LR(0) PARSER

Sebuah produksi disebut LR (0) apabila A→ a yang dipilih dan jika β dan γ adalah dua symbol string (termasuk empty) dimana $\beta\gamma = a$ maka A→ $\beta.\gamma$. (tidak ada acuan secara eksplisit untuk menunu string yang didepan “lookahead”)

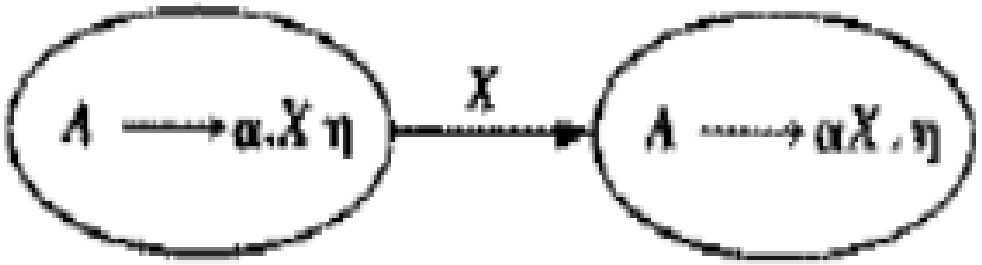
$$S' \rightarrow S$$
$$S \rightarrow (S) S \mid \varepsilon$$

Grammar tersebut memiliki 3 produksi yang dipilih dan 8 item :

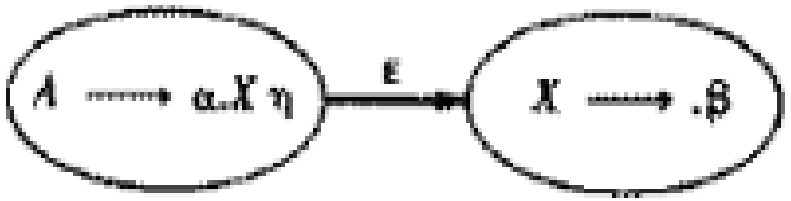
$$S' \rightarrow .S$$
$$S' \rightarrow S.$$
$$S \rightarrow .(S)S$$
$$S \rightarrow (.S)S$$
$$S \rightarrow (S.)S$$
$$S \rightarrow (S).S$$
$$S \rightarrow (S)S.$$
$$S \rightarrow .$$

LR (0) PARSER terjadi apabila tidak “lookahead” yang dapat diperiksa setelah muncul parsing di stack, dan jika ini terjadi maka tidak dapat dihitung sebagai string.

Transisi pada LR(0) digunakan untuk menunjukkan state-state yang menginformasikan stack parsing dan proses dari shift dan reduksi pada parser. Yaitu apabila $A \rightarrow \alpha.\beta$ dan γ dimulai dengan symbol X, yang mana X mungkin adalah sebuah token dan nonterminal maka dapat ditulis $A \rightarrow a.Xn$ sehingga terdapat transisi pada symbol X yaitu :



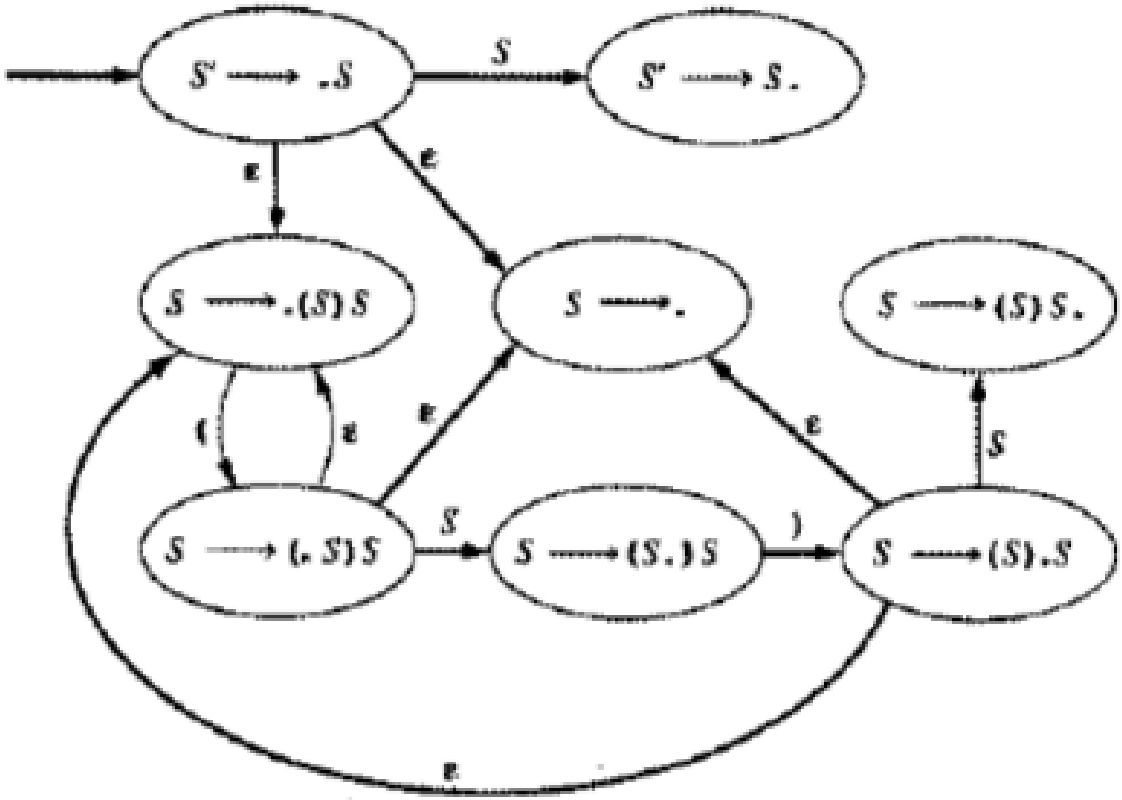
Dan jika X adalah sebuah token, maka transisi merupakan shift dari X dimana input paling atas stack merupakan sebuah parsing. Pada kasus lain jika X adalah sebuah nonterminal, maka push X kedalam stack dimana ini hanya terjadi apabila $X \rightarrow \beta$. Karena adanya β dan state $X \rightarrow \beta$ maka proses di mulai, dimana setiap $A \rightarrow a.Xn$ harus ditambah sebuah transisi ε



Perhatikan contoh soal :

S'→S
S→(S)S|

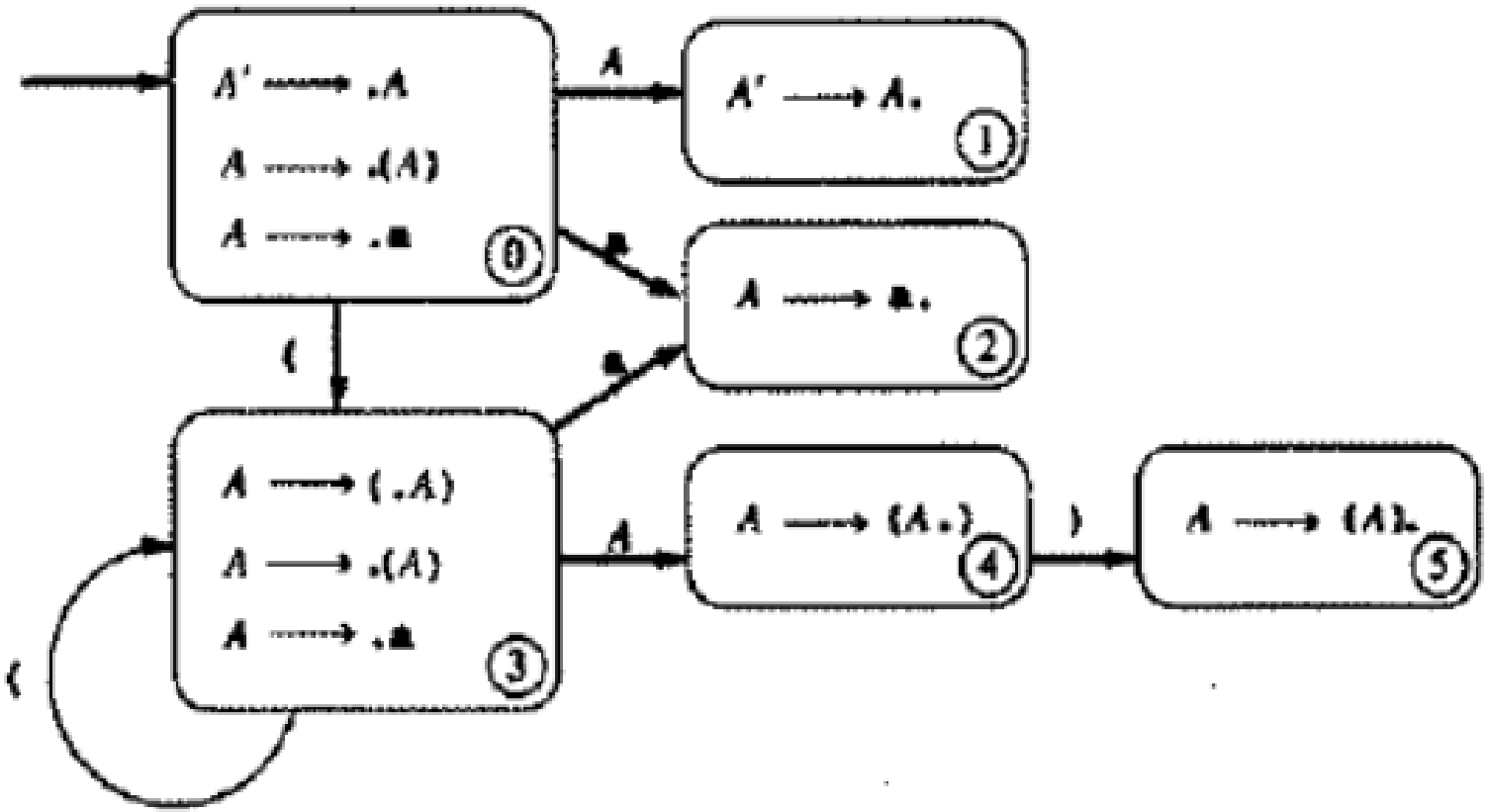
Transisi dari grammar diatas :



Algoritma LR (0) untuk table parse

1. Jika state S berisi$\boldsymbol{A \rightarrow a.X\beta}$, dimana X adalah terminal, maka pop string dari stack jika token adalah X dan state S berisi $A \rightarrow a.X\beta$, maka state baru di **push** kedalam stack yang berisi $A \rightarrow aX.\beta$ jika token bukan X untuk beberapa item di dalam stack S maka berikan penjelasan error
2. Jika state memiliki item lengkap $(A \rightarrow \gamma.)$ maka langkah berikutnya $A \rightarrow \gamma$. Reduksi aturan $S' \rightarrow S$, dimana S adalah state awal yang diterima, memberikan input kosong dan pesan kesalahan apabila input tidak kosong.

Transisi Grammar :



Tabel LL (0) parser dengan string " ((a)) "

State	Action	Rule	Input			Goto
			(	a	)	A
0	shift		3	2		1
1	reduce	$A' \rightarrow A$				
2	reduce	$A \rightarrow a$				
3	shift		3	2		4
4	shift				5	
5	reduce	$A \rightarrow (A)$				

Tabel Action parser dengan string "((a))"

	Parsing stack	Input	Action
1	$ 0	((a)) $	shift
2	$ 0 (3	(a)) $	shift
3	$ 0 (3 (3	a)) $	shift
4	$ 0 (3 (3 a 2	)) $	reduce $A \rightarrow a$
5	$ 0 (3 (3 A 4	)) $	shift
6	$ 0 (3 (3 A 4) 5	) $	reduce $A \rightarrow (A)$
7	$ 0 (3 A 4	) $	shift
8	$ 0 (3 A 4) 5	$	reduce $A \rightarrow (A)$
9	$ 0 A 1	$	accept

Latihan :

1. Tentukan Tabel LR 0 dan Tabel Parsing Action dari CFG berikut :

   S → AA

   A → aA | b

   Dengan string “aabb$”

2. Buatlah LR (0) Parser pada *grammar* dibawah ini, serta buktikan bahwa string **“ cbbad”**

S → BAC | ADd

A → ba | ed

B → cb

C → d

D→ fa