BAB 2

PENDAHULUAN

2.1 Sintaksis Bahasa Indonesia

Sintaksis 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 Penyusunan Frase

Frase adalah satuan sintaksis yang tersusun dari dua buah kata atau lebih, yang didalam 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 sebagai inti frase dan ada yang berkedudukan sebagai bawahan, yang disebut tambahan penjelas frase [5]. Frase dibagi menjadi beberapa kelompok:

1. Frase Nomina Koordinatif
2. Frase Nomina Subordinatif
3. Frase Verba Koordinatif
4. Frase Verba Subordinatif
5. Frase Ajektifa Koordinatif
6. Frase Ajektifa Subordinatif
7. Frase Preposisional

2.1.1.1 Penyusunan Frase Nomina

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

1. Penyusunan Frase Nomina Koordinatif

   FNK dapat disusun dari:

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

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

2. Penyusunan Frase Nomina Subordinatif

   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 dari FNS yang berstruktur N + N dapat dilihat pada tabel 2.1.

Tabel 2.1 Contoh frase nomina (N + N)

Makna Gramatikal	Contoh Frase
milik	rumah paman
bagian	awal tahun
asal bahan	cincin emas
asal tempat	jeruk bali
campuran	roti keju
hasil	lukisan Afandi
jenis	mobil sedan
gender	anak laki-laki
seperti	jamur kuping
model	topi koboi
memakai	kereta listrik
peruntukkan	pensil alis
ada di	angkatan darat
tempat	botol kecap

letak	pintu belakang
dilengkapi	kursi roda
sasaran	perluasan kota
pelaku	pemberian kakek
alat	perang mulut

b. FNS yang berstruktur N + V

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

Tabel 2.2 Contoh frase nomina (N + V)

Makna Gramatikal	Contoh Frase
tempat	halaman parkir
kegunaan	pisau cukur
yang di	sambal goreng
yang biasa melakukan	tukang copet

c. FNS yang berstruktur N + A

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

Tabel 2.3 Contoh frase nomina (N + A)

Makna Gramatikal	Contoh Frase
keadaan	mobil rusak
derajat	sekolah dasar
rasa	kecap manis
bentuk	rumah mungil

d. FNS yang berstruktur Adv + N

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

Tabel 2.4 Contoh frase nomina (Adv + N)

Makna Gramatikal	Contoh Frase
ingkar	tanpa usaha
kuantitas	semua penduduk
batas	hanya nasi

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 (tak ada yang lain), Uang saja (bukan benda lain), Dia saja (orang lain tak ada), Garam saja (tanpa bumbu lain), Pensil saja (tidak pakai pena).

f. FNS yang berstruktur Num + N

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

Tabel 2.5 Contoh frase nomina (Num + N)

Makna Gramatikal	Contoh Frase
banyaknya	sepuluh rumah
himpunan	kelima rumah

g. FNS yang berstruktur N + Num

Sejauh ini FNS yang berstruktur N + Num memiliki makna gramatikal ‘tingkat’. Contoh: Juara kedua, Kursi ketiga, Rumah kelima, Simpangan kedua, Anak kelima.

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 pronomina deomnstratifa. Contoh: Anak itu, Topi ini, Mereka itu, Pegawai ini, Universitas itu.

2.1.1.2 Penyusunan Frase Verba

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

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 keduanya dapat disisipkan kata ‘dan’. Contoh: Makan minum, Usap raba, Peluk cium, Makan pakai, Dengar lihat.

2. Penyusunan Frase Verba Subordinatif (FVS)

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

   a. FVS yang berstruktur Adv + V

      Beberapa contoh dari FVS yang berstruktur Adv + V dapat dilihat pada tabel 2.6.

Tabel 2.6 Contoh frase verba (Adv + V)

Makna Gramatikal	Contoh Frase
ingkar	tidak membayar
frekuensi	sering muncul
kuantitas	sedikit bicara
waktu	sudah mandi
keinginan	akan mandi
keselesaian	sedang bertemu
keharusan	wajib hadir
kepastian	pasti hadir
pembatasan	hanya bergurau

b. FVS yang berstruktur V + Adv

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

Tabel 2.7 Contoh frase verba (V + Adv)

Makna Gramatikal	Contoh Frase
berulang	makan lagi
ikut serta	minum juga

c. FVS yang berstruktur V + N

Beberapa contoh dari 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 payung, 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.

2.1.1.3 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)

   Frase ajektifa koordinatif (FAK) dapat disusun dari:

   a. Dua buah kata berkategori ajektifa yang merupakan anggota dari antonim relasional dan memiliki makna gramatikal ‘pilihan’, sehingga di antara

kedua dapat disisipkan kata ‘atau’. Contoh: baik buruk, tua muda, jauh dekat, lama baru.

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, besar tetapi jelek, kecil tetapi mungil, repot tetapi menyenangkan, sedih tapi senang.

2. Penyusunan Frase Ajektifa Subordinatif (FAS)

Frase ajektifa subordinatif disusun dengan struktur A + N, A + A, A + V, Adv + A, dan A + Adv. Aturannya sebagai berikut:

a. FAS yang berstruktur A + N dan memiliki makna gramatikal ‘seperti’ apabila unsur pertama berketegori ajektifa dan memiliki komponen makna (+ warna) dan unsur kedua berkategori nomina dan memiliki komponen makna (+ perbandingan), sehingga di antara kedua unsrunya 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.
   2. Unsur pertama berketegori ajektifa dan memiliki komponen makna (+ warna), sedangkan unsur kedua berkategori ajektifa dan berkomponen makna (+ warna) dan (+ benda). Contoh: kuning kehijau-hijauan.

c. FAS yang berstruktur A + V dan bermakna gramatikal ‘untuk’ dapat disusun aabila 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 takut, tidak berani, tidak malu.
e. FAS yang berstruktur Adv + A dan bermakna gramatikal ‘derajat’ dapat disusun bilsa 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 baik, cukup 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, panas sekali.

2.1.1.4 Penyusunan Frase Preposisional

Frase preposisional adalah frase yang berfungsi sebagai pengisi fungsi keterangan di dalam sebuah klausa. Frase preposisional ini buka frase koordinatif maupun subordinatif, melainkan fase 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. Contoh: di pasar, ke dalam kamar, dari rumah sakit, dengan pensil warna, oleh ibu tiri.

2.1.2 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 disebut 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 [5].

Berdasarkan kategori yang mengisi fungsi P dapat dibedakan menjadi:

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

2.1.2.1 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 kategori:

   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, 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 tabel 2.8.

Tabel 2.8 Contoh klausa verba tindakan bersasaran tak berpelengkap

Kalimat	Pak Lurah	membaca	koran
Sintaksis	S	P	O
Makna	(+ manusia)	(+ manusia)	-
	-	(+ bacaan)	(+ bacaan)

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 tabel 2.9.

Tabel 2.9 Contoh klausa verba tindakan bersasaran berpelengkap

Kalimat	Saya	membukakan	ayah	pintu
Sintaksis	S	P	Pel	O
Makna	(+ 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: Anak-anak itu menari, Anjing mengonggong, Mereka sedang berlibur di Bali.

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 2.10.

Tabel 2.10 Contoh klausa verba kejadian

Kalimat	Kompor gas baru itu	meledak
Sintaksis	S	P
Makna	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 2.11.

Tabel 2.11 Contoh klausa verba keadaan

Kalimat	Kami	malu dengan kejadian semalam
Sintaksis	S	P
Makna	yang mengalami	keadaan

2.1.2.2 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 pengisi fungsi P (generik). Contoh dapat dilihat pada tabel 2.12.

Tabel 2.12 Contoh pertama penyusunan klausa nomina

Kalimat	Anjing itu	binatang
Sintaksis	S	P

2. Nomina yang mengisi fungsi S mempunyai nama pada nomina pengisi fungsi P. Contoh dapat dilihat pada tabel 2.13.

Tabel 2.13 Contoh kedua penyusunan klausa nomina

Kalimat	Petani itu	Pak Ridwan
Sintaksis	S	P

3. Nomina pengisi fungsi P adalah profesi (jabatan, pekerjaan) bagi nomina pengisi fungsi S. Contoh dapat dilihat pada tabel 2.14.

Tabel 2.14 Contoh ketiga penyusunan klausa nomina

Kalimat	Ibunya	dokter gigi di puskesmas itu
Sintaksis	S	P

4. Nomina pengisi fungsi P adalah relasi bagi nomina pengisi fungsi S. Contoh dapat dilihat pada tabel 2.15.

Tabel 2.15 Contoh keempat penyusunan klausa nomina

Kalimat	Mereka itu	murid Pak Rahmat
Sintaksis	S	P

5. Nomina pengisi fungsi S mempunyai ciri atau sifat khas yang disebutkan oleh nomina pengisi fungsi S. Contoh dapat dilihat pada tabel 2.16.

Tabel 2.16 Contoh kelima penyusunan klausa nomina

Kalimat	Kereta api	kendaraan murah
Sintaksis	S	P

2.1.2.3 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 P yang berkategori ajektifa memiliki komponen makna (+ keadaan fisik). Contoh dapat dilihat pada tabel 2.17.

Tabel 2.17 Contoh pertama klausa ajektifa

Kalimat	Gadis itu	tinggi sekali
Sintaksis	S	P

2. Fungsi P yang berkategori ajektifa memiliki komponen makna (+ sifat batin). Contoh dapat dilihat pada tabel 2.18.

Tabel 2.18 Contoh kedua klausa ajektifa

Kalimat	Pemuda desa itu	sangat berani
Sintaksis	S	P

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

Tabel 2.19 Contoh ketiga klausa ajektifa

Kalimat	Saya	khawatir dengan keadaanmu
Sintaksis	S	P

2.1.2.4 Penyusunan Klausa Preposisional

Klausa preposisional adalah klausa yang fungsi P nya diisi oleh frase preposisional. Contoh dapat dilihat pada tabel 2.20.

Tabel 2.20 Contoh pertama klausa preposisional

Kalimat	Ibu dan Ayah	ke pasar
Sintaksis	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 2.21.

Tabel 2.21 Contoh kedua klausa preposisional

Kalimat	Mereka	datang	dari Medan
Sintaksis	S	P	Ket

2.1.2.5 Penyusunan Klausa Numeral

Klausa numeral adalah klausa yang fungsi P nya diisi oleh frase numeral. Contoh dapat dilihat pada tabel 2.22.

Tabel 2.22 Contoh klausa numeral

Kalimat	Mobil pejabat itu	empat buah
Sintaksis	S	P

2.1.3 Penyusunan Kalimat

Satuan bahasa yang menjadi inti dalam pembicaraan sintaksis adalah kalimat yang merupakan satuan di atas klausa dan di bawah wacana. 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 (yang 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.

2.1.3.1 Kalimat Sederhana

Kalimat sederhana adalah kalimat yang dibentuk dari sebuah klausa dasar atau klausa sederhana, yaitu klausa yang fungsi-fungsi sintaksisnya hanya diisi oleh sebuah kata atau sebuah frase sederhana. Contoh: Ayah membaca koran, Kakek tidur di kamar belakang, Istrinya pegawai swasta.

Semua kalimat sederhana yang dibentuk dari klausa dasar 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 2.23.

Tabel 2.23 Contoh kalimat verba monotransitif

Kalimat	Nenek	membaca	buku	di kamar
Sintaksis	S	P	O	Ket

2. Kalimat Verba Bitransitif

   Kalimat verba bitransitif yaitu kalimat yang predikatnya berupa verba yang memiliki komponen makna (+ tindakan), (+ sasaran), dan (+ pelengkap). Contoh dapat dilihat pada tabel 2.24.

Tabel 2.24 Contoh kalimat verba bitransitif

Kalimat	Kakak	membelikan	adik	sepatu baru
Sintaksis	S	P	O	Pel

3. Kalimat Verba Intransitif

Kalimat verba intransitif yaitu kalimat yang predikatnya berupa verba yang memiliki komponen makna (+ tindakan) dan (- sasaran). Contoh dapat dilihat pada tabel 2.25.

Tabel 2.25 Contoh kalimat verba intransitif

Kalimat	Anak-anak itu	menari	di aula
Sintaksis	S	P	O

4. Kalimat Nomina

Kalimat nomina yaitu kalimat yang predikatnya berkategori nomina atau dibentuk dari sebuah klausa nomina dan intonasi *final*. Contoh dapat dilihat pada tabel 2.26.

Tabel 2.26 Contoh kalimat nomina

Kalimat	Orang itu	petani
Sintaksis	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 2.27.

Tabel 2.27 Contoh kalimat ajektifa

Kalimat	Rumahnya	besar sekali
Sintaksis	S	P

6. Kalimat Preposisional

Kalimat prepoisisonal yaitu kalimat yang predikatnya berupa frase preposisional atau dibentuk dari sebuah klausa preposisional dan intonasi *final*. Contoh dapat dilihat pada tabel 2.28.

Tabel 2.28 Contoh kalimat preposisional

Kalimat	Anak-anak itu	di masjid
Sintaksis	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 2.29.

Tabel 2.29 Contoh kalimat numeral

Kalimat	Mobil saya	tiga buah
Sintaksis	S	P

Namun, dalam bahasa ragam formal harus dimunculkan predikat verbanya. Contoh: Pensilnya *berjumlah* lima buah

2.1.3.2 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 satu pada sebuah kalimat. Contoh dapat dilihat pada tabel 2.30.

Tabel 2.30 Contoh pertama kalimat luas

Kalimat	Semalam	dengan diam-diam	adik	membaca	komik	di kamar
Sintaksis	Ket. waktu	Ket. cara	S	P	O	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 2.31.

Tabel 2.31 Contoh kedua kalimat luas

Kalimat	Preman itu	dengan brutal	membunuh	Mang Karta,	seorang pedagang gorengan	di pasar Klender
Sintaksis	S	Ket. cara	P	O	Kel. tambahan pada O	Ket. tempat

3. Disusun dengan memberi keterangan aposisi pada fungsi subjek, objek, atau fungsi lainnya pada kalimat itu. Contoh dapat dilihat pada tabel 2.32.

Tabel 2.32 Contoh ketiga kalimat luas

Kalimat	Jalan kereta api,	alat transportasi masal,	akan dibangun	di pulau Kalimantan
Sintaksis	S	Ket. Aposisi	P	Pel.

4. Disusun dengan cara menyisipkan sebuah klausa pada klausa lain. Klausa yang disispkan disebut klausa sisipan, dan klausa yang tersisipi disebut klausa utama. Penyisipan dilakukan dengan menggunakan konjungsi. Contoh dapat dilihat pada tabel 2.33.

Tabel 2.33 Contoh keempat kalimat luas

Kalimat	Orang yang sedang antre minyak tanah itu bukan kakak saya.
Klausa utama	Orang itu bukan kakak saya
Klausa sisipan	Orang itu sedang antre minyak tanah

2.2 POS Tag

POS (*Part-of-Speech*) Tag merupakan suatu cara pengkategorian kelas kata, seperti kata benda, kata kerja, kata sifat, dan lain-lain. POS *Tagging* adalah suatu aktivitas menganotasi setiap kata/token dengan nilai *part-of-speech* tag yang sesuai. Informasi nilai POS Tag ini merupakan hal yang mendasar bagi keperluan pengolahan bahasa manusia [6].

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. POS Tag diambil pada tanggal 29 Mei 2015.

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. Secara garis besar, fungsionalitas POS *Tagger* dapat terbagi menjadi 6 modul besar [7], yaitu:

1. *Multi-word Expression Tokenizer*

   Merupakan modul tokenisasi yang memperhatikan ekspresi frase yang terdiri lebih dari satu kata. Dokumen diproses oleh *tokenizer* untuk menghasilkan token-token untuk diberikan *part-of-speech* yang sesuai kemudian.

2. *Name Entity Recognizer*

   Merupakan modul yang secara khusus menangani token-token entitas (*proper-noun*), seperti nama orang, nama tempat, nama organisasi, dan lain-lain.

3. *Closed-Class Word Tagging*

   Merupakan modul yang berguna untuk menangani token-token yang termasuk ke dalam kategori *closed-class word* menggunakan sebuah kamus.

4. *Open-Class Word Tagging*

   Merupakan modul yang berguna untuk menangani token-token yang termasuk ke dalam kategori *open-class word* dengan memanfaatkan MorphInd.

5. *Rule-Based Tagging*

   Merupakan modul yang berguna untuk menyelesaikan token-token ambigu berdasarkan aturan-aturan yang telah didefinisikan sebelumnya.

6. *Resolver*

   Merupakan modul yang berguna untuk memberikan tag kepada token-token yang tidak diketahui.

2.3 Teori Bahasa dan Automata

Tata bahasa (*grammar*) bisa didefinisikan secara formal 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, yang disebut dengan *Hirarki Chomsky*.

Tabel 2.34 Tabel penggolongan tingkatan bahasa menurut *Chomsky* [8]

Bahasa	Mesin Otomata	Batasan Aturan Produksi
Natural Language	Mesin Turing	Tidak ada batasan
Context Sensitive	*Linear Bounded Automata*	$\lvert \alpha \rvert \leq \lvert \beta \rvert$
Bebas Konteks	Push Down Automata	α berupa sebuah simbol variabel.
Regular	*Finite State Automata* (FSA) meliputi *Deterministic Finite Automata* (DFA) & *Nondeterministic Finite Automata* (NFA)	α adalah sebuah simbol variabel. β maksimal memiliki sebuah simbol variabel yang bila ada terletak di posisi paling kanan.

Semua 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 "α → β" (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 sedangkan simbol terminal dinyatakan dengan huruf kecil [8].

Berdasarkan tabel 2.34 pembahasan lebih diarahkan ke bahasa bebas konteks karena algoritma CYK hanya mengenali aturan produksi dari *Chomsky Normal Form* (CNF), sedangkan CNF berawal dari *Context Free Grammar* (CFG) atau tata bahasa bebas konteks.

2.3.1 Tata Bahasa Bebas Konteks (TBBK)

TBBK atau CFG adalah suatu cara untuk menunjukkan bagaimana menghasilkan untai-untai dalam sebuah bahasa. Suatu CFG terdiri dari himpunan

simbol terminal dan simbol variabel. Pada CFG aturan produksinya dapat dilihat pada aturan produksi 2.1.

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

Dimana α merupakan sebuah simbol variabel dan β merupakan hasil produksi dari α. Contoh aturan produksi yang dapat dibuat dari CFG dapat dilihat pada aturan produksi 2.2.

$$S \rightarrow aBCde$$
$$B \rightarrow fGe \quad \ldots(2.2)$$

Dimana S merupakan variabel awal dan aBCde merupakan hasil produksi dari variabel S begitu juga dengan variabel B 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.

2.3.2 Penyederhanaan Tata Bahasa Bebas Konteks

Penyederhanaan ini bertujuan untuk melakukan pembatasan sehingga tidak menghasilkan aturan produksi yang rumit dan tidak berguna. Suatu tata bahasa bebas konteks dapat disederhanakan dengan cara:

1. Menghilangkan Produksi *Useless* (tidak berguna)

   Produksi *useless* merupakan produksi yang memuat simbol variabel yang tidak memiliki penurunan yang akan menghasilkan terminal-terminal. Produksi ini tidak berguna karena bila diturunkan tidak akan pernah selesai. Contoh dapat dilihat pada aturan produksi 2.3.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid D \mid \varepsilon$$
$$B \rightarrow b \mid E \mid F$$
$$C \rightarrow bb$$
$$F \rightarrow G$$
$$G \rightarrow bc \quad \ldots(2.3)$$

Berdasarkan contoh diatas dapat dilihat bahwa variabel D dan E tidak memiliki penurunan yang menuju terminal, sehingga variabel ini dapat dihilangkan. Konsekuensinya semua variabel D dan E dihapus dari aturan produksi. Variabel C juga dapat dihapus karena tidak ada variabel yang mengarah ke variabel C. Hasil penyederhanaannya dapat dilihat pada aturan produksi 2.4.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid \varepsilon$$
$$B \rightarrow b \mid F$$
$$F \rightarrow G$$
$$G \rightarrow bc \quad \ldots(2.4)$$

2. Menghilangkan Produksi Unit

Produksi unit adalah produksi dimana ruas kiri dan kanan aturan produksi hanya berupa satu simbol variabel. Keberadaan produksi unit membuat tata bahasa memiliki kerumitan yang tak perlu atau menambah panjang penurunan. Penyederhanaan ini dilakukan dengan cara mengganti aturan produksi unit. Hasil penyederhanaan dapat dilihat pada aturan produksi 2.5.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid \varepsilon$$
$$B \rightarrow b \mid F$$
$$F \rightarrow G$$
$$G \rightarrow bc \quad \ldots(2.5)$$

Berdasarkan contoh aturan produksi 2.5, variabel B memiliki turunan F, variabel F memiliki turunan G dan variabel G memiliki turunan bc, sehingga dapat digambarkan bahwa B → F → G → bc. Aturan ini dapat disederhanakan

dengan cara mengganti aturan produksi dari variabel B menjadi bc. Hasil penyederhanaannya dapat dilihat pada aturan produksi 2.6.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid \varepsilon$$
$$B \rightarrow b \mid bc \quad \ldots(2.6)$$

3. Menghilangkan Produksi Kosong (ε)

Produksi ε atau produksi kosong dapat dihilangkan, karena produksi ini tidak menghasilkan apa-apa. Contoh aturan produksi dapat dilihat pada aturan produksi 2.7.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab \mid \varepsilon$$
$$B \rightarrow b \mid bc \quad \ldots(2.7)$$

Pada aturan produksi 2.7 variabel A mengandung produksi kosong sehingga dapat dihilangkan. Hasil penyederhanaannya dapat dilihat pada aturan produksi 2.8.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab$$
$$B \rightarrow b \mid bc \quad \ldots(2.8)$$

Apabila terdapat kasus seperti aturan produksi 2.9.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow \varepsilon$$
$$B \rightarrow b \mid bc \quad \ldots(2.9)$$

Semua variabel A yang ada didalam aturan produksi dihapus karena variabel A mengandung produksi kosong. Hasil penyederhanannya dapat dilihat pada aturan produksi 2.10.

$$S \rightarrow a \mid B$$
$$B \rightarrow b \mid bc \quad \ldots(2.10)$$

2.3.3 *Chomsky Normal Form* (CNF)

CNF merupakan salah satu bentuk normal yang sangat berguna untuk CFG. Bentuk CNF dapat dibuat dari sebuah CFG yang telah mengalami penyederhanaan

dengan menghilangkan produksi *useless*, unit, dan *empty* (himpunan kosong yang dilambangkan dengan ε).

Aturan produksi dalam bentuk normal *Chomsky* ruas kanannya tepat berupa sebuah terminal atau dua variabel. Contoh aturan produksi dapat dilihat pada 2.11.

$$A \rightarrow BC$$
$$A \rightarrow b$$
$$B \rightarrow a$$
$$C \rightarrow BA \mid d \quad \ldots(2.11)$$

Langkah-langkah dalam pembentukan CNF yaitu:

1. Biarkan aturan produksi yang sudah dalam bentuk CNF.
2. Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1.
3. Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol variabel.
4. Penggantian dapat dilakukan berkali-kali sampai semua aturan produksi berada dalam bentuk CNF.
5. Selama dilakukan penggantian, kemungkinan akan memperoleh aturan-aturan produksi baru dan memunculkan simbol-simbol variabel baru.

Contoh CFG yang sudah disederhanakan dapat dilihat pada aturan produksi 2.12.

$$S \rightarrow bC \mid aB \mid AB$$
$$A \rightarrow bAC \mid a$$
$$B \rightarrow aCB \mid b$$
$$C \rightarrow cBA \mid c \quad \ldots(2.12)$$

Pada bentuk CFG diatas, aturan produksi yang sudah dalam bentuk CNF adalah $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, dan $C \rightarrow c$. Selebihnya harus dilakukan penggantian aturan produksi. Hasil penggantian dapat dilihat pada aturan produksi 2.13.

$S \rightarrow bC$	menjadi	$S \rightarrow V_1C$	
$S \rightarrow aB$	menjadi	$S \rightarrow V_2B$	
$A \rightarrow bAC$	menjadi	$A \rightarrow V_1AC$	
$A \rightarrow V_1AC$	menjadi	$A \rightarrow V_1V_3$	
$B \rightarrow aCB$	menjadi	$B \rightarrow V_2CB$	
$B \rightarrow V_2CB$	menjadi	$B \rightarrow V_2V_4$	
$C \rightarrow cBA$	menjadi	$C \rightarrow V_5BA$	
$C \rightarrow V_5BA$	menjadi	$C \rightarrow V_5V_6$	…(2.13)

Aturan produksi baru dapat dilihat pada aturan 2.14.

$$V_1 \rightarrow b$$
$$V_2 \rightarrow a$$
$$V_3 \rightarrow AC$$
$$V_4 \rightarrow CB$$
$$V_5 \rightarrow c$$
$$V_6 \rightarrow BA \quad \text{…(2.14)}$$

Hasilnya dapat dilihat pada aturan produksi 2.15.

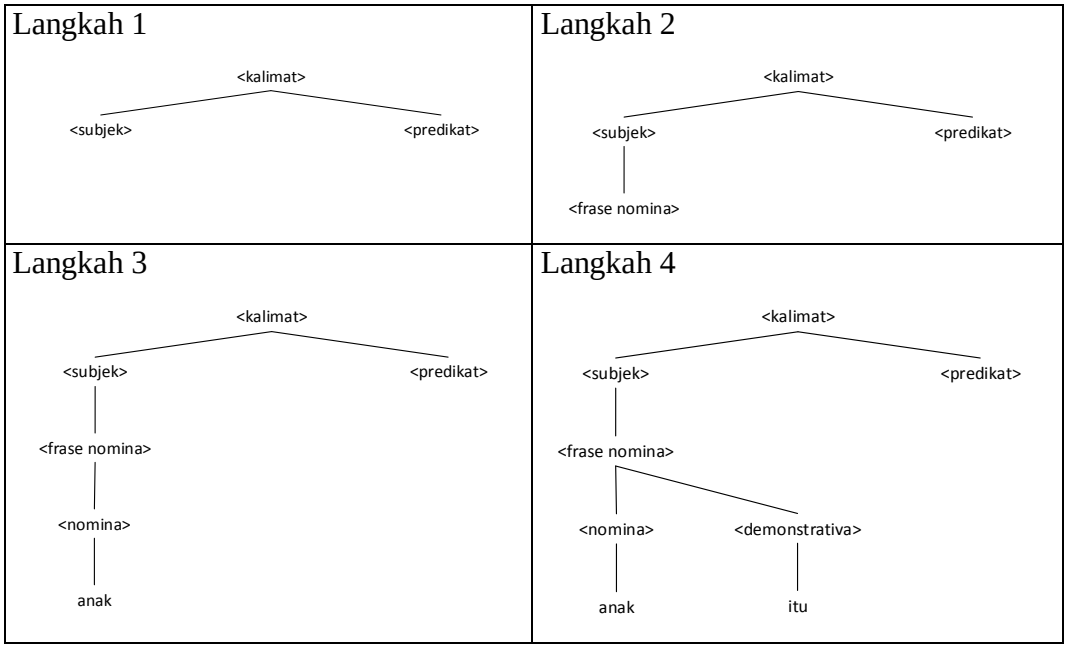
$$S \rightarrow V_1C$$
$$S \rightarrow V_2B$$
$$S \rightarrow AB$$
$$A \rightarrow V_1V_3$$
$$A \rightarrow V_2$$
$$B \rightarrow V_2V_4$$
$$B \rightarrow V_3$$
$$C \rightarrow V_5V_6$$
$$C \rightarrow V_3$$
$$V_1 \rightarrow b$$
$$V_2 \rightarrow a$$
$$V_3 \rightarrow AC$$
$$V_4 \rightarrow CB$$
$$V_5 \rightarrow c$$
$$V_6 \rightarrow BA \quad \text{…(2.15)}$$

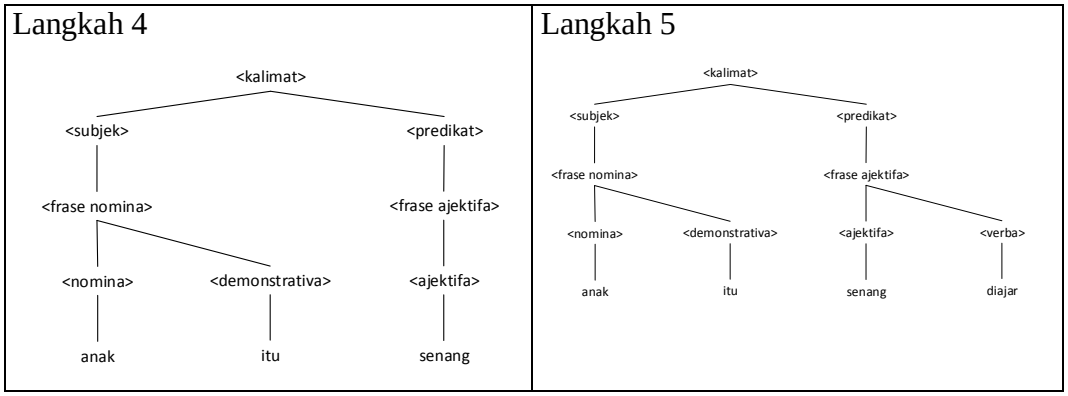
2.3.4 Teknik-Teknik *Parsing*

Dalam ilmu komputer dan linguistik, *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:

2.3.4.1 *Top-Down Parsing*

Pada teknik ini proses *parsing* dilakukan mulai dari *root* dan mencari terminal-terminal yang berupa kata-kata. Jika penelusuran pencarian terminal tidak ditemukan pada satu simbol, dapat dilakukan pencarian ulang (*backtracking*) pada simbol lain. Cara kerja *top-down parsing* dapat dilihat pada gambar 2.1. Kata yang digunakan sebagai contoh adalah "Anak itu senang diajar".



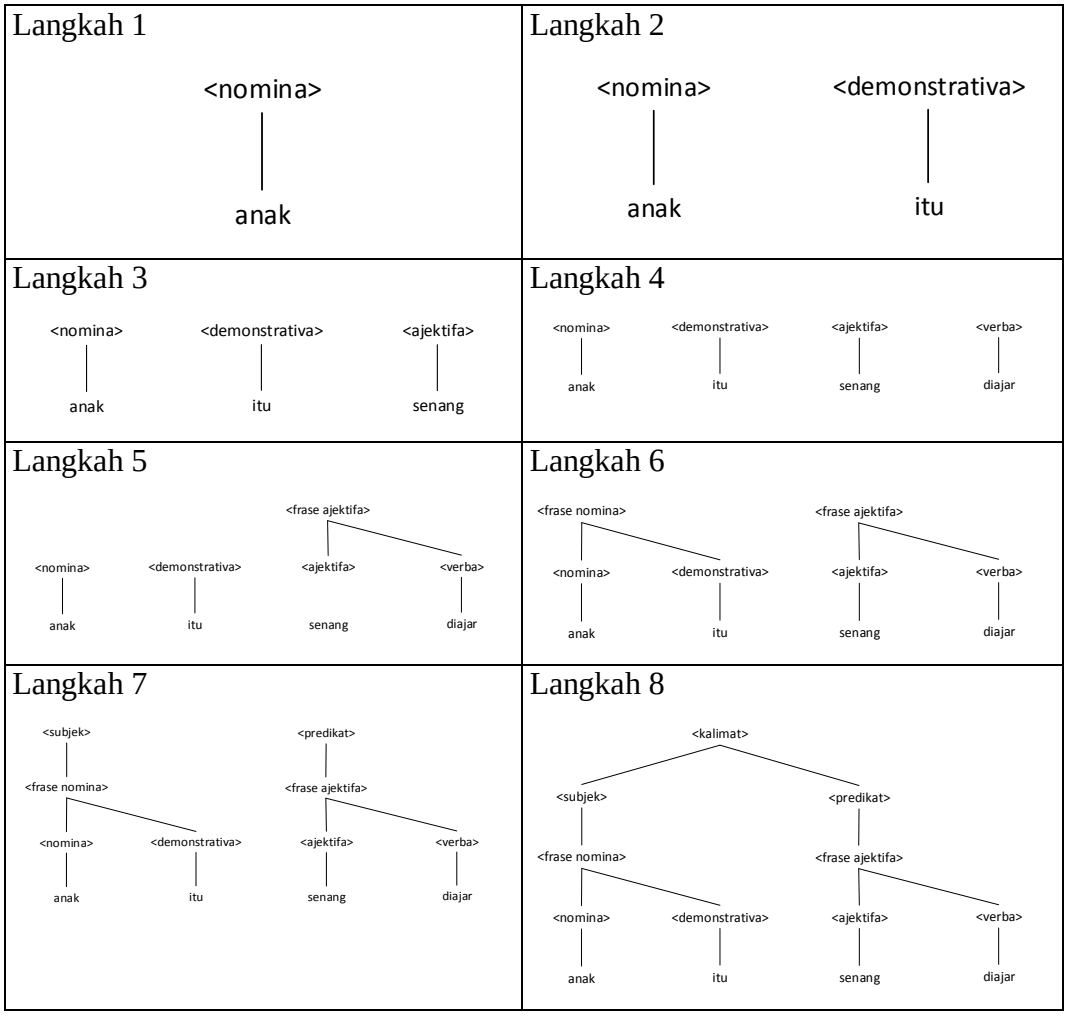


Gambar 2.1 Cara kerja *top-down parsing* [9]

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* <demonstrativa> dilakukan penelusuran *tag* <nomina>, <verba>, <ajektiva>, <adverbia>, dan <numeralia>. Penelusuran tersebut disesuaikan dengan penyusunan frase nomina yang terdapat pada sub bab 2.1.1.1.

2.3.4.2 *Bottom-Up Parsing*

Teknik ini bekerja secara terbalik dibandingkan dengan *top-down parsing*. Pada teknik ini proses *parsing* dilakukan mulai dari terminal-terminal yang kemudian dirangkaikan menjadi sebuah kalimat atau hingga menuju *root*. Metode parsing ini digunakan oleh algoritma CYK.



Gambar 2.2 Cara kerja *bottom-up parsing* [9]

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

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

Algoritma CYK merupakan algoritma *parsing* dan keanggotaan (*membership*) untuk tata bahasa bebas konteks. Syarat untuk menggunakan

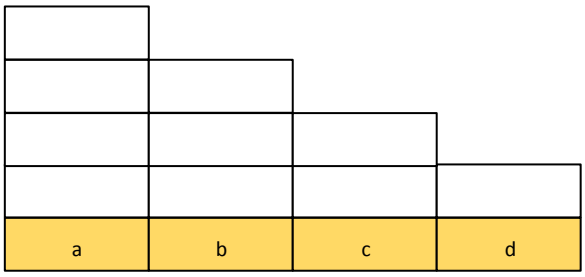
algoritma ini adalah tata bahasa harus berada dalam bentuk CNF. Tujuan dari algoritma ini adalah untuk menunjukkan apakah suatu *string* dapat diterima dari suatu tata bahasa.

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 ini memanfaatkan hasil *parsing* sebelumnya untuk menentukan apakah proses yang sedang berlangsung dapat diterima atau tidak.

Sebagai contoh terdapat suatu aturan produksi 2.16 dengan variabel awal S.

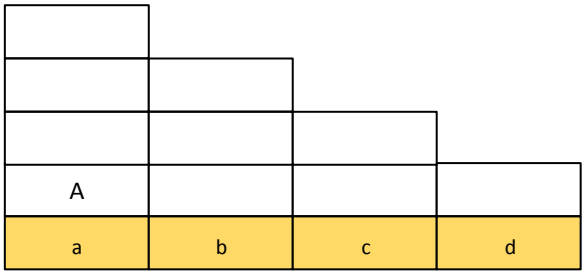
$$S \rightarrow EF \mid EC$$
$$E \rightarrow AB$$
$$F \rightarrow CD$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$C \rightarrow c$$
$$D \rightarrow d \quad \ldots(2.16)$$

String yang akan diperiksa misalnya “abcd”. Langkah pertama yang dilakukan adalah dengan membuat tabel dua dimensi seperti gambar 2.1.



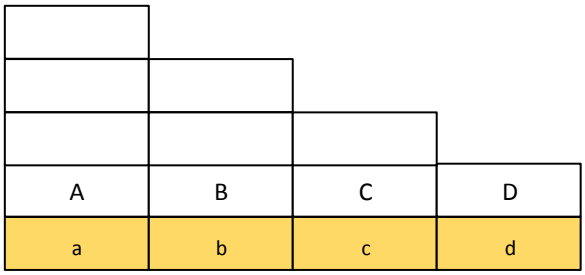
Gambar 2.3 Pembentukan tabel *parsing* algoritma CYK

Setiap kata yang ada pada suatu *string* ditempatkan pada masing-masing blok seperti gambar 2.3. 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 2.4.



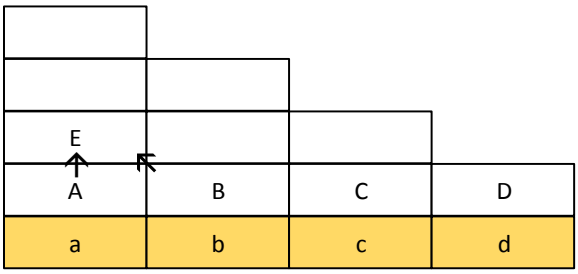
Gambar 2.4 Pengisian tabel dengan variabel A

Blok selanjutnya diisi dengan langkah yang sama, sehingga dapat dilihat baris pertama isiannya seperti gambar 2.5.



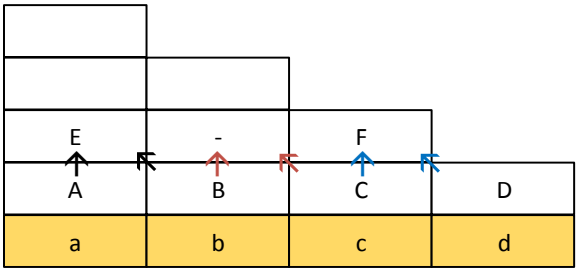
Gambar 2.5 Pengisian tabel baris pertama

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 E, sehingga cara pengisiannya terlihat seperti pada gambar 2.6.



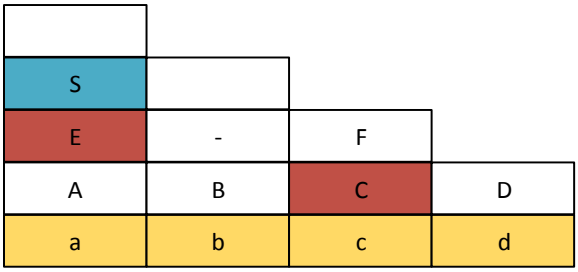
Gambar 2.6 Pengisian tabel dengan variabel B

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 2.7.

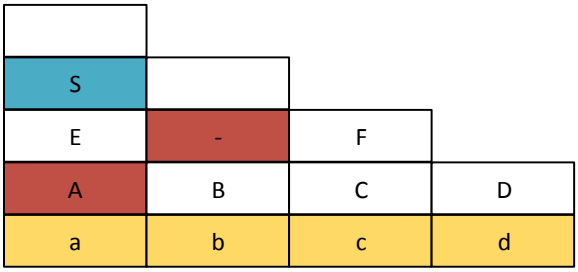


Gambar 2.7 Pengisian tabel baris kedua

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 2.8 dan 2.9.



Gambar 2.8 Cara pengecekkan langkah pertama



Gambar 2.9 Cara pengecekkan langkah kedua

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

S			
S	-		
E	-	F	
A	B	C	D
a	b	c	d

Gambar 2.10 Hasil *parsing string* dengan algoritma CYK

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

2.5 Algoritma Levenshtein

Algoritma Levenshtein merupakan algoritma pencarian jumlah perbedaan *string* yang ditemukan oleh Vladimir Levenshtein, seorang ilmuwan Rusia, pada tahun 1965. Algoritma ini digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spell checking*), pengenalan pembicaraan (*speech recognition*), pengucapan dialek, analisis DNA, dan lain-lain.

Dua buah *string* dibandingkan untuk mengetahui nilai *edit distance*-nya. *Edit distance* adalah suatu pengukuran (metrik) yang dihasilkan melalui perhitungan jumlah perbedaan yang terdapat pada dua *string*. *Edit distance* antara dua *string* didefinisikan sebagai jumlah minimum perubahan yang diperlukan untuk mengganti suatu *string* dengan *string* lain, dengan operasi penambahan (*insert*), operasi penggantian (*substitute*), operasi penghapusan (*delete*).

Algoritma ini menggunakan matriks dua dimensi dalam membandingkan kedua *string*. Algoritma ini berjalan mulai dari pojok kiri atas hingga ke pojok kanan bawah. Seluruh baris pertama diisi dengan nilai berurut mulai dari 0 hingga n begitu juga dengan kolom pertama. Cara pengisiannya dapat dilihat pada gambar 2.9.

		T	O	P
	0	1	2	3
T	1			
O	2			
P	3			
I	4			

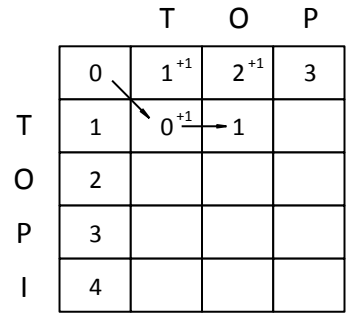
Gambar 2.11 Inisialisasi awal tabel *levenshtein*

Pada gambar diatas kata “top” akan dilakukan perbandingan dengan kata “topi” untuk melihat nilai *edit distance*-nya. Pengisian dilakukan mulai dari blok [1,1] dengan membandingkan karakter yang ada pada blok [0,1] dan blok [1,0]. Apabila karakter yang dibandingkan sama, turunkan nilai yang ada pada blok [0,0] yaitu angka 0. Nilai *edit distance*-nya dapat dilihat pada gambar 2.12.

		T	O	P
	0	1	2	3
T	1	0		
O	2			
P	3			
I	4			

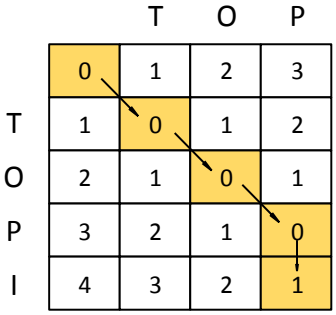
Gambar 2.12 Penurunan angka 0 ke baris 1 kolom 1

Pengisian blok berikutnya dilakukan dengan membandingkan karakter dengan baris yang sama. Apabila karakter yang dibandingkan tidak sama, maka nilai blok [0,2] ditambahkan dengan angka 1. Sama halnya dengan blok[1,1] dan blok [0,1]. Setelah itu lakukan perbandingan nilai dari blok [0,2], [1,1], dan [0,1] untuk mencari nilai minimumnya. Selanjutnya simpan pada blok [1,2]. Hasilnya dapat dilihat pada gambar 2.13.



Gambar 2.13 Pengisian blok [1,2]

Langkah selanjutnya isi semua blok sampai akhir dengan langkah yang sama. Hasil akhir dari pengisian tabel Levenshtein diatas dapat dilihat pada gambar 2.14.



Gambar 2.14 Hasil akhir pengisian tabel Levenshtein

Nilai *edit distance* dapat dilihat dari baris dan kolom terakhir, yaitu angka 1. Nilai inilah yang akan mengetahui seberapa banyak perbedaan karakter antara dua *string*.

2.6 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi 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.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah 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

   Pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Sifat pewarisan 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.

Beberapa konsep dasar dari pemrograman berorientasi objek adalah:

1. Kelas (*Class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Sebuah kelas akan mempunyai sifat (*attribut*), tingkah laku (operasi/metode), hubungan (relationship) dan arti. 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 *file* yang berisi kode program, dimana kode program merupakan semua hal yang terkait dengan nama kelas.

2. Objek (*Object*)

   Objek adalah abstrak 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 dan mempunyai operasi 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, dimanupulasi, 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. Metode berfungsi untuk memanipulasi objek itu sendiri. Metode merupakan fungsi atau transformasi uang dapat dilakukan terhadap objek atau yang dilakukan objek. Metode dapat berasal dari *event*, aktivitas atau keadaan, fungsi, atau tingkah laku. Contohnya seperti *read*, *write*, *move*, *copy*, dan sebagainya.

4. Atribut

   Atribut adalah variabel yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya seperti berat, jenis, nama, dan sebagainya.

5. Abstrak (*Abstract*)

   Abstrak 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

   Enkapsulasi merupakan pembungkusan atribut data dan layanan (operasi-operasi) yang dimiliki objek untuk menymbunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan

   Pewarisan merupakan mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

   Interface sangat mirip dengan kelas, tetapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode dari sebuah *interface* dapat diimplementasikan oleh kelas lain. Sebuah kelas dapat mengimplementasikan lebih dari satu *interface*.

2.7 *Unified Modelling Language* (UML)

UML adalah sebuah *tools* atau alat yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan, membuat analisis & *design*, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

2.7.1 Use Case Diagram

Use case diagram merupakan pemodelah untuk mengetahui tingkah laku (*behaviour*) sistem yang akan dibuat. *Use case* 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 dan siapa saja yang berhak menggunakan sistem fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesingkat dan sejelas mungkin agar dapat dipahami. Ada dua hal dalam membuat sebuah *use case,* yaitu:

1. Aktor

   Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem utama yang akan dibuat. Meskipun simbol aktor itu adalah orang, tetapi belum tentu aktor itu adalah 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 Class Diagram

Class diagram atau kelas diagram menggambarkan struktur sistem dari segi pemdefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode. Diagram kelas dibuat agar pembuat

program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak dapat sinkron.

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

1. Kelas *main*

   Kelas main merupakan kelas yang mempunyai fungsi awal yang dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*)

   Kelas ini mendefinisikan dan mengatur tampilan ke pengguna.
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas ini menangani fungsi-fungsi yang harus ada diambil dari pendefinisisan *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 ini digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas.

2.7.3 Activity Diagram

Activity diagram atau diagram aktivitas mengambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Perlu diperhatikan bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

2.7.4 Sequence Diagram

Sequence diagram atau diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Untuk menggambarkan diagram sekuen harus diketahui metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.