

Kakek	membacakan	adik	cerita lucu
S	P	O	pel

3. Kalimat Verba Intransitif

   Kalimat verba intransitive kalimat yang predikatnya berupa verba yang memiliki komponen makna (+ tindakan) dan (- sasaran). Contoh dapat dilihat pada tabel 2.26.

Tabel 2.26 Contoh Kalimat Verba Intransitive

Kami	berjalan	ke stasiun
S	P	O

4. Kalimat Nomina

   Kalimat nomina adalah kalimat yang predikatnya berkategori nomina, dibentuk dari sebuah klausa nomina dan intonasi *final*. Contoh dapat dilihat pada tabel 2.27.

Tabel 2.27 Contoh Kalimat Nomina

Kera itu	binatang primata
S	P

5. Kalimat Ajektifa

   Kalimat ajektifa adalah kalimat yang predikatnya berkategori ajektifa, dibentuk dari sebuah klausa ajektifa dan intonasi *final*. Contoh dapat dilihat pada tabel 2.28.

Tabel 2.28 Contoh Kalimat jektifa

Jalannya	licin
S	P

6. Kalimat Preposisional

   Kalimat preposisional adalah kamlimat yang predikatnya berupa frase preposisional, atau dibentuk dari sebuah klausa preposisional dan intonasi *final.* Contoh dapat dilihat pada tabel 2.29.

Tabel 2.29 Contoh Kalimat Preposisional

Guru kami	dari Medan
S	P

7. Kalimat Numeral

   Kalimat numeral adalah kalimat yang predikatnya berupa frase numeral, dibentuk dari sebuah klausa numeral dan intonasi *final*. Contoh dapat dilihat pada tabel 2.30.

Tabel 2.30 Contoh Kalimat Numeral

Gaji beliau	lima juta	sebulan
S	P	Ket

2.1.8.2 Kalimat Luas

Dalam praktik berbahasa yang sebenarnya seringkali tidak cukup hanya dengan menggunakan kalimat dasar atau kalimat sederhana. Sebuah kalimat biasanya terangkum informasi lebih banyak di dalamnya dan disebut dengan kalimat luas. Beberapa cara dalam menyusun kalimat luas diantaranya:

1. Disusun dengan cara memberi fungsi keterangan lebih dari sebuah pada sebuah kalimat. Contoh dapat dilihat pada tabel 2.31.

Tabel 2.31 Contoh Pertama Kalimat Luas

Dengan tekun	adik	mengerjakan	PR	semalaman	di kamar tidur ayah
Ket. Cara	S	P	O	Ket. Waktu	Ket. Tempat

2. Disusun dengan cara memberi keterangan tambahan pada fungsi subjek, fungsi objek, atau fungsi lainnya pada kalimat tersebut. Contoh dapat dilihat pada tabel 2.32.

Tabel 2.32 Contoh Kedua Kalimat Luas

Suryani	seorang mahasiswi sebuah unviersitas swasta	di Jakarta	diduga tewas	akbiat kompor gas yang digunakannya meledak
S	Ket. Tambahan pada S	Ket. Tempat	P	Ket. Sebab

3. Disusun dengan cara memberi keterangan aposisi pada fungsi subjek, objek, atau fungsi lainnya pada kalimat tersebut. Contoh dapat dilihat pada tabel 2.33.

Tabel 2.33 Contoh Ketiga Kalimat Luas

Fauzi Bowo	gubernur DKI Jakarta periode 2007-2011	berjanji	akan mengatasi bencana banjir
S	Ket. Aposisi	P	Pelengkap

4. Disusun dengan cara menyisipkan sebuah klausa pada klausa lain. Klausa yang disisipkan disebut klausa sisipan, dan klausa yang tersisipi disebut klausa utama. Penyisipan dilakukan dengan bantuan konjungsi ‘yang’ atau konjungsi lain. Penyisipan bisa dilakukan jika pada klausa sisipan dan klausa utama terdapat maujud yang sama. Contoh dapat dilihat pada tabel 2.34.

Tabel 2.34 Contoh Keempat Kalimat Luas

Kalimat	Wanita yang sedang antre tiket itu bukan ibu saya
Klausa utama	Wanita itu bukan ibu saya
Klausa sisipan	Wanita itu sedang antre tiket

2.2 Teori Bahasa Automata

Teori bahasa dan automata merupakan bagian dari teori komputasi pada ilmu komputer sedangkan tata bahasa (*grammar*) didefinisikan sebagai kumpulan dari himpunan-himpunan variabel, simbol-simbol terminal, simbol awal yang dibatasi oleh aturan-aturan produksi. Pada tahun 1959 seorang ahli bernama Noam

Chomsky melakukan penggolongan tingkatan bahasa menjadi empat (*Hirarky Chomsky*). Penggolongan tersebut dapat dilihat pada tabel 2.35.

Tabel 2.35 *Hirarky Chomsky* [6]

Bahasa	Mesin otomata	Batasan Aturan Produksi
Regular / Tipe 3	*Finite State Automata* (FSA) meliputi *Deterministic Finite Automata* (DFA) dan *Nondeterministic Finite Automata* (NFA)	α adalah sebuah simbol variable. β maksimal memiliki sebuah simbol variable yang bila ada terletak di posisi paling kanan.
Bebas Konteks / *Context Free* / Tipe 2	*Push Down Automata* (PDA)	α berupa sebuah simbol variabel
Context Sensitive / Tipe 1	*Linear Bounded Automata*	$\lvert \alpha \rvert \leq \lvert \beta \rvert$
Unrestricted / *Phase Structure* / *Natural Language* / Tipe 0	Mesin Turing	Tidak ada batasan

Penggolongan diatas berdasarkan pembatasan yang dilakukan pada aturan produksinya. Aturan produksi merupakan pusat dari tata bahasa, yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu *string* ke bentuk lainnya. Melalui aturan produksi tersebut didefinisikan suatu bahasa yang berhubungan dengan tata bahasa tersebut. Semua aturan produksi dinyatakan dalam bentuk "$\alpha \rightarrow \beta$" (dibaca: α menghasilkan β atau α menurunkan β). Dimana α menyatakan simbol-simbol pada ruas kiri aturan produksi dan β menyatakan menyatakan simbol-simbol pada ruas kanan aturan produksi (disebut juga sebagai hasil produksi). Simbol-simbol tersebut dapat berupa simbol terminal atau variabel. Simbol variabel masih bisa diturunkan sedangkan simbol terminal sudah tidak dapat diturunkan lagi. Simbol variabel dinyatakan dalam huruf besar seda ngkan simbol terminal dinyatakan dengan huruf kecil [7].

2.2.1 Tata Bahasa Bebas Konteks

Bahasa bebas konteks memiliki notasi rekursif alamiah yang disebut "tata bahasa bebas konteks". Tata bahasa bebas konteks memiliki peran penting dalam teknologi kompiler sejak tahun 1960. Tata bahasa bebas konteks atau *context free grammar* adalah suatu cara yang menunjukkan bagaimana menghasilkan untai-

untai dalam sebuah bahasa. Suatu CFG terdiri dari himpunan simbol terminal dan simbol variabel. Aturan produksinya dapat dilihat pada aturan produksi 2.1.

$$\alpha \rightarrow \beta \quad \ldots(2.1)$$

Contoh aturan produksi yang termasuk CFG dapat dilihat pada aturan produksi 2.2.

$$S \rightarrow CDeFg$$
$$D \rightarrow BcDe \quad \ldots(2.2)$$

Dimana S merupakan variabel awal dan CDeFg merupakan hasil produksi dari variabel S begitu juga dengan variabel D yang merupakan turunan dari variabel S. Saat menurunkan suatu *string*, simbol-simbol variabel akan mewakili bagian-bagian yang belum terturunkan dari *string* tersebut. Pada CFG letak bagian yang belum terturunkan bisa dimana saja. Saat penurunan itu telah lengkap, semua bagian yang belum terturunkan telah diganti oleh *string-string* (yang mungkin saja kosong) dari himpunan simbol terminal. Tata bahasa bebas konteks menjadi dasar dalam pembentukan suatu proses analisis sintaksis. Bahasa bebas konteks menjadi dasar dalam pembentukan suatu *parser*/proses analisis sintaksis.

2.2.2 Penyederhanaan Tata Bahasa Bebas Konteks

Penyederhanaan tata bahasa bebas konteks bertujuan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tidak perlu atau aturan produksi yang tidak berarti. Suatu tata bahasa bebas konteks dapat disederhanakan dengan melakukan:

1. Penghilangan produksi *useless* (tidak berguna)

   Produksi *useless* merupakan aturan yang memuat simbol variabel yang tidak memiliki penurunan yang akan menghasilkan terminal-terminal seluruhnya, produksi *useless* tidak berguna karena bila diturunkan tidak akan pernah selesai.

   Contoh dapat dilihat pada aturan produksi 2.3.

$$S \rightarrow aSa \mid Abd \mid Bde$$
$$A \rightarrow Ada$$
$$B \rightarrow BBB \mid a \quad \ldots(2.3)$$

Berdasarkan contoh di atas dapat dilihat bahwa simbol variabel A tidak memiliki penurunan yang menuju termina, sehingga bisa dihilangkan. Konsekuensi aturan produksi S → Abd tidak memiliki penurunan. Hasil penyederhaan dapat dilihat pada aturan produksi 2.4.

S → aSa | Bde

B → BBB | a ...(2.4)

2. Penghilangan produksi unit

Produksi unit adalah produksi di mana ruas kiri dan kanan aturan produksi hanya berupa satu simbol variabel, misalkan A → B, C → D. Keberadaan produksi unit membuat tata bahasa memiliki kerumitan yang tidak perlu atau menambah panjang penurunan. Penyederhanaan ini dilakukan dengan melakukan penggantian aturan produksi unit.

Contoh aturan produksi dapat dilihat pada aturan produksi 2.5.

S → Sb

S → C

C → D

C → ef

D → dd …(2.5)

Penggantian dilakukan berurutan mulai dari aturan produksi yang paling dekat menuju terminal-terminal.

C → D => C → dd

S → C => S → dd | ef …(2.6)

Hasil penyederhaan dapat dilihat pada aturan produksi 2.7.

S → Sb | dd | ef

C → dd | ef

D → dd ...(2.7)

3. Penghilang produksi ε

Produksi ε atau produksi kosong dapat dihilangkan, karena produksi ini tidak menghasilkan apa-apa. Contoh aturan produksi dapat dilihat pada aturan produksi 2.8.

S → bcAd

$$A \rightarrow \varepsilon \quad \ldots(2.8)$$

A nullable serta A → ε merupakan satu-satunya produksi dari A maka variabel A bisa ditiadakan. Maka hasil penyederhanaan dapat dilihat pada aturan produksi 2.9.

$$S \rightarrow bcd \quad ...(2.9)$$

2.2.3 *Chomsky Normal Form* (CNF)

CNF merupakan salah satu bentuk normal yang sangat berguna untuk CFG yang telah mengalami penyederhanaan. Aturan produksi dalam bentuk CNF ruas kanannya tepat berupa sebuah terminal atau dua variabel. Dalam CNF, ruas kanan hanya boleh berupa sebuah simbol terminal atau dua buah simbol variable. Jika terdapat lebih dari satu simbol terminal maka harus dilakukan penggantian dan juga jika terdapat lebih dari dua buah simbol variable maka harus dilakukan perubahan.

Contoh aturan produksi dapat dilihat pada 2.10.

$$A \rightarrow BC$$
$$A \rightarrow b$$
$$B \rightarrow a$$
$$C \rightarrow BA \mid d \quad ...(2.10)$$

Langkah-langkah pembentukan bentuk CNF secara umum sebagai berikut:

1. Biarkan aturan-aturan produksi yang sudah dalam bentuk CNF.
2. Lakukan penggantian-penggantian terhadap aturan-aturan produksi yang ruas kanannya mengandung lebih dari satu simbol terminal.
3. Lakukan penggantian-penggantian terhadap aturan-aturan produksi yang ruas kanannya mengandung lebih dari dua simbol variabel.
4. Dalam melakukan penggantian ini dimungkinkan terciptanya suatu simbol-simbol variable baru dan aturan-aturan produksi baru.
5. Hasil akhir adalah gabungan dari aturan-aturan produksi yang sudah dalam bentuk CNF, aturan-aturan produksi yang telah dilakukan penggantian, dan aturan-aturan produksi yang baru.

Contoh CFG yang sudah disederhanakan dapat dilihat pada aturan produksi 2.11.

$$S \rightarrow aB \mid SS \mid c$$

$$B \rightarrow BBB \mid cd \mid a \qquad \ldots(2.11)$$

Pada contoh CFG diatas, aturan produksi yang sudah dalam bentuk CNF adalah $S \rightarrow SS$, $S \rightarrow c$, $B \rightarrow a$. Selebihnya harus dilakukakan penggantian aturan produksi. Hasil penggantian dapat dilihat pada aturan produksi 2.12.

$$\begin{array}{lll} S \rightarrow aB & \text{menjadi} & S \rightarrow V_1B \\ B \rightarrow BBB & \text{menjadi} & B \rightarrow V_2B \\ B \rightarrow cd & \text{menjadi} & B \rightarrow V_3V_4 \end{array} \qquad \ldots(2.12)$$

Aturan produksi baru dapat dilihat pada aturan 2.13

$$\begin{array}{l} V_1 \rightarrow a \\ V_2 \rightarrow BB \\ V_3 \rightarrow c \\ V_4 \rightarrow d \end{array} \qquad \ldots(2.13)$$

Hasilnya dapat dilihat pada aturan produksi 2.14.

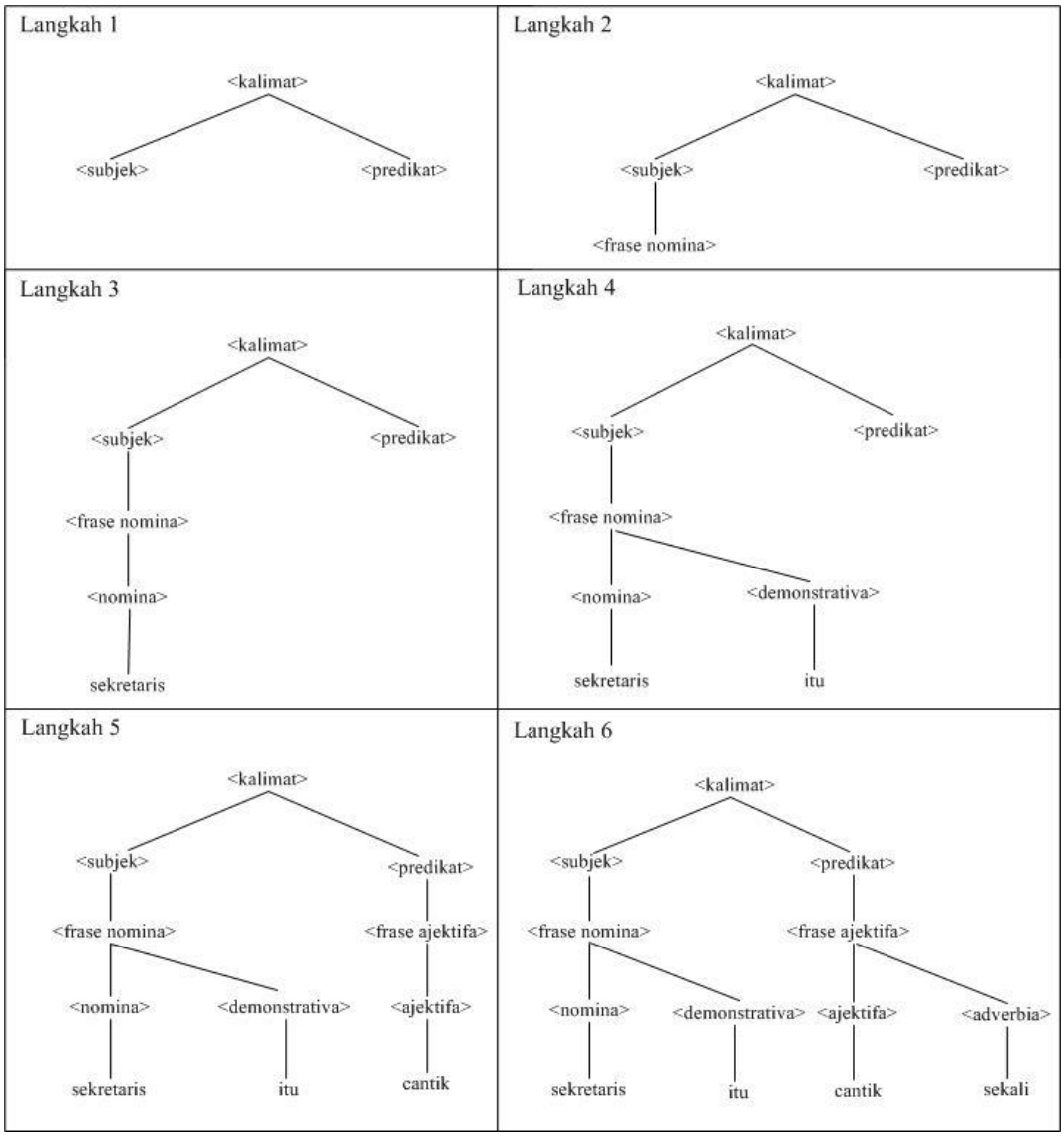
$$\begin{array}{l} S \rightarrow V_1\,B \\ S \rightarrow SS \\ S \rightarrow c \\ B \rightarrow V_2\,B \\ B \rightarrow V_3\,V_4 \\ B \rightarrow a \\ V_1 \rightarrow a \\ V_2 \rightarrow BB \\ V_3 \rightarrow c \\ V_4 \rightarrow d \end{array} \qquad \ldots(2.14)$$

2.2.4 Teknik-Teknik *Parsing*

Parsing merupakan proses memecah *input* teks menjadi token untuk menentukan pembentukan struktur kalimat. Proses *parsing* bertujuan untuk menentukan apakah *input* tersebut dapat diturunkan dari *state* awal tata bahasa yang membentuk pohon penurunan. Ada 2 teknik parsing yang dapat digunakan diantaranya:

2.2.4.1 *Top-Down Parsing*

Top-Down Parsing bekerja dengan cara menguraikan sebuah kalimat mulai dari *constituent* yang terbesar sampai menjadi *constituent* yang terkecil. *Constituent* yaitu unsur-unsur pembentuk kalimat yang dapat berdiri sendiri seperti *noun* dan *verb*. Hal ini dilakukan terus-menerus sampai semua komponen yang dihasilkan ialah *constituent* yang terkecil dalam kalimat. Cara kerja *top-down parsing* dapat dilihat pada gambar 2.1. Kata yang digunakan sebagai contoh adalah "Sekretaris itu cantik sekali".

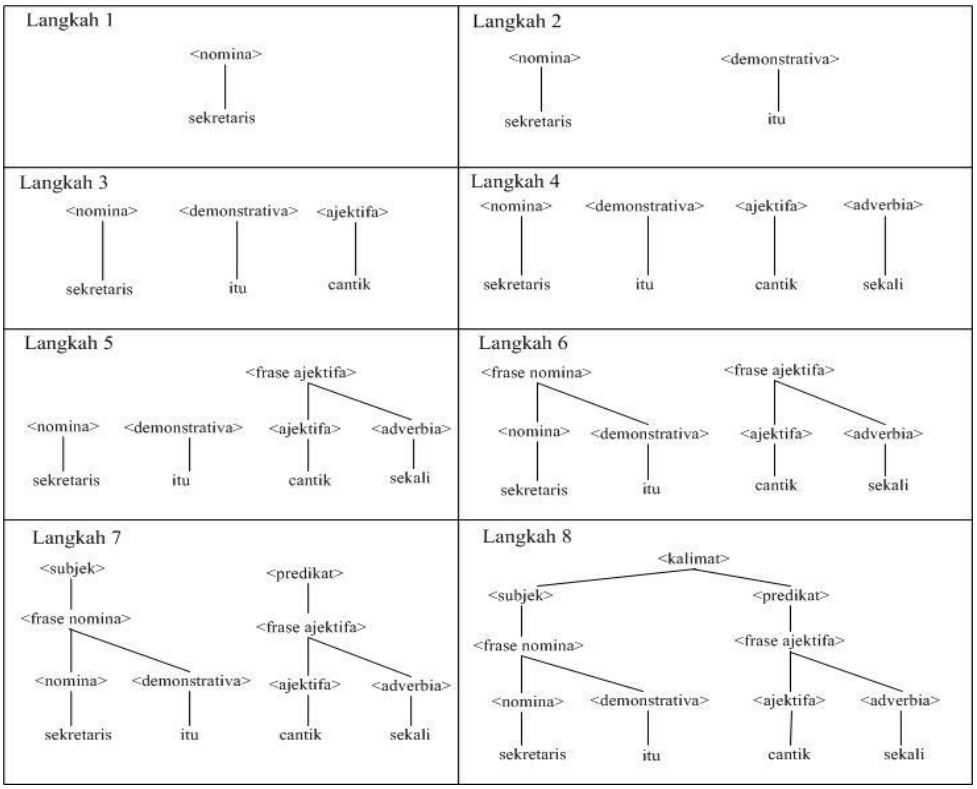


Gambar 2.1 Cara kerja *top-down parsing* [7]

Langkah-langkah pada proses *parsing* ini dilakukan dengan menelusuri setiap cabang yang akan mengarah ke terminal. Pada proses *parsing* ini terdapat metode *backtracking*. Apabila tidak menemukan terminal, cabang sebelumnya melakukan penelusuran dengan mengganti cabang yang dicari. Sebagai contoh pada penelusuran langkah ke-4 sebelum mencapai *tag* <demonstrative> dilakukan penelusuran *tag* <nomina>, <verba>, <ajektiva>, <adverbia>, dan <numeralia>. Penelusuran tersebut disesuaikan dengan penyusunan frase nomina yang terdapat pada sub bab 2.1.6.1.

2.2.4.2 *Bottom-Up Parsing*

Bottom-Up Parsing bekerja dengan cara mengambil satu demi satu kata dari kalimat yang diberikan, untuk dirangkaikan menjadi *constituent* yang lebih besar. Hal ini dilakukan secara terus-menerus sampai *constituent* yang terbentuk ialah kalimat. Metode *Bottom-Up Parsing* bekerja dengan cara yang terbalik dari cara kerja *Top-Down Parsing*. Cara kerja *bottom-up parsing* dapat dilihat pada gambar 2.2.



Gambar 2.2 Cara kerja *bottom-up parsing* [7]

Langkah-langkah pada proses *parsing bottom-up* dilakukan dengan mengenali terlebih dahulu cabang dari terminal-terminal yang ditelusuri. Pada contoh gambar 2.2 kata “sekretaris” merupakan terminal dari cabang <nomina>, kata “itu” merupakan terminal dari cabang <demonstrativa>, kata “cantik” merupakan terminal dari cabang <ajektiva>, dan kata “sekali” merupakan terminal dari cabang <adverbia>. Pada tahap selanjutnya dilakukan penelusuran *root* dari cabang <ajektiva> dan <adverbia>. Jika ada, penelusuran berlanjut ke cabang yang ada disebelah kirinya. Penelusuran dilakukan hingga dapat mencapai *root* <kalimat>.

2.3 Post Tag

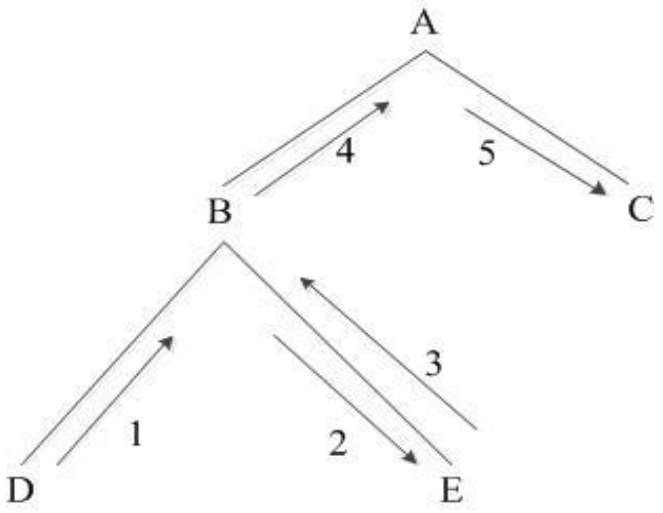
POS (*Part-of-Speech*) Tag merupakan suatu cara pengkategorian kelas kata, seperti kata benda, kata kerja, kata sifat, dan lain-lain. POS Tagger merupakan sebuah aplikasi yang mampu melakukan proses anotasi *part-of-Speech* tag untuk setiap kata di dalam dokumen secara otomatis. POS Tag yang digunakan sebagai bantuan dalam mengenali token/tag setiap kata diambil dari POS Tag Indonesia yang dibuat oleh Arawinda Dinakaramani, Fam Rashel, Andry Luthfi, dan Ruli Manurung [8].

POS Tag akan mengenali kata mana yang termasuk kata benda, kata kerja, kata sifat, kata keterangan, kata depan, kata sambung, kata ganti benda, dan kata bilangan. POS Tag ini menggunakan pendekatan *Rule-Based* berdasarkan aturan tata bahasa Indonesia. Pertama-tama, POS Tag akan melakukan tokenisasi terhadap teks menggunakan kamus bahasa Indonesia. Selanjutnya kata-kata yang termasuk ke dalam jenis *closed-class word* diproses. Lalu setiap kata yang ambigu diproses menggunakan aturan-aturan yang sudah didefinisikan untuk menemukan kelas kata yang tepat.

2.4 Algoritma Left Corner Parsing

Algoritma *left corner parsing* pertama kali dikemukakan oleh Rosen krantz dan Lewis II pada tahun 1970. Algoritma *left corner parsing* merupakan gabungan dari *top-down parsing* dan *bottom-up parsing* yaitu merupakan strategi yang menggunakan data secara *bottom-up parsing* dan prediksi dari *top-down parsing.* Cara kerja algoritma ini yaitu dengan mula-mula menerima sebuah kata,

menentukan jenis *constituent* apa yang dimulai dengan jenis kata tersebut. Kemudian akan dilakukan proses *parsing* terhadap sisa dari *constituent* secara *top-down.* Dengan demikian proses *parsing* pada algoritma *left corner parsing* dimulai secara *bottom-up* dan diakhiri secara *top-down* [9]. Proses cara kerja algoritma *left corner* dapat dilihat pada gambar 2.3.



Gambar 2.3 Cara kerja *left corner parsing*

Algoritma *Left Corner Parsing* dalam cara kerjanya menggunakan tiga *stacks* yaitu:

1. Sentence, merupakan *stack* dari kalimat yang akan diperiksa.
2. Categories, membaca ruas sebelah kanan dari aturan produksi yang akan dicari.
3. Constituent, hasil dari pengelompokkan kelas kata atau ruas kiri dari aturan produksi.

Dalam proses pemeriksaan algoritma *left corner parsing* operasi yang digunakan adalah *reduce*, *move*, dan *remove.* Operasi *reduce* digunakan untuk mengetahui ruas kiri aturan produksi dari suatu terminal atau variabel. Untuk operasi *move* digunakan ketika ruas paling kiri dari *categories* terdapat simbol $, dimana simbol tersebut merupakan pembatas. Operasi *move* yaitu perpindahan dari *stack constituent* ke *stack categories.* Sedangkan operasi *remove* yaitu operasi yang digunakan untuk menghapus jika terdapat variabel atau terminal yang sama pada *stack sentence* dan *categories* (*stack* di pop).

Sebagai contoh terdapat suatu aturan produksi sebagai berikut:

S → ASB | d

A → a

B → b

Langkah-langkah dari cara kerja algoritma *left corner parsing* dapat dilihat pada tabel 2.36 sebagai berikut:

Tabel 2.36 Langkah-langkah *Left Corner Parsing*

Id	Sentence	Categories	Constituent	Operation
1	adb	S	ε	
2	db	$S	A	reduce A →a
3	Adb	S	ε	move
4	db	SB$S	S	reduce S →ASB
5	b	SBS	SS	reduce S →d
6	Sb	SB$S	S	move
7	b	B$S	S	remove
8	ε	BS	BS	reduce B →b
9	B	B$S	S	move
10	ε	S	S	remove
11	S	S	ε	move
12	ε	ε	ε	remove

Penjelasan langkah-langkah algoritma *left corner parsing* dari tabel 2.36 yaitu sebagai berikut:

1. Langkah pertama yaitu dengan mengisi semua *stack* sesuai dengan string yang akan diperiksa.
2. Kemudian langkah kedua yaitu melihat pada *stack categories* terdapat simbol $ atau tidak, jika tidak terdapat simbol $ maka cek pada *stack sentence* dan *categories* terdapat variabel atau terminal yang sama. Ketika tidak ditemukan variabel atau terminal yang sama pada kedua ruas kirinya maka lakukan operasi *reduce* pada ruas paling kiri dari *stack sentence.* Selanjutnya sisa variabel dan terminal dari ketiga *stack* setelah dilakukannya *reduce* diturunkan.

3. Langkah ketiga dan seterusnya masih sama seperti langkah kedua, namun ketika terdapat simbol $ pada *stack categories* maka operasi yang digunakan yaitu *move*.
4. Untuk langkah selanjutnya ketika saat melihat *stack sentence* dan *categories* terdapat variabel atau terminal yang sama dari masing-masing ruas kirinya, maka lakukan operasi *remove.*
5. Jika pada baris hasil akhir dari ketiga *stack* menghasilkan ε maka *string* tersebut dapat dikenali oleh algoritma *left corner parsing.*

2.5 Algoritma *Cocke-Younger-Kasami* (CYK)

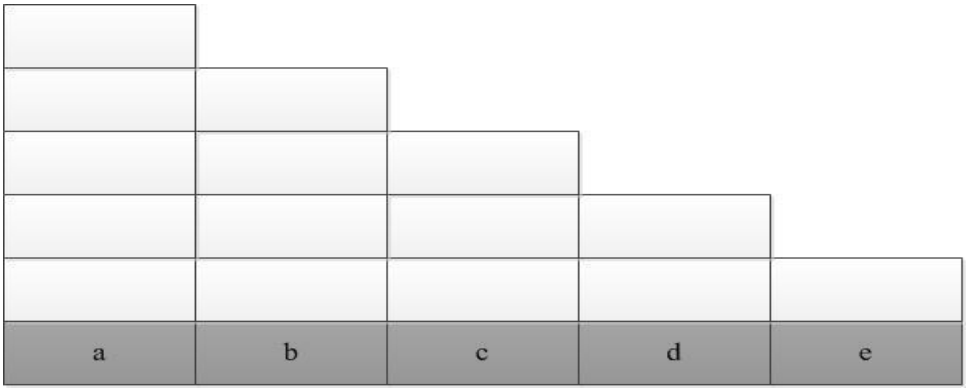
Algoritma CYK merupakan algoritma *parsing* dan keanggotaan untuk tata bahasa bebas konteks. Algoritma CYK diciptakan oleh J. Cocke, DH. Younger, dan T. Kasami. Syarat untuk penggunaan algoritma CYK adalah tata bahasa harus berada dalam bentuk CNF. Tujuan dari algoritma ini adalah untuk menunjukkan apakah suatu *string* dapat diperoleh dari suatu tata bahasa [7].

Proses *parsing* untai dengan algoritma CYK menggunakan struktur data sebuah *array* dua dimensi dengan jumlah baris dan kolom n x n. Nilai n ditentukan dari jumlah kata dalam suatu kalimat. Proses *parsing* dari algoritma CYK memanfaatkan hasil *parsing* sebelumnya untuk menentukan apakah proses yang sedang berlangsung dapat diterima atau tidak.

Sebagai contoh terdapat suatu aturan produksi 2.15 dengan variabel awal s.

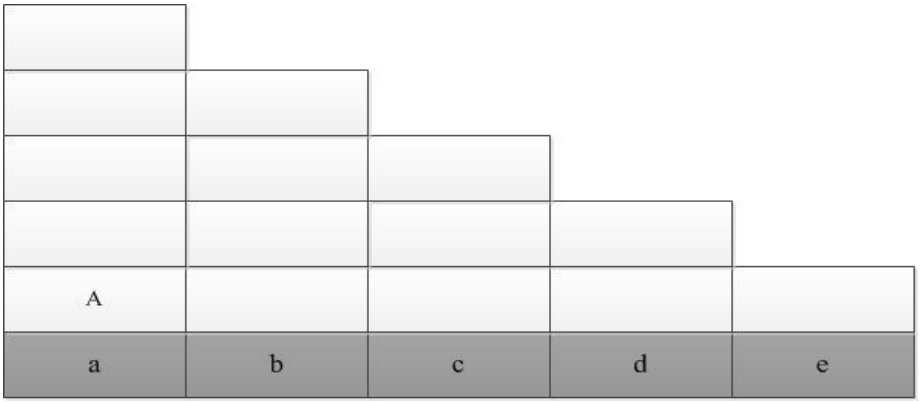
$$
\begin{aligned}
&S \rightarrow AB \mid EC \\
&F \rightarrow CA \\
&G \rightarrow CD \\
&H \rightarrow EF \\
&A \rightarrow a \\
&B \rightarrow b \\
&C \rightarrow c \\
&D \rightarrow d \\
&E \rightarrow e
\end{aligned}
\qquad \ldots(2.15)
$$

String yang akan diperiksa misalnya “abcde”. Langkah pertama yang dilakukan adalah dengan membuat tabel dua dimensi seperti gambar 2.4.



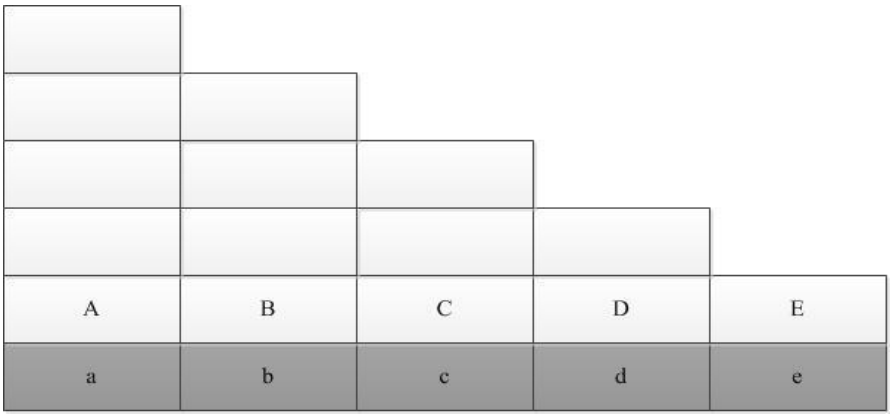
Gambar 2.4 Pembentukan Tabel *Parsing* Algoritma CYK

Setiap kata yang ada pada suatu *string* ditempatkan pada masing-masing blok seperti gambar 2.4. Setiap blok yang kosong diisi berdasarkan variabel yang sesuai dengan aturan produksi dari CNF yang sudah dibentuk sebelumnya. Sebagai contoh pada baris pertama dan kolom pertama diisi dengan variabel yang dapat menghasilkan huruf a, yaitu variabel A. Hasilnya dapat dilihat pada gambar 2.5.



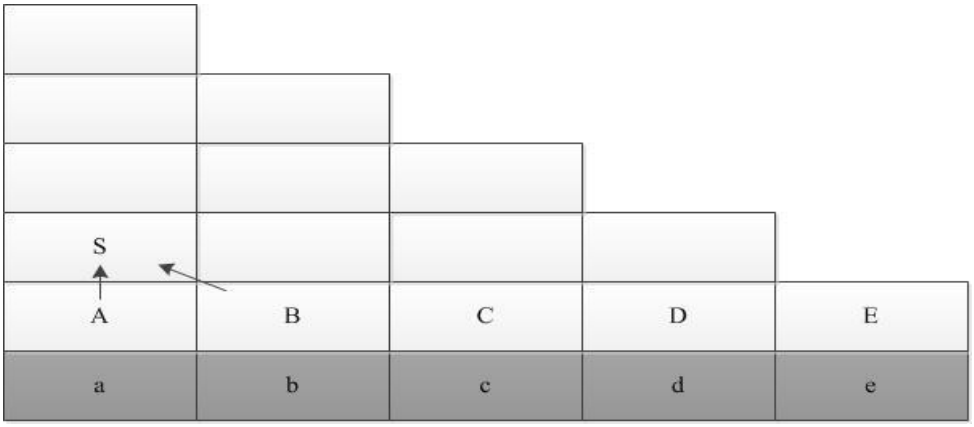
Gambar 2.5 Pengisian Tabel dengan Variabel A

Blok selanjutnya diisi dengan langkah yang sama, sehingga dapat dilihat baris pertama isiannya seperti gambar 2.6.



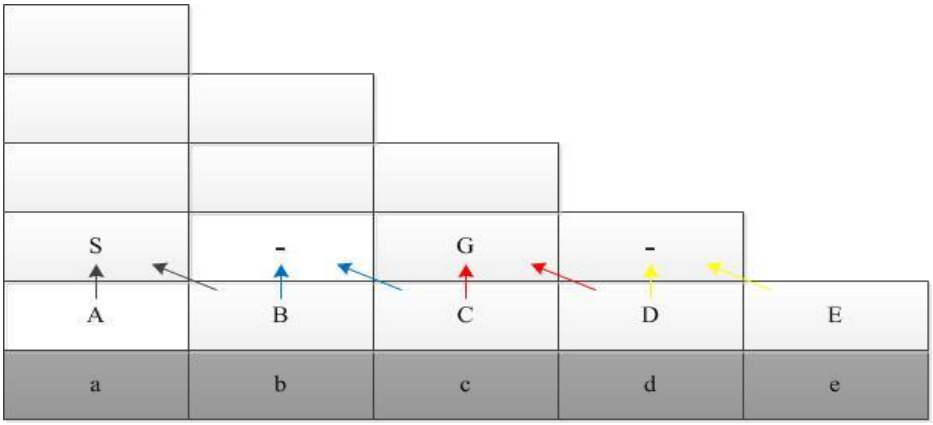
Gambar 2.6 Pengisian Tabel Baris Pertama

Pada baris selanjutnya cara pengisian dilakukan dengan melihat variabel pada dua blok, kemudian dilihat variabel mana yang menghasilkan kedua variabel tersebut. Sebagai contoh baris kedua kolom pertama diisi dengan melihat variabel pada blok pertama dan blok dua, yaitu variabel A dan B yang termasuk ke dalam variabel S, sehingga cara pengisiannya terlihat seperti pada gambar 2.7.



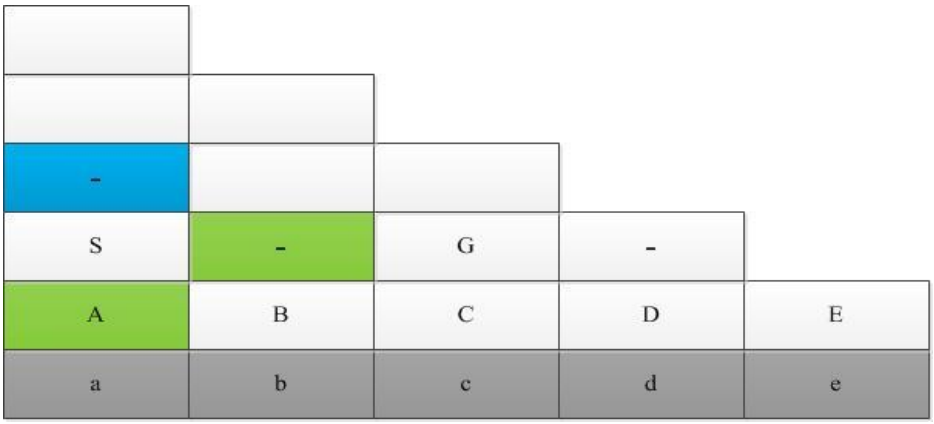
Gambar 2.7 Pengisian Tabel dengan Variabel B

Blok selanjutnya diisi dengan langkah yang sama sesuai dengan blok yang akan diisi. Apabila pengisian dilakukan pada blok kedua, variabel diambil dari kolom kedua dan ketiga. Untuk lebih jelasnya dapat dilihat pada gambar 2.8.



Gambar 2.8 Pengisian Tabel Baris Kedua

Pengisian pada baris ketiga dan selanjutnya hampir sama seperti baris kedua, namun terdapat perbedaan dari cara pengecekkan yang dilakukan. Pengisian baris ketiga dilakukan dari baris pertama hingga sebelum baris ketiga (n-1). Cara pengecekkan dapat dilihat seperti gambar 2.9.



Gambar 2.9 Cara Pengecekan Langkah Pertama

Warna biru menunjukkan blok yang akan diisi, sedangkan warna hijau merupakan blok yang dicek variabelnya. Pada kedua gambar diatas dapat dilihat untuk baris ketiga cara pengecekkan dilakukan secara diagonal dan vertikal. Cara pengisian selanjutnya sama seperti gambar 2.9. Hasil akhir dari pengecekan *string* diatas dapat dilihat pada gambar 2.10.

-				
-	-			
-	-	-		
S	-	G	-	
A	B	C	D	E
a	b	c	d	e

Gambar 2.10 Hasil *Parsing String* dengan Algoritma CYK

Dapat dilihat dari gambar diatas *string* "a b c d e" termasuk ke dalam aturan produksi yang sudah dibuat, artinya *string* tersebut dapat dikatakan sebagai kalimat dalam contoh ini. Jika baris dan kolom terakhir menghasilkan tanda "-", *string* tersebut belum dapat dikenali oleh algoritma CYK.

2.6 Pemrograman Berorientasi Objek

Metodologi beorientasi objek adalah suatu startegi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek [5].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk massalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan

   Dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi

   Sifat pewaris dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak

   Pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

   Berikut ini adalah beberapa konsep dasar metodologi berorientasi objek:

1. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek. Kelas secara fisik adalah berkas atau f*ile* yang berisi kode program, di mana kode program merupakan semua hal yang terkait dengan nama kelas.

2. Objek (*objek*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat dari segi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi.

3. Metode (*method*)

   Metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode dapat berasal dari *event*, aktivitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata. Contoh metode misalnya *read, write, move, copy.*

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapt berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut sebaiknya bersifat privat untuk menjaga konsep enkapsulasi.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek yang lain mengetahui cara kerjanya.

7. Pewaris *(inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka. Antarmuka atau *interface* biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas, mengakses antarmukanya.

2.7 *Unified Modelling language* (UML)

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson [5].

2.7.1 Use Case Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Diagram ini juga mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use case*, digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu:

1. Aktor

   Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat

itu sendiri. Simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case*

   Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.7.2 Activity Diagram

Activity diagram merupakan diagram menggambarkan *workflow* (alur kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

2.7.3 Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Diagram kelas dibuat supaya pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang akan menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (model)

   Kelas yang digunakan untuk memegang atau menghubungkan data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas.

2.7.4 Sequence Diagram

Sequence diagram merupakan diagram yang menggambarkan kelakuan perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan (*message*) yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

2.8 Pengujian

Pengujian adalah satu set aktivitas yang direncanakan dan sistematis untuk mengevaluasi kebenaran dari suatu perangkat lunak atau sebuah algoritma. Aktivitas pengujian terdiri dari sekumpulan langkah dimana dapat menempatkan desain kasus uji yang spesifik dan metode pengujian [6]. Kualitas perangkat lunak harus mampu melakukan beberapa hal diantaranya adalah:

1. Bertahan hidup di dunia bisnis perangkat lunak.
2. Bersaing dengan perangkat lunak lainnya.
3. Global marketing.
4. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran atau kegagalan produksi.
5. Mempertahankan pelanggan dan meningkatkan keuntungan.

Perangkat lunak terkadang mengandung kesalahan (*error*) pada proses-proses tertentu. Kesalahan-kesalahan (*error*) ini sering disebut *bug*. Untuk menghindari banyaknya *bug* maka diperlukan pengujian perangkat lunak sebelum sampai pada pelanggan (*end user*). *Bug* merupakan suatu hal yang biasa sehingga yang perlu dilakukan adalah meminimalisir *bug* dengan melaukan pengujian.

Secara umum pengujian perangkat lunak adalah sebagai berikut:

1. Pengujian dilakukan dari level komponen hingga integrasi antar komponen menjadi sebuah sistem.
2. Teknik pengujian disesuaikan dengan berbagai unit dalam waktu yang berbeda-beda bergantung pada pengujian bagian yang dibutuhkan.
3. Pengujian dilakukan oleh pengembang perangkat lunak atau tim uji yang tidak terkait dengan pengembang perangkat lunak (*independent test group*).

4. Pengujian dan penirkutuan (*debugging*) merupakan aktifitas yang berbeda namun harus diakomodasi dengan berbagai strategi pengujian. Pengujian lebih focus untuk mencari adanya kesalahan (*error*) baik dari sudut pandang orang secara umum atau dari sudut pandang pengembang tanpa harus menemukan lokasi kesalahan pada kode program. Penirkutuan (*debugging*) adalah proses mencari lokasi kesalahan (*error*) pada kode program sehingga dapat segera diperbaiki oleh pembuat program.

2.8.1 Pengujian *Black Box*

Pengujian yang dilakukan untuk menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan kebutuhan. Pengujian fungsional perangkat lunak yang dilakukan pada sistem, lengkap terpadu dan digunakan untuk mengevaluasi kepatuhan sistem dengan persyaratan yang ditentukan. Pada pengujian fungsional terdapat beberapa jenis pengujian yaitu:

1. Pengujian Unit

   Pengujian unit fokus pada usaha verifikasi pada unit yang terkecil pada desain perangkat lunak. Setiap unit perangkat lunak diuji agar dapat diperiksa apakah aliran masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan. Pengujian unit biasanya dilakukan saat kode program dibuat. Karena dalam sebuah perangkat lunak banyak memiliki unit-unit kecil maka untuk menguji nuit-unit kecil tersebut dibuat program kecil untuk menguji unit-unit perangkat lunak. Unit disini secara fisik dapat berupa prosedur atau fungsi, sekumpulan prosedur atau fungsi yang ada dalam berkas (*file*) jika dalam pemrograman terstruktur atau kelas namun bisa juga kumpulan kelas dalam satu *package* dalam pemrograman berorientasi objek.

2. Pengujian Integrasi

   Pengujian integrasi adalah sebuah teknis yang sistematik untuk mengonstruksi struktur program seiring dengan menggabungkan fungsi program dengan antarmukanya. Pengujian terintegrasi bertujuan untuk mempergunakan komponen unit program yang sudah diuji dan membangun struktur seperti yang

telah didesain sebelumnya. Pada pengujian ini dilakukan secara langsung pada akhir pengembangan perangkat lunak ("*big bang*"). Pendekatan *big bang* digunakan sebuah sistem sistem diuji secara kesatuan sehingga sering ketika terjadi kesalahan (*error*) akan menemui kesulitan untuk menemukan dimana letak kesalahan (*error*) yang terjadi.

3. Pengujian *Regression Integration*

   Pengujian regresi adalah eksekusi dari beberapa subset pengujian terhubung atau saling terkait untuk menjamin bahwa model yang baru masuk pengujian tidak mengubah funsionalitas yang sudah diuji sebelumnya. Pengujian regresi dapat dilakukan secara manual dengan mengeksekusi sebuah subset untuk semua kasus uji dari subset itu atau bisa juga menggunakan perangkat (*tool*). Pengujian regresi lebih seusai menggunakan tiga kelompok kasus pengujian sebagai berikut:

   1. Kelas uji yang berisi contoh kasus pengujian yang dapat menguji semua fungsi perangkat lunak.
   2. Kelas kasus uji yang berisi kasus tambahan yang fokus pada fungsi perangkat lunak yang akan terpengaruh jika ada tambahan modul baru untuk diuji.
   3. Kelas uji yang berisi kasus yang fokus pada komponen atau modul baru atau yang mengalami perubahan.

4. Pengujian *Smoke Integration*

   Pengujian asap (*smoke testing*) adalah sebuah pendekatan pengujian integrasi yang biasa digunakan ketika pengerjaan perangkat lunak cukup singkat dan biasanya untuk komponen atau modul yang ditambahkan pada perangkat lunak. Pengujian ini meliputi hal-hal sebagai berikut:

   1. Mempersiapkan komponen yang telah ditranslasi menjadi kode program kemudian diintegrasikan dengan komponen lain ynag terkait seperti berkas (*file*), modul lain yang digunakan kembali untuk mengimplementasi satu atau lebih fungsi perngkat lunak.

2. Mempersiapkan sekumpulan penguji yang didesain untuk menemukan kesalahan (*error*) yang menjaga perangkat lunak tetap memenuhi fungsinya.
3. Mengimplementasikan sekumpulan kode program, berkas (*file*), pustaka, modul lain yang digunakan kembali dan komponen rekayasa lainnya yang diperlukan dengan kumpulan yang lain da n diuji perhari agar setiap pertambahan komponen perhari dapat diuji, pendekatan yang dilakukan bisa menggunakan *top-down* atau *bottom-up.*

2.8.2 Pengujian Matrik Confusion

Kinerja prediksi suatu sistem tidak bisa bekerja 100% benar. Dalam contoh kasus yang memiliki banyak prediksi, hasil pada data uji sering ditampilkan sebagai matrik confusion dua dimensi dengan baris dan kolom untuk setiap kelas. Setiap elemen matrik menunjukkan jumlah data uji. Pada bagian baris adalah kelas data uji yang sebenarnya dan pada bagian kolom adalah kelas prediksi [10]. Jumlah data dari masing-masing kelas yang diprediksi secara benar adalah (f_{11} + f_{00}), dan data yang diprediksi secara salah adalah (f_{10} + f_{01}). Contoh tabel pengukuran akhir: akurasi dan laju error dapat dilihat pada Tabel 2.37 berikut.

Tabel 2.37 Tabel Matrik Confusion [10]

f_{ij}		Kelas hasil prediksi (*j*)	
		Kelas = 1	Kelas = 0
Kelas asli (*i*)	Kelas = 1	f_{11}	f_{10}
	Kelas = 0	f_{01}	f_{00}

$$\text{Akurasi} = \frac{\text{Jumlah data yang diprediksi secara benar}}{\text{Jumlah data yang dilakukan}} = \frac{f11+ f00}{f11+ f10+ f01+f00}$$

$$\text{Laju error} = \frac{\text{Jumlah data yang diprediksi secara salah}}{\text{Jumlah data yang dilakukan}} = \frac{f10+ f01}{f11+ f10+ f01+f00}$$