PERTEMUAN 11

PENGURAIAN BAWAH ATAS (BOTTOM UP PARSING)

A. TUJUAN PEMBELAJARAN

Setelah pembelajaran materi untuk pertemuan 11, mahasiswa mampu menggunakan prinsip dasar kerja penguraian bawah atas (Bottom up parsing).

B. URAIAN MATERI

1. Pengertian Pengurai LR

Berbeda dengan penguraian dari atas kebawah, penguraian dari bawah ke atas adalah metode penguraian yang menggunakan pola kerja pembentukan node dari pohon urai dimulai dari daun pohon dan berlanjut ke arah akar dari pohon. Pada subbab ini akan dibahas tiga model penguraian yang menggunakan pola kerja penguraian dari bawah ke atas, yaitu penguraian shift-reduce, penguraian operator preccedence, dan penguraian LR.

2. Penguraian Shift-Reduce

Penguraian yang berusaha untuk membuat pohon urai untuk suatu masukan yang dimulai dari daun dan bergerak ke atas menuju akar. Kita dapat menganggap bahwa proses shift reduce ini sebagai proses reduksi suatu string wae menuju ke simbol awal dari suatu tatabahasa pada setiap langkah reduksi suatu substring yang sesuai dengan sisi kanan dari produksi diganti dengan simbol yang berada di sebelah kiri produksi, dan jika substring itu dipilih secara tepat pada setiap langkah, maka penurunan paling kanan di dapat diperoleh dengan melihat kebalikan proses diatas.

Contoh 11.1 Diketahui kalimat masukan abbcde, dan tatabahasa (4) berikut ini:

$S \rightarrow aabe$

$A \rightarrow Abc \mid b$

$B \rightarrow d$

Tatabahasa (4)

kalimat abbcde dapat direduksi ke S dengan langkah-langkah sebagai berikut:

Abbcde

Aabcde

Aabe

S

Penjelasannya adalah sebagai berikut:

Pertama-tama kita baca *abbcde* untuk mencari substring yang cocok dengan sisi kanan dari produksi tatabahasanya. Ternyata substring b dan d memenuhi. Kita pilih simbol b yang paling kiri dan menggantinya dengan A yaitu Sisi kiri dari produksi A → b, sehingga diperoleh sering aabcde. Kemudian perhatikan bahwa dari string tersebut, substring Abc, b, dan d sesuai dengan sisi kanan dari salah satu produksi. Walaupun b adalah substring paling kiri yang sesuai dengan sisi kanan salah satu produksi, namun kita pilih substring Abc untuk mengganti A, yaitu Sisi kiri dari produksi A → Abc sehingga diperoleh aade. Kemudian kita ganti d dengan B, yaitu Sisi kiri dari produksi B → d, sehingga diperoleh string aabe. Sekarang kita ganti rangkaian terakhir ini dengan S. Maka dengan empat urutan reduksi kita dapat mereduksi abbcde menjadi S.

dengan melihat proses reduksi di atas, dapat dibuat penurunan paling kanan yaitu dengan melihat kebalikan dari proses diatas sebagai berikut:

$$S \underset{rm}{\Rightarrow} aABe \underset{rm}{\Rightarrow} aAde \underset{rm}{\Rightarrow} aAbcde \underset{rm}{\Rightarrow} abbcde$$

a. Handel

secara informal, handel dari suatu string adalah sebuah substring yang sesuai dengan sisi kanan dari produksi, dan yang reduksinya ke non-terminal pada bagian kiri dari produksi menjadi satu langkah balik dari penurunan paling kanan. dalam banyak kasus, substring paling kiri β yang cocok dengan sisi kanan dari salah satu produksi A → β bukan sebuah handel Karena produksi dari produksi A → β menghasilkan string yang tidak dapat direduksi ke simbol awal. dalam contoh 11.1. di atas, apabila b diganti dengan A dalam string aAbcde sehingga diperoleh string aAAcde maka string ini tidak dapat direduksi ke S.

secara formal, handel dari bentuk sentential kanan γ adalah suatu produksi A → β dan suatu Posisi γ di mana string β dapat ditemukan dan diganti oleh A untuk menghasilkan bentuk sentential kanan sebelumnya dalam penurunan paling kanan dari γ. Jika $S \underset{rm}{\Rightarrow} aAw \underset{rm}{\Rightarrow} a\beta w$, maka a → β dalam posisi yang mengikuti α adalah handel dari aβw. String w di sebelah kanan handel memuat hanya simbol-simbol terminal.

dalam contoh 11.1. diatas abbcde adalah bentuk sentential kanan yang handelnya adalah A → b pada posisi 2.

Contoh 11.2., Andaikan terdapat suatu bahasa sebagai berikut:

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow (E)$$

$$E \rightarrow var$$

tatabahasa (5)

dan penurunan paling kanan sebagai berikut:

$$E \underset{rm}{\Rightarrow} \underline{E + E}$$

$$\underset{rm}{\Rightarrow} E + \underline{E * E}$$

$$\underset{rm}{\Rightarrow} E + E * \underline{var_3}$$

$$\underset{rm}{\Rightarrow} E + \underline{var_2} * var_3$$

$$\underset{rm}{\Rightarrow} \underline{var_1} + var_2 * var_3$$

keterangan subskrip pada var hanyalah untuk notasi dan garis bawah menyatakan handel dari masing-masing bentuk sentential kanan.

dalam contoh tersebut var1 adalah handel dari bentuk sentential kanan $var_1 + var_2 * var_3$ karena var adalah sisi kanan dari produksi E → var, sehingga dengan mengganti var_1 dengan E akan diperoleh bentuk sentential kanan sebelumnya, yaitu $E + var_2 * var_3$.

dari contoh diatas terlihat bahwa rangkaian yang muncul di sebelah kanan

handel hanya memuat simbol terminal.

b. Handel Prunning

Kebalikan dari penurunan paling kanan dapat diperoleh dengan handel pruning. Dimulai dengan string dari terminal w yang akan diuraikan. Jika w adalah suatu kalimat dari tatabahasa, maka w = y_{rm} dimana γ_n adalah bentuk sentential kanan dari penurunan paling kanan:

$$S = \gamma_0 \underset{rm}{\Rightarrow} \gamma_1 \underset{rm}{\Rightarrow} \gamma_2 \underset{rm}{\Rightarrow} \dots \underset{rm}{\Rightarrow} \gamma_i \underset{rm}{\Rightarrow} \gamma_n = w$$

Untuk menata ulang penurunan tersebut dalam order yang berlawanan, cari handel βn-1 dalam γn dan tempatkan βn dengan sisi kiri dari produksi An → βn untuk memperoleh sentential kanan γn-i yang ke n-1. Ulangi proses ini, yaitu cari handel βn-i dalam γn-1 dan reduksi handel tersebut untuk memperoleh bentuk sentential-kanan γn-2. Jika dengan melanjutkan proses ini bentuk sentential kanan yang dihasilkan hanya memuat simbol awal S, maka proses dihentikan dan beri pesan proses penguraian berhasil. Kebalikan dari rangkaian produksi yang dipakai dalam proses reduksi tersebut adalah penurunan paling kanan dari string yang dimasukkan.

Contoh 11.3 dengan mempergunakan tatabahasa (5) dalam contoh 11.2, dan dengan string masukan $var_1 + var_2 * var_3$ maka urutan reduksi diperlihatkan dalam Tabel 11.1 dibawah ini, yaitu reduksi dari $var_1 + var_2 * var_3$ menuju ke simbol awal E.

Tabel 11.1 Reduksi yang dibuat oleh pengurai shift-reduce.

Bentuk Sentential kanan	Handel	Reduksi produksi
$var_1 + var_2 * var_3$	var_1	$E \rightarrow var$
$E + var_2 * var_3$	var_2	$E \rightarrow var$
$E + E * var_3$	var_3	$E \rightarrow var$
$E + E * E$	$E * E$	$E \rightarrow E * E$
$E + E$	$E + E$	$E \rightarrow E + E$
E		

Dalam tabel 11.1 tersebut terlihat bahwa rangkaian dari bentuk sentential kanan adalah kebalikan dari rangkaian dalam penurunan yang paling kanan dari contoh 11.2

c. Implementasi Stack Dari Penguraian Shift-Reduce.

Terdapat dua permasalahan yang harus dipecahkan apabila pengurai mempergunakan handel prunning, yaitu :

1) Harus mencari substring yang akan direduksi dari bentuk sentential kanannya,
2) Harus menentukan pilihan produksi yang mana yang akan dipakai jika terdapat lebih dari satu pilihan.

Untuk mengatasi masalah tersebut, terlebih dahulu akan diketengahkan mengenai jenis struktur data yang akan dipakai dalam pengurai shift reduce.

Cara yang tepat untuk diimplementasikan dalam pengurai shift reduce adalah dengan mempergunakan stack untuk menyimpan simbol-simbol tatabahasa dan memakai penyangga masukan untuk menyimpan string w yang akan diurai. $ di pakai untuk menandai dasar dari stack se*gus menandai akhir kanan dari masukan. Pada awalnya stack dikosongkan, dan string w terdapat pada masukkan, yang ditunjukkan sebagai berikut:

STACK	MASUKAN
$	w $

Pengurai beroperasi dengan meletakkan nol atau lebih simbol ke dalam stack sampai suatu handel β berada di puncak dari stack. Pengurai kemudian mereduksi β menjadi sisi kiri dari produksi. Pengurai selanjutnya akan mengulangi langkah ini sampai kesalahan terdeteksi atau sampai stack memuat simbol awal dan masukannya sudah kosong yaitu:

STACK	MASUKAN
$ S	$

Setelah memperoleh konfigurasi seperti ini pengurai berhenti dan proses penguraian berhasil dilakukan.

Contoh 11.4 diberikan string masukan $\boldsymbol{var_1 + var_2 * var_3}$ dan tatabahasa (5) seperti pada contoh 11.2. diatas. Dengan mempergunakan penurunan yang pertama pada contoh 11.2. diatas, urutan langkah aksi pengurai ditunjukkan dalam tabel 11.2 berikut ini:

Tabel 11.2 Konfigurasi pengurai shift reduce untuk masukan $var_1 + var_2 * var_3$.

STACK	MASUKAN	AKSI
(1) $	$var_1 + var_2 * var_3$ $	shift
(2) $ var_1	$+var_2 * var_3$ $	reduce oleh E → var
(3) $ E	$+var_2 * var_3$ $	shift
(4) $ $E +$	$var_2 * var_3$ $	shift
(5) $ $E + var_2$	$* var_3$ $	reduce oleh E → var
(6) $ $E + E$	$* var_3$ $	shift
(7) $ $E + E *$	var_3 $	shift
(8) $ $E + E * var_3$	$	reduce oleh E → var

(9) $\$E + E * E$	$	reduce oleh E → E*E
(10) $\$E + E$	$	reduce oleh E → E+E
(6) $\$E$	$	accept

Pada penguraian terdapat dua operasi utama, yaitu shift dan reduce, sedangkan Sedangkan pada pengurai shift reduce Terdapat empat aksi yang dapat dibuat, yaitu shift, Accept, dan error.

1) pada aksi shift, simbol masukan berikutnya dipindahkan ke Puncak stack.
2) pada aksi reduce, pengurai mengetahui bahwa sisi kanan dari handel berada di puncak stack. Pengurai akan mencari sisi kiri dari handel tersebut kemudian mengganti dengan suatu non-terminal yang sesuai.
3) pada aksi accept, pengurai menyatakan bahwa penguraian telah berhasil.
4) pada aksi error pengurai menemukan kesalahan sintaks dan memanggil suatu rutin pembenahan kesalahan.

3. Penguraian Operator Precedence

Suatu tatabahasa yang sisi kanan dari produksinya tidak mengandung 20 Terminal yang berdampingan disebut tatabahasa operator.

Contoh 11.5., tatabahasa Berikut ini bukan merupakan tatabahasa operator.

$$E \rightarrow EAE \mid (E) \mid -E \mid var$$

$$A \rightarrow + \mid - \mid * \mid / \mid \uparrow$$

tatabahasa tersebut bukan merupakan tatabahasa operator karena sisi kanan produksi yang pertama EAE mempunyai tiga non-terminal yang saling berdampingan. tetapi tatabahasa tersebut dapat diubah menjadi tatabahasa operator dengan cara mengganti setiap a dengan Sisi kanannya, yaitu

menjadi:

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid E \uparrow E \mid (E) \mid -E \mid var$$

(Tatabahasa 6)

menurut sejarahnya Teknik penguraian operator precedence digambarkan sebagai suatu manipulasi pada token tanpa melihat tatabahasa utamanya. Setelah selesai membentuk pengurai operator precedence-nya tatabahasa yang asli tidak perlu dipakai, yang dipakai hanya non-terminal yang terdapat di atas stack, yaitu sebagai tempat menyimpan atribut yang berhubungan dengan non-terminalnya.

Pada penguraian operator precedence didefinisikan tiga relasi precedence yang disjoin, yaitu <·, =, ·> di antara pasangan dari terminal. Relasi precedence ini menuntun pemilihan handel yang mempunyai arti sebagai berikut:

Tabel 11.3 Relasi Precedence

Relasi	Arti Relasi
a <· b	a memberi precedence pada b
a = b	a memmpunyai precedence yang sama dengan b
a ·> b	a mengambil precedence b

Terdapat dua cara untuk menentukan precedence dari dua Terminal yaitu:

a. precedence yang didasarkan pada konsep asosiatif dan precedence dari operator. Contoh, jika * mempunyai precedence yang lebih tinggi dari +, maka + <· * dan *·> +

b. dengan terlebih dahulu membuat tatabahasa yang tidak ambigu untuk bahasanya dan kemudian dibuat mekanisme untuk membuat relasi operator precedence dari tatabahasa tadi.

4. Penggunaan Relasi Operator Precedence

Tujuan dari relasi precedence adalah untuk membatasi handel dari bentuk sentential kanan yaitu di sebelah kiri dengan <·, di tengah-tengah dengan =, dan diakhiri kanan dengan ·>. sebagai contoh, misal terdapat bentuk $var + var * var$ dan relasi precedence yang diberikan dalam tabel 11.4. dibawah ini.

maka string yang disisipi relasi Residence adalah:

Tabel 11.4 Relasi operator-precedence

$ <· var ·> + <· var ·> * <· var ·> $

	var	+	*	$
var		·>	·>	·>
+	<·	·>	<·	·>
*		·>	·>	·>
$	<·	<·	<·	

relasi <· disisipkan diantara $ dan var yang paling kiri karena <· adalah masukkan pada baris $ dan kolom var. Handel nya dapat diperoleh dengan proses berikut:

a. baca string dari kiri sampai ·> yang pertama ditemukan. pada string di atas hal ini terjadi antara file yang pertama dan +.
b. baca kebalikannya yaitu ke kiri melewati semua = sampai <· ditemukan. dalam string di atas berarti dilakukan pembacaan sampai $.
c. handel memuat semua yang berada di sebelah kiri ·> yang pertama dan sebelah kanan dari <· yang dijumpai pada Langkah 2), sehingga dari string diatas handelnya adalah var yang pertama

jika dengan mempergunakan tatabahasa (6) maka var direduksi ke E, sehingga akan diperoleh bentuk sentential kanan E+var*var. setelah mereduksi

dua var yang lain ke E, maka akan diperoleh bentuk sentential kanan E + E * E. kemudian apabila string di atas dihapus semua unsur non-terminalnya, maka yang tersisa adalah simbol $+*$. dengan menyisipkan relasi precedence maka akan diperoleh:

$$\$ <\cdot + <\cdot * \cdot> \$$$

yang mengindikasikan bahwa bagian kiri dari handel terletak di antara tanda + dan * dan bagian kanan terletak antara * dan $. relasi precedence ini mengindikasikan bahwa dalam bentuk sentential kanan E + E * E hasilnya adalah E * E disini terlihat bahwa E yang berada di sekitar * menjadi bagian dari handel.

bentuk sentential kanan harus dibaca pada setiap Langkah untuk mendapatkan handel, namun hal ini tidak akan terjadi kalau dipergunakan stack untuk menyimpan simbol masukan yang telah dilihat dipergunakan relasi precedence untuk mengarahkan aksi penguraian shift reduce. jika relasi precedence = atau $<\cdot$ terdapat di antara simbol-simbol Terminal paling atas pada stack dan simbol masukkan berikutnya, maka pengurai akan melakukan shift. dengan aksi shift ini belum dapat ditemukan bagian kanan dari handel jika relasi $\cdot>$ yang dimiliki, maka yang dilakukan adalah aksi reduce. dengan reduksi pengurai menemukan bagian kanan dari handel dan relasi precedence dapat dipergunakan menemukan bagian kiri dari handel dalam stack.

Jika tidak terdapat relasi precedence di antara pasangan-pasangan terminal (dalam tabel 11.4 dilambangkan dengan sel kosong) maka pengurai adalah menemukan adanya kesalahan sintaks dan memanggil rutin pembenahan kesalahan. algoritma berikut ini menjelaskan bagaimana hal tersebut diatas dilakukan

algoritma penguraian operator precedence adalah sebagai berikut titik sebelumnya siapkan string y dan tabel dari relasi-relasi precedence-nya mula-mula stack memuat $ dan w$ berada di dalam penyangga masukan, pengurai kemudian akan mengeksekusi program 11.1 Berikut ini:

setel ip pada symbol pertama dari w$;

repeat forever

if misalkan $ dipuncak stack dan ip menunjuk $ ***than return***

```
else begin
        misalkan a symbol terminal paling atas pada stack,
        dan b symbol yang ditunjuk oleh ip;

        if a <· b atau a = b then begin    /* shift */
                pushlah b kedalam stack;
                ganti ip dengan symbol masukan berikutnya;
end;
        else if a ·> b then                /* reduce */
                reperat
                        pop-lah symbol dalam stack
                until terminal pada stack paling atas berelasi <·
                        dengan terminal yang terakhir di-pop
                else error ()
        end
```

Program 11.1 Program penguraian Operator precedence

Setelah program di atas dijalankan, maka jika w dibentuk dengan baik, kerangka pohon urai dengan non-terminal E menjadi label semua node dalam, tetapi bila tidak, maka mengindikasikan adanya kesalahan.

Contoh 11.6., dengan mempergunakan tatabahasa (6), masukan var + var * var, dan dengan mempergunakan tabel relasi operator-precedence dari tabel 11.4. diatas, maka kerja dari pengurai digambarkan dalam tabel 11.5 berikut ini:

Tabel 11.5 Konfigurasi pengurai operator-precedence untuk masukan var + var * var.

STACK	MASUKAN	AKSI
(1) $	var + var * var $	shift
(2) $ var	+ var * var $	reduce oleh e → var

(3)	$	+ var * var $	shift
(4)	$ +	var * var $	shift
(5)	$ + var	* var $	reduce oleh e → var
(6)	$ +	*var $	shift
(7)	$ + *	var $	shift
(8)	$ + * var	$	reduce oleh e → var
(9)	$ + *	$	reduce oleh $e \rightarrow e * e$
(10)	$ +	$	reduce oleh $e \rightarrow e + e$
(11)	$	$	accept

Yang menjadi pertanyaan sekarang adalah bagaimana membuat tabel relasi operator precedence, dan mengenai hal tersebut akan dibahas pada subbab berikut ini.

5. Relasi Operator Precedence Berdasarkan Sifat Asosiatif dan Precedence

Pada prinsipnya kita diberi kebebasan untuk membuat relasi operator-precedence, namun yang terpenting harus diingat adalah bahwa algoritma penguraian operator akan bekerja dengan benar Ketika kita memakai tabel tersebut. Sebagai contoh, untuk tatabahasa pada tatabahasa (6) cara-cara berikut ini untuk membuat relasi yang benar. sebelumnya perlu diingat bahwa tatabahasa (6) ambigu, dan bentuk sentential kanan dapat mempunyai banyak handel, maka aturan yang dibuat harus dirancang untuk memilih handel yang tepat sehingga memenuhi aturan asosiatif dan precedence untuk operator-operator biner.

a. jika operator θ_1 mempunyai precedence yang lebih tinggi dibandingkan dengan operator θ_2 maka $\theta_1 \cdot> \theta_2$ dan $\theta_1 \cdot> \theta_2$ contoh jika * mempunyai precedence yang lebih tinggi dibandingkan dengan + maka * $\cdot>$ + dan $+ <$

$\cdot *$. relasi ini memberi kepastian dalam ekspresi $E + E * E + E$ yaitu Bahwa $E * E$ adalah handel yang harus direduksi terlebih dulu.

b. jika θ_1 dan θ_2 adalah operator yang mempunyai precedence sama, maka tetapkan $\theta_1 \cdot> \theta_2$ dan $\theta_2 \cdot> \theta_1$ apabila operator-operator asosiatif kiri, atau $\theta_1 <\cdot \theta_2$ dan $\theta_2 <\cdot \theta_1$ apabila operator-operator yang asosiatif kanan. sebagai contoh jika + dan - asosiatif kiri, maka + ·> +, + ·> -, - ·> -, dan - · > +. Relasi ini memberi kepastian bahwa $E - E + E$ akan mempunyai handel $E - E$, dan untuk $E \uparrow E \uparrow E$ maka $E \uparrow E$ yang terakhir adalah handelnya, karena operator ↑ bersifat asosiatif kanan.

c. tetapkan $\theta <\cdot var, var \cdot> \theta$, $\theta <\cdot (, (<\cdot \theta,) \cdot> \theta, \theta \cdot>), \theta \cdot> \$$, dan $\$ <\cdot \theta$ untuk semua operator θ Demikian juga tetapkan:

(=)	$ <· (	$ <· var
(<· (	var ·> $	) ·> $
(<· var	var ·>)	) ·>)

Aturan ini akan memastikan bahwa var dan (X) akan direduksi menjadi Selain itu karena $ juga berfungsi sebagai akhir kiri dan kanan maka menyebabkan handel-handel akan ditemukan diantara dua $ sebagai contoh dari tabel relasi dapat dilihat dalam tabel 11.4 diatas.

C. SOAL LATIHAN/ TUGAS

1. Sebuah grammar dikatakan bersifat *left recursion* apabila *grammar* tersebut mengandung?
2. Sebutkan sebuah grammar yang tidak dapat menggunakan grammar rekursi kiri?
3. Buatlah Contoh Program penguraian Operator precedence?
4. Sebutkan model penguraian yang menggunakan pola kerja penguraian dari bawah ke atas?
5. Diketahui $S \rightarrow aB \mid bA, A \rightarrow a \mid aS \mid bAA, B \rightarrow b \mid bS \mid aBB$

   Penurunan untuk string **aaabbabbba** Berikan solusi untuk derivasi dan analisis sintaksnya!

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

Operator precedence adalah urutan evaluasi dimana operator yang ada di suatu ekspresi akan dievaluasi berdasarkan aturan prioritas yang ditentukan.

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.