PERTEMUAN 12

PENGURAIAN BAWAH ATAS (BOTTOM UP PARSING)

A. TUJUAN PEMBELAJARAN

Setelah pembelajaran materi untuk pertemuan 12, mahasiswa mampu menggunakan prinsip dasar kerja penguraian dari bawah ke atas, serta memahami pola kerja penguraian dari bawah ke atas.

B. URAIAN MATERI

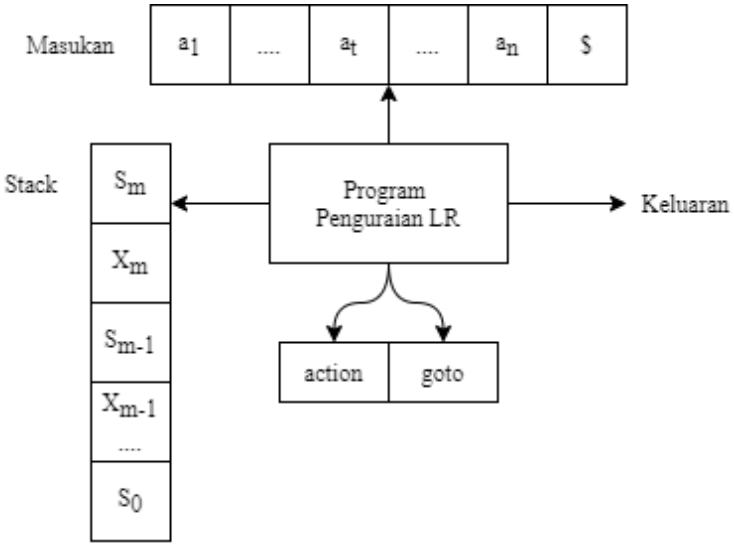
1. Pengertian Pengurai LR

Pengurai LR juga termasuk dalam golongan pengurai bawah ke atas. Sebenarnya pengurai ini tidak cukup di beri nama pengurai LR tetapi LR(1), karena berasal dari LR(k), dengan L berarti melihat masukan dari kiri (left), R yang berarti untuk membuat penurunan paling kanan (rightmost derivation) dalam kebalikannya, dan k yang melambangkan banyaknya simbol yang harus dilihat. Jika k tidak disebutkan maka berarti k dianggap 1.

a. Algoritma Pengurai LR

Bentuk Semantik dari penguraian LR diperlihatkan dalam gambar 12.1 dibawah penguraian LR digambarkan mempunyai lima komponen, yaitu masukan, keluaran, stack, program driver, dan tabel penguraian LR yang terbagi menjadi dua bagian yaitu action dan goto.

Program pengurai membaca karakter karakter dari penyangga masukan sekali setiap waktu . Program pengurai mempergunakan stack untuk memasukkan string yang berbentuk $S_0X_1S_1X_2 \dots X_mS_m$ Di mana S_m Berada di puncak dari stack X_1, adalah simbol tatabahasa sedangkan S_m Adalah suatu symbol biasa disebut state. Setiap state merangkum informasi pada stack yang berada dibawah state tersebut, dan kombinasi dari state yang berada di Puncak stack dengan simbol masukan yang sedang dibaca digunakan untuk mengindek tabel penguraian dan menentukan Tindakan selanjutnya dari pengurai, yaitu shift atau reduce.



Gambar 12.1 Model Pengurai LR

Tabel penguraian memuat dua komponen, yaitu aksi penguraian dengan fungsi action dan aksi pergi ke dengan fungsi goto. Pada awalnya program pengurai menentukan S_m, yaitu state yang berada di puncak stack dan a_1 simbol masukan yang sedang dibaca. Langkah selanjutnya adalah melihat tabel action $[S_m, a_i]$, yaitu isi dalam tabel penguraian bagian action untuk state S_m dan masukan a_i nilai dari tabel action $[S_m, a_i]$, dapat berarti :

1) Shift s, dimana s adalah state, atau
2) Reduce dengan produksi tatabahasa A→β atau
3) Accept, atau
4) Eror.

sedangkan fungsi goto mengambil sebuah states dan symbol tatabahasa sebagai argument dan memproduksi sebuah state.

Konfigurasi dari penguraian LR merupakan suatu pasangan yang komponen pertama berisi tumpukan/stack dan komponen keduanya adalah masukkan yang tidak dikembangkan, yaitu:

$$(S_0 X_1 S_1 X_2 S_2 \dots X_m\ S_m, a_i + 1 \dots a_n\ \$)$$

Konfigurasi ini mempresentasikan bentuk sentential kanan:

$$X_1 X_2 \dots X_m\ a_i\ a_i + 1 \dots a_n$$

yang =bentuk sentential kanan dari pengurai shift reduce.

Langkah selanjutnya ditentukan oleh pembacaan a_1 dan S_m, yaitu dengan melihat aksi pengurai pada tabel penguraian action $[S_m, a_i]$, yang mempunyai 4 kemungkinan yaitu:

1) jika Action $[S_m, a_i]$, - shift s, maka pengurai akan melakukan shift, dan menghasilkan konfigurasi $(S_0X_1S_1X_2S_2 \dots X_mS_m a_i S, a_i{+}_1 \dots a_n \$)$, disini pengurai telah melakukan shift pada simbol masukan a_i, dan state s berikutnya yang diperoleh dari tabel action $[S_m, a_i]$, ke dalam stack dan a_{i+1} menjadi simbol masukkan berikutnya.

2) Jika action $[S_m, a_i]$ = reduce A→β maka pengurai akan melakukan reduce, dan menghasilkan konfigurasi $(S_0X_1S_1X_2S_2 \dots X_mS_m a_i S, a_i{+}_1 \dots a_n \$)$ di mana S = goto $[S_{m-}r_3, A]$ dan r adalah Panjang dari β.

   dengan adanya reduce pengurai pertama * akan mem-pop simbol sebanyak 2r dari stack (r simbol state dan r simbol tatabahasa), sehingga mendapatkan state S_{m-r} pengurai kemudian mem-push A dan s ke dalam stack, di mana A adalah Sisi kiri dari produksi yang direduksi, dan s adalah state yang diperoleh dari tabel goto $S_{m-r}\ A$. sementara itu simbol masukan tidak berubah dengan adanya aksi reduce Ini.

   keluaran dari penguraian LR akan dibentuk setelah reduksi dengan mengeksekusi aksi semantik yang berkaitan dengan reduksi produksi.

3) jika action $[S_m\ a_1]$ = accept maka penguraian selesai .

4) jika action $[S_m\ a_1]$ = error maka pengurai telah menemukan adanya kesalahan dan memanggil rutin pembenahan kesalahan.

algoritma pengurai LR bekerja dengan cara demikian siapkan string w dan tabel penguraian LR dengan fungsi action dan goto untuk tatabahasa G. Mula-mula pengurai mempunyai S_0 pada puncak stack di mana S_0 adalah awal dan w$ berada di dalam penyangga masukkan . Pengurai kemudian akan mengeksekusi program 12.2 sampai ditemukan aksi Accept atau mungkin error. Jika w berada di dalam G, maka keluaran yang diperoleh

berupa penguraian dasar ke atas untuk string w, sebaliknya jika tidak maka keluaran yang akan diperoleh adalah sebuah pesan kesalahan.

```
Setel ip pada symbol dari w$;
repeat forever begin
        Misalkan s di puncak stack dan a symbol yang ditunjuk oleh ip;
        If actions [s, a] = shift's then begin
                Push-lah a lalu s ke dalam stack;
                Ganti ip dengan symbol masukan berikutnya;
        End
        Else if action [ s, a] = reduce A→β then begin
                Pop-lah symbol dalam stack sebanyak 2*|β| dari stack;
                Misalkan s adalah state pada puncak stack;
                Push-lah A dan isi tabel goto [ s, A] ke dalam stack;
                Berikan keluaran produksi A→β
        End
        Else if action [s,a] = accept then
                Return
        Else error()
End
```

Program 12.3 Program penguraian LR.

Contoh 12.1 diberikan tatabahasa sebagai berikut:

(1) $E \rightarrow E + T$

(2) $E \rightarrow T$

(3) $T \rightarrow T * F$

(4) $T \rightarrow F$

(5) $F \rightarrow (E)$

(6) $F \rightarrow var$

(Tatabahasa 7)

Dan tabel penguraian untuk tatabahasa tersebut dapat dilihat dalam tabel 12.1 dibawah dengan kode-kode untuk aksi adalah:

1) *si* yang berarti shift dan stack state ke i,
2) *rj* yang berarti reduce dengan produksi nomor j,
3) *acc* yang berarti accept
4) tidak ada simbol atau kosong berarti *error*.

Tabel 12.1 Tabel Penguraian untuk ekspresi tatabahasa (7)

State	Action						goto		
	var	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7				s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Maka pergerakan pengurai LR dengan masukkan $var * var + var$ seperti diperlihatkan dalam tabel 12.2 berikut ini :

Tabel 12.2 Tindakan – Tindakan pengurai LR dengan masukan var * var + var

Stack	Masukan	Aksi
(1) 0	var * var + var $	shift
(2) 0 var 5	* var + var $	reduce oleh F → var
(3) 0 F 3	* var + var $	reduce oleh T → F
(4) 0 T 2	* var + var $	shift
(5) 0 T 2 * 7	var + var $	shift
(6) 0 T 2 * 7 var 5	+ var $	reduce oleh F → var
(7) 0 T 2 * 7 F 10	+ var $	reduce oleh T → T * F
(8) 0 T 2	+ var $	reduce oleh E → T
(9) 0 E 1	+ var $	shift
(10) 0 E 1 + 6	var $	shift
(11) 0 E 1 + 6 var 5	$	reduce oleh F → var
(12) 0 E 1 + 6 F 3	$	reduce oleh T → F
(13) 0 E 1 + 6 T 9	$	reduce oleh E → E + T
(14) 0 E 1	$	accept

b. Membentuk Tabel Penguraian SLR

SLR adalah singkatan dari simple LR, yaitu sebuah metode untuk membentuk tabel penguraian LR yang dikenal mempunyai kemudahan dalam implementasinya. Masih ada dua metode pembentukan tabel selain SLR yaitu LR kanonik, dan LALR yang merupakan singkatan dari look ahead LR, namun penulis hanya akan mengambil 1 metode pembentukan tabel penguraian yaitu SLR.

Sebuah item LR (0) atau disebut item saja untuk tatabahasa G adalah suatu produksi dari G dengan suatu dot (.) Muncul di sisi kanan produksi . Jadi, produksi $A \rightarrow XYZ$ memberikan 4 item sebagai berikut:

$A \rightarrow XYZ$

$A \rightarrow X \cdot Y\ Z$

$A \rightarrow XY \cdot Z$

$A \rightarrow XYZ \cdot$

Produksi A→ ε hanya membentuk satu item, yaitu A → . Suatu item dapat dipresentasikan dengan sebuah pasangan integer, dengan integer yang pertama menggambarkan banyaknya produksi dan integer yang ke-2 menggambarkan posisi dari dotnya. Suatu item dapat mengindikasikan Berapa banyak produksi yang sudah dilihat dalam satu langkah proses penguraian. dari contoh diatas, dari item pertama menyatakan bahwa kemungkinan dapat terdapat harapan untuk melihat sebuah string yang dapat diturunkan dari xyz item yang kedua mengindikasikan bahwa telah terlihat pada masukan suatu string yang dapat diturunkan dari X, dan diharapkan selanjutnya dapat terlihat suatu string yang dapat diturunkan dari YZ.

Ide utama dari metode SLR adalah pertama buat dari tatabahasa yang dipakai sebuah otomata deterministik yang berhingga yang dapat mengenal Prefiks Prefiks yang viable . Kelompok item-item kedalam himpunan-himpunan yang akan menjadi state – state dari pengurai SLR.

satu koleksi dari himpunan item LR(0) yang disebut dengan koleksi kanonik LR (0) memberikan dasar pembentukan pengurai SLR untuk membentuk koleksi kanonik LR(0) dari suatu tatabahasa perlu di definisikan suatu tatabahasa tertambah dan dua fungsi yaitu fungsi closure dan goto.

Jika G adalah suatu tatabahasa dengan simbol awal S maka G adalah tatabahasa tertambah untuk G dengan simbol awal S dan produksi S → S tujuan dari produksi awal ini adalah untuk mengindikasikan pada pengurai Kapan pengurai harus berhenti melakukan penguraian dan kapan menyatakan penerimaan suatu masukan.

dengan perkataan lain penerimaan terjadi bila hanya bila pengurai melakukan reduksi dengan $S' \rightarrow S$.

2. Prefiks Viable

Prefiks viable adalah himpunan Prefiks dari bentuk sentential kanan yang dapat muncul pada stack pengurai shift reduce. berdasarkan definisi ini maka selalu mungkin untuk menambahkan simbol terminal di akhir Prefiks viable untuk memperoleh bentuk sentential kanan akibatnya tidak akan dijumpai kesalahan jika Sebagian masukan yang dilihat dapat direduksi dengan Prefiks viable.

3. Operasi Closure

Jika a adalah himpunan item untuk tatabahasa g. Maka closure (1) adalah himpunan item yang dibangun dari dengan mempergunakan 2 aturan berikut ini:

a. Mula-mula setiap item dalam I di+kan kedalam closure (I).

b. Jika A → a Bβ didalam closure (I) dan B → y adalah suatu produksi, maka +kan item B→γ ke dalam I, apabila item B → γ belum terdapat di dalam I, pakai aturan ini sampai tidak ada lagi item-item baru yang dapat di+kan kedalam closure (I). A→αBβ di dalam closure (I) mengindikasikan bahwa pada suatu saat dalam proses penguraian diperkirakan adanya suatu substring yang dapat diturunkan dari Bβ sebagai masukan. Jika B → γ adalah suatu produksi dapat diharapkan juga adanya suatu substring yang dapat diturunkan dari γ. Dengan alasan inilah B → γ dimasukkan ke dalam (1) dengan mempergunakan tatabahasa (9), tatabahasa tertambah g untuk kata Bahasa (9) adalah sebagai berikut Jika a adalah himpunan 1 item maka closure (I).

Contoh 12.2 Dengan mempergunakan tatabahasa (9), tatabahasa tertambah G' untuk tatabahasa (9) adalah sebagai berikut:

$E' \rightarrow E$

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid var.$

Jika 1 adalah himpunan satu item $\{[\, E' \rightarrow E \,]\}$, maka closure(I) memuat:

$E' \rightarrow E$

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow var$

keterangan dari hasil perolehan di atas adalah sebagai berikut mula-mula item $E' \rightarrow E$ dimasukkan ke dalam Closer(1) dengan mempergunakan aturan a). Karena terdapat nonterminal E tepat disebelah kanan dot, maka berdasarkan aturan b). +kan item-item $E \rightarrow \gamma$ ke dalam closure(I). item tersebut adalah $E \rightarrow E + T \; dan \; E \rightarrow T$ dari item E → T, nonterminal T tepat di sebelah kanan dot., maka berdasarkan aturan b). +kan item-item T → ·T*F dan T→·F dari item T → ·F nonterminal F tepat di sebelah kanan dot. maka berdasarkan aturan b) di+kan item-item F → ·(E) dan F→var, dan ternyata tidak mungkin lagi dibuat item baru, Maka proses pencarian anggota closure (I) selesai.

Fungsi closure dapat dihitung seperti diperlihatkan dalam program 3.3 berikut:

```
function closure(I);
Begin
    J;" I;
    Repeat
    For untuk setiap item A→α:Bβ didalam J dan
    Setiap produksi B→γ dalam G sehingga B→·γ tidak terdapat dalam J
        do +kan item B→γ ke dalam J
    null tidak ada lagi item baru yang dapat di+kan ke dalam J;
    return (J)
    end
```

Program 12.3 Program perhitungan closure.

4. Operasi Goto

fungsi ke dua yang sangat berguna adalah fungsi goto(I, X) dengan I adalah himpunan item dan X adalah simbol tatabahasa. goto(I, X) adalah closure dari himpunan semua item $[A \rightarrow \alpha \cdot \beta]$ sedemikian sehingga $[A \rightarrow \alpha \cdot X\beta]$ berada didalam I. Jika I adalah himpunan semua item yang untuk beberapa prefiks viable γ, maka go to (I, X) adalah himpunan item yang valid untuk Prefiks viable γX.

contoh 12.3 jika I adalah himpunan maka $\{[\, E' \rightarrow E \cdot], [E \rightarrow E \cdot + T]\}$, maka dapat dicari goto (I, +) untuk menghitung goto (I, +) pertama * mencari item dalam I yang mempunyai simbol + tepat di sebelah kanan dot. Item [E' → E] tidak memenuhi Aturan ini tetapi pada item $[E \rightarrow E \cdot + T]$ terlihat bahwa simbol + tepat di sebelah kanan Dot. sehingga diperoleh item baru $[E \rightarrow E + \cdot T]$. Langkah berikutnya adalah mencari closure $\{[E \rightarrow E + \cdot T]\}$. Dari item baru ini terlihat bahwa nonterminal T tepat di sebelah kanan dot. Maka berdasarkan aturan b) dari aturan pembentukan closure, tambahkan item-item $T \rightarrow \cdot T * F \; dan \; T \rightarrow \cdot F$ Dari item $T \rightarrow \cdot F$, nonterminal F tepat di sebelah kanan dot. Maka berdasarkan aturan b) dari aturan pembentukan closure tambahkan item-item $F \rightarrow \cdot (E)$ dan $F \rightarrow var$. Proses selesai, dan goto $(I, +)$ memuat:

$E \rightarrow E + \cdot T$
$T \rightarrow \cdot T * F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot var.$

5. Pembentukan Himpunan Item

Program berikut ini dipakai untuk membuat koleksi kanonik dari himpunan item LR (0) untuk sebuah tatabahasa tertambah G'.

```
procedure item (G');
begin
    c:"(closure({[S'→S]});
    repeat
        for untuk setiap di dalam I dan setiap symbol tatabahasa X
        sedemikian sehingga goto(LX) tidak kosong dan tdk terdapat
dalam C
                do tambahkan goto (LX) ke dalam C
    null tidak ada lagi himpunan item baru yang dapat ditambahkan ke
dalam C
end
```

Program 12.4 Program pembentukan himpunan item.

Contoh 12.4, dengan mempergunakan tatabahasa (9) akan dicari koleksi kanonik dari himpunan item LR(0) untuk tatabahasa (9) tersebut dengan mempergunakan panduan dari program 12.4. diatas.

mula-mula hitung closure $(\{[E' \rightarrow \cdot E]\})$. misalkan closure $(\{[E' \rightarrow \cdot E]\})$. disebut I_0 maka berdasarkan pada contoh 12.2 . dapat disimpulkan:

$I_0 = \{[E' \rightarrow \cdot E], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot var]\}$. I_0 mempunyai tujuh buah item dan delapan buah simbol tatabahasa X yaitu $E, F, T, +, *, (,), dan\ var$.

langkah selanjutnya adalah menghitung goto (I_0, X) untuk berbagai nilai X. untuk $X = E, item$ dalam I_0 dengan non-terminal E tepat disebelah kanan dot adalah $[E' \rightarrow \cdot E]$ dan $[E \rightarrow \cdot E + T]$. Untuk $[E' \rightarrow \cdot E]$ maka $E' \rightarrow \cdot E\ \varepsilon$ goto (I_0, E). Kemudian hitung closure $(\{[E' \rightarrow E \cdot]\})$ karena dalam item baru tersebut dot sudah terletak diakhir kanan maka tidak terdapat tambahan closure dari item tersebut. Untuk $[E \rightarrow E \cdot +T]$ maka $E \rightarrow E \cdot +T\ \varepsilon\ goto(I_0, E)$. Kemudian hitung closure $(\{[E \rightarrow E \cdot +T]\})$. karena ternyata di sebelah kanan dot adalah simbol Terminal maka juga tidak ada tambahan closure.

jadi $I_1 = \{[E' \rightarrow E \cdot], [E \rightarrow E \cdot T]\}$.

untuk $X = T, item$ dalam I_0 dengan non-terminal T tepat di sebelah kanan dot adalah $[E \rightarrow \cdot T]$ dan $[T \rightarrow \cdot T * F]$ untuk $[E \rightarrow \cdot T]$ maka $E \rightarrow T \cdot$. ε goto (I_0, T). kemudian hitung closure $(\{[E \rightarrow T \cdot]\})$. karena dalam item baru tersebut sudah

terletak diakhir kanan maka tidak terdapat tambahan closure dari item tersebut. untuk ($\{[T \rightarrow \cdot T * F]\}$) maka $T \rightarrow T \cdot * F\ \varepsilon\ goto\ (I_0, T)$. kemudian hitung closure ($\{[T \rightarrow T \cdot * F]\}$) karena ternyata di sebelah kanan dot adalah suatu simbol Terminal maka juga tidak ada tambahan Closure.

jadi $I_2 = \{[E \rightarrow T \cdot],\ [T \rightarrow T \cdot * F]\}$.

untuk X = F item I_0 dengan non-terminal F tepat di sebelah kanan dot adalah $[T \rightarrow \cdot F]$ maka $[T \rightarrow F \cdot]\varepsilon\ goto\ (I_0, F)$. kemudian hitung closure ($\{[T \rightarrow F \cdot]\}$) Karena dalam item baru tersebut deot sudah terletak diakhir kanan maka tidak terdapat tambahan closure dari item tersebut.

Jadi $I_3 = \{[T \rightarrow F \cdot]\}$.

Untuk X = +, item dalam I_0 dengan terminal + tepat disebelah kanan dot ternyata tidak ada, demikian juga untuk $X = *\ dan\ X =)$.

Untuk X = (, item dalam I_0 dengan terminal (tepat di sebelah kanan dot adalah $[F \rightarrow \cdot (E)]$. Maka $F \rightarrow (\cdot E)\ \varepsilon\ goto\ (I_0, ($. kemudian hitung closure ($\{[F \rightarrow (\cdot E)]\}$). Karena terdapat suatu non-terminal E tepat di sebelah kanan dot, maka tambahkan E-produksi dengan dot berada di sebelah kiri akhir yaitu E → ·E+T dan E→·T. dari item E→·T, nonterminal T tepat disebelah kanan dot, maka tambahkan lagi item-item $T \rightarrow \cdot T * F$ dan $T \rightarrow \cdot F$. Dari item $T \rightarrow \cdot F$, nonterminal F tepat disebelah kanan dot, maka tambahkan juga item-item F→·(E) dan F→·var.

Jadi I_4 = $\{[F \rightarrow (\cdot E)], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot \boldsymbol{var}]$.

Untuk X = var, item dalam I_0 dengan terminal var tepat disebelah kanan dot adalah $[F \rightarrow \cdot \boldsymbol{var}]$. maka $[F \rightarrow \cdot \boldsymbol{var}]$ ε goto(I_0, var). Kemudian hitung closure ({[*F* → ·**var**] }).

Karena dalam item baru tersebut dot sudah terletak di akhir maka tidak terdapat tambahan closure dari item tersebut.

Jadi I_5 = $\{[F \rightarrow var.]\}$.

Langkah selanjutnya adalah menghitung goto(I_1, X) untuk berbagai nilai X, yaitu $E, +, dan\ T$.

Untuk X = E, item dalam I_1 dengan non-terminal E tepat di sebelah kanan dot Ternyata tidak ada, demikian juga untuk X = T.

Untuk X = + lihat contoh 12.3. maka diperoleh:

$I_0 = \{[\ E \rightarrow E + \cdot\ T], [T \rightarrow \cdot\ T * F], [T \rightarrow \cdot\ F], [F \rightarrow \cdot\ (E)], F \rightarrow \cdot\ var.\ ]\}$.

Langkah dilanjutkan dengan menghitung goto(I_2, X) untuk berbagai nilai X, yaitu $T, *, dan\ F$.

Untuk X = T, item dalam I_2 dengan non-terminal T tepat di sebelah kanan dot Ternyata tidak ada, demikian juga untuk X = F.

untuk X = *, item dalam I_2 dengan Terminal * tepat di sebelah kanan dot adalah $[T \rightarrow T \cdot * F]$. Maka $[T \rightarrow T * \cdot\ F]$ ε goto$(I_2, *)$. kemudian hitung closure({$[T \rightarrow T * \cdot\ F]$. karena terdapat suatu nonterminal F tepat di sebelah kanan dot, maka tambahkan F-produksi dengan dot berada di sebelah kiri akhir, yaitu $F \rightarrow \cdot\ (E)$ dan $F \rightarrow \cdot\ var$.

jadi I_7 = {$[T \rightarrow T * \cdot\ F], [F \rightarrow \cdot\ (E)], [F \rightarrow \cdot\ var]$}.

Langkah dilanjutkan dengan menghitung goto(I_3, X) karena item dalam I_3 dot Sudah terletak di sebelah kanan akhir, maka goto$(I_3, X) = \{\ \}$.

selanjutnya akan dihitung goto (I_4, X) untuk berbagai nilai X, yaitu E, T, F, +, *, (,), dan var.

untuk X = +, item dalam I_4 dengan terminal + tepat di sebelah kanan dot ternyata tidak ada demikian juga untuk X = *, dan X =).

Untuk X = E, item dalam I_4 dengan non-terminal E tepat di sebelah kanan dot Adalah $[F \rightarrow (\cdot\ E)]$ dan $[E \rightarrow E + T]$. untuk $[F \rightarrow (\cdot\ E)]$ maka $F \rightarrow (E \cdot)$ ε goto(I_4, E). kemudian hitung closure ({$[F \rightarrow (E \cdot)]$}) karena dalam item tersebut dot Sudah terletak di akhir kanan maka tidak terdapat tambahan closer dari item tersebut. untuk $[E \rightarrow \cdot\ E + T]$ maka $E \rightarrow E \cdot + T\ \varepsilon$ goto(I_3, E). kemudian hitung closure({$[E \rightarrow E \cdot + T]$}) . Karena ternyata di sebelah kanan dot adalah suatu simbol terminal maka juga tidak ada tambahan closure.

Jadi $I_8 = \{[E' \rightarrow E \cdot], [E \rightarrow E \cdot + T]\}$.

Untuk X = $T\ ternyata\ goto\ (I_4, T)\ =\ goto\ (I_0, T) = I_2$.

Untuk X = $F\ ternyata\ goto(I_4, F)\ =\ goto(I_0, F) = I_3$.

Untuk X = ($ternyata\ goto(I_4, (\ ) = goto(I_0, (\ ) = I_4$.

Untuk X = $var\ ternyata\ goto(I_4, var) = goto(I_0, var) = I_5$.

Langkah dilanjutkan dengan menghitung goto(I_5, X) karena item dalam I_5 dot Sudah terletak di sebelah kanan akhir, maka goto$(I_5, X) = \{\}$.

selanjutnya akan dihitung goto(I_6, X) untuk berbagai nilai X, yaitu E, T, F, +, *, (,), dan var.

untuk X = E item dalam I_6 dengan non-terminal E tepat di sebelah kanan dot Ternyata tidak ada, demikian juga untuk $X = +, X = *, dan\ X =)$

untuk X = T, item dalam I_6 dengan non-terminal T tepat di sebelah kanan dot Adalah $[E \rightarrow E + \cdot T]$ dan $[T \rightarrow \cdot T * F]$. untuk $[E \rightarrow E + \cdot T]$ maka $[E \rightarrow E + T \cdot]$ ε goto(I_6, T). kemudian hitung closure $(\{[E \rightarrow E + T \cdot]\})$. karena dalam item baru tersebut dot Sudah terletak diakhir kanan maka tidak terdapat tambahan closure dari item tersebut. Untuk $[T \rightarrow \cdot T * F]$ maka $T \rightarrow T \cdot * F$ ε goto(I_6, T). kemudian hitung closure $(\{[T \rightarrow T \cdot * F]\})$. karena ternyata di sebelah kanan adalah suatu simbol Terminal maka juga tidak ada tambahan closure.

Jadi $I_9 = \{[E \rightarrow E + T \cdot], [T \rightarrow T \cdot * F]\}$.

untuk X = F ternyata goto$(I_6, F) =$ goto$(I_0, F) = I_3$

untuk X = (ternyata goto$(I_6, (\ ) =$ goto$(I_3, (\ ) = I_4$

untuk X = var ternyata goto$(I_6, var) =$ goto$(I_3, var) = I_5$

Langkah dilanjutkan dengan menghitung goto(I_7, X) untuk berbagai nilai X yaitu, $E, T, F, *, (,), dan\ var$.

untuk X = E dalam I_7 dengan non-terminal E tepat di sebelah kanandot Ternyata tidak ada, demikian juga untuk $X = T, X = *. dan\ X =)$.

untuk X = E, maka item dalam I_7 dengan non-terminal F tepat di sebelah kanan dot Adalah $[T \rightarrow T * \cdot]$. Maka $[T \rightarrow T * F \cdot]$ ε goto(I_7, F). kemudian hitung closure $(\{[T \rightarrow T * F \cdot]\})$. karena dalam item baru tersebut dot Sudah terletak diakhir kanan maka tidak terdapat tambahan closure dari item tersebut.

Jadi $I_{10} = \{[T \rightarrow T * F \cdot]\}$.

untuk X = (ternyata goto$(I_7, (\ ) =$ goto$(I_{10}, (\ ) = I_4$.

untuk X = var ternyata goto$(I_7, var) =$ goto$(I_{10},\ var) = I_5$.

selanjutnya akan dihitung goto(I_8, X) untuk berbagai nilai X, yaitu $E, T, +, (, dan\ )$. untuk X = E, item dalam I_8 dengan non-terminal E tepat di sebelah kanan dot Ternyata tidak ada, demikian juga untuk X = T, X = (.

Untuk X =), item dalam I_8 dengan terminal) tepat di sebelah kanan dot adalah $[F \rightarrow (E \cdot)]$. Maka $[F \rightarrow (E) \cdot]$ ε goto$(I_8,))$. Kemudian hitung closure ({[F → (E)·]}). Karena dalam item baru tersebut dot sudah terletak di akhir kanan maka tidak terdapat tambahan closure dari item tersebut.

Jadi $I_{11} = \{[F \rightarrow (E) \cdot]\}$.

untuk X = + ternyata goto$(I_8, +) =$ goto$(I_8, +) = I_6$

Langkah dilanjutkan dengan menghitung goto(I_9, X) untuk berbagai nilai X yaitu, $E, T, F, +, dan *$.

untuk X = E dalam I_9 dengan non-terminal E tepat di sebelah kanan dot Ternyata tidak ada, demikian juga untuk $X = T, X = F. dan\ X = +$.

untuk X = *, ternyata goto$(I_9, *) =$ goto$(I_2, *) = I_7$.

Langkah dilanjutkan dengan menghitung goto(I_{10}, X). Karena item dalam I_{10}, X. Karena item dalam I_{10} dot sudah terletak disebelah kanan akhir, maka goto$(I_{10}, X) = \{\}$. hal yang sama juga terjadi untuk goto(I_{11}, X).

Dengan perhitungan – perhitungan di atas, maka diperoleh koleksi kanonik dari himpunan item LR(0) untuk tatabahasa (9) sebagai berikut:

I_0	$E' \rightarrow \cdot E$	I_5	$F \rightarrow var.$
	$E \rightarrow \cdot E + F$		
	$E \rightarrow \cdot T$	I_6	$E \rightarrow E + \cdot T$
	$T \rightarrow \cdot T * F$		$T \rightarrow \cdot T * F$
	$F \rightarrow \cdot (E)$		$T \rightarrow \cdot F$
	$F \rightarrow \cdot var$		$F \rightarrow \cdot (E)$
			$F \rightarrow var$
I_1	$E' \rightarrow E \cdot$		
	$E \rightarrow E \cdot + T$	I_7	$T \rightarrow T * \cdot F$
			$F \rightarrow \cdot (E)$

I_2 $E \rightarrow T \cdot$

$T \rightarrow T \cdot * F$

I_3 $T \rightarrow F \cdot$

I_4 $F \rightarrow (\cdot E)$

$E \rightarrow \cdot E\_T$

$E \rightarrow \cdot T$

$T \rightarrow \cdot T * F$

$T \rightarrow \cdot F$

$F \rightarrow \cdot (E)$

$F \rightarrow . var.$

$F \rightarrow var$

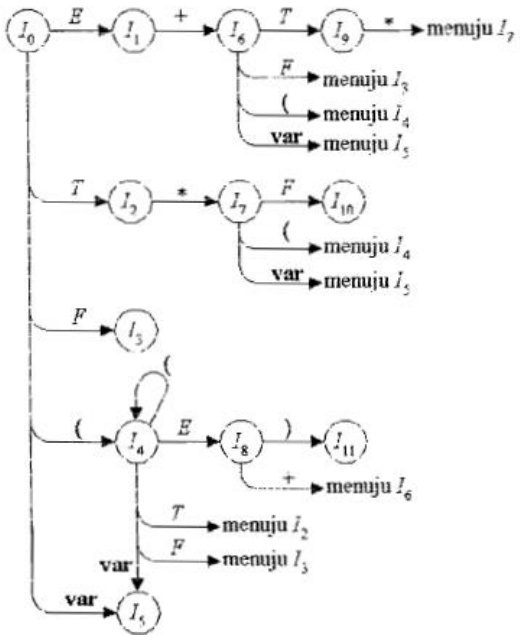
I_8 $F \rightarrow (E \cdot)$

$E \rightarrow E \cdot + T$

I_9 $E \rightarrow E + T \cdot$

$T \rightarrow T \cdot * F$

Fungsi goto dari himpunan item diatas diperlihatkan dalam diagram transisi dari dfa D seperti terlihat dalam gambar 12.3 dibawah ini.



Gambar 12.2 Diagram Transisi untuk DFA D untuk prefix-prefiks viable.

C. SOAL LATIHAN/ TUGAS

1. Jelaskan apa yang dimaksud Relasi Preseden dan Grammar Preseden Sederhana?
2. Diketahui grammar dengan $\mathbf{G} = \{\mathbf{Z} \rightarrow \mathbf{bMb}, \mathbf{M} \rightarrow (\mathbf{L} \mid \mathbf{a}, \mathbf{L} \rightarrow \mathbf{Ma})\}$.
   Dari 3 sentensial : $bab, b\,(Lb, b(Ma)b$, tentukan handel dan relasi yang ada.
3. Sebutkan Komponen dalam tabel penguraian?
4. Dalam Bottom Up Parsing apa fungsi kedua yang sangat berguna?
5. Diketahui : $G = \{W \rightarrow cNc\,, N \rightarrow (\,B|a\,, B \rightarrow Na\,)\}$
   Ditanya : Dari 3 sentensial : $cac, a\,(Bc\,, c\,(Na)\,c$, tentukan handel dan relasi yang ada

D. REFERENSI

Aho,A.V.,R.Sethi, and J. D. Ullmann, 1988. *Compiler*: Principles, Techniques, and Tools. Massachusetts: Addison Wesley Publishing Company.

Tremblay, Jean-Paull, Paul G. Sorenson, 1985. The Teory and Practice of *Compiler*, New York : McGraw-Hill Co. (Buku Pegangan Pendamping)

Sukamdi, Merekayasa *Interpreter*. 1995. (Sebuah Penerapan Teknik Kompilasi), Jakarta: PT Elex Media Komputindo. (Buku Pegangan Pendamping)

Firrar Utdiartomo, 2001. Teknik Kompilasi, Yogyakarta: J&J Learning. (Buku Pegangan Pendamping)

Sumantri Slamet, Heru,S. 1995. Teknik Kompilasi, Jakarta: PT. Elex Media Komputindo.

GLOSARIUM

Semantik adalah suatu pembelajaran arti/ makna yang terdapat pada cabang linguistic yang mengandung suatu kode, Bahasa, atau jenis representasi lain.
Ambigu adalah kemampuan mengekspresikan lebih dari satu penafsiran.
Stack adalah Tumpukan sebuah data dan biasanya di proses dari data yang terakhir.
Sentential adalah Kedudukan beberapa kata yang berada dalam sebuah kalimat, dan seperti apa pola pemakaiannya dalam kalimat tersebut.