Linguagens Formais e Autómatos / Compiladores

Análise sintática ascendente

Artur Pereira <artur@ua.pt>,
Miguel Oliveira e Silva <mos@ua.pt>

DETI, Universidade de Aveiro

Sumário

Análise sintática ascendente
Ilustração por um exemplo

- Considere a gramática

$$
\begin{aligned}
D &\rightarrow T\ L\ \texttt{;} \\
T &\rightarrow \texttt{i} \mid \texttt{r} \\
L &\rightarrow \texttt{v} \mid L\ \texttt{,}\ \texttt{v}
\end{aligned}
$$

  que representa uma declaração de variáveis *a la* C
- Como reconhecer a palavra "$u = \texttt{i v , v ;}$" como pertencente à linguagem definida pela gramática dada?
- Se u pertence à linguagem definida pela gramática, então $D \Rightarrow^{+} u$
- Gerando uma derivação à direita, tem-se

$$D \Rightarrow T\,L\,\texttt{;} \Rightarrow T\,L\,\texttt{,}\,\texttt{v;} \Rightarrow T\,\texttt{v,v;} \Rightarrow \texttt{i v,v;}$$

- Tente-se agora fazer a derivação no sentido contrário, isto é indo de u para D

Análise sintática ascendente
Ilustração por um exemplo (cont.)

- Considere a gramática

$$
\begin{array}{l}
D \to T\ L\ \texttt{;} \\
T \to \texttt{i} \mid \texttt{r} \\
L \to \texttt{v} \mid L\ \texttt{,}\ \texttt{v}
\end{array}
$$

  e *reduza-se* a palavra "$u = \texttt{i v , v ;}$" a D

- $$
\begin{array}{ll}
 & \texttt{i v , v ;} \\
\Leftarrow & T\ \texttt{v , v ;} \quad (\text{por aplicação da produção } T \to \texttt{i}) \\
\Leftarrow & T\ L\ \texttt{, v ;} \quad (\text{por aplicação da produção } L \to \texttt{v}) \\
\Leftarrow & T\ L\ \texttt{;} \quad (\text{por aplicação da produção } L \to L\ \texttt{,}\ \texttt{v}) \\
\Leftarrow & D \quad (\text{por aplicação da produção } D \to T\ L\ \texttt{;})
\end{array}
$$

- Colocando ao contrário, tem-se

$$D \Rightarrow T\,L\,\texttt{;} \Rightarrow T\,L\,\texttt{,}\,\texttt{v}\,\texttt{;} \Rightarrow T\,\texttt{v}\,\texttt{,}\,\texttt{v}\,\texttt{;} \Rightarrow \texttt{i}\,\texttt{v}\,\texttt{,}\,\texttt{v}\,\texttt{;}$$

  vê-se que corresponde à derivação à direita

Análise sintática ascendente
Ilustração por um exemplo (cont.)

- A tabela seguinte mostra como, na prática, se realiza esta (retro)derivação

pilha	entrada	próxima ação
	i v , v ; $	deslocamento
i	v , v ; $	redução por $T \to$ i
T	v , v ; $	deslocamento
T v	, v ; $	redução por $L \to$ v
$T\,L$	, v ; $	deslocamento
$T\,L$,	v ; $	deslocamento
$T\,L$, v	; $	redução por $L \to L$, v
$T\,L$	; $	deslocamento
$T\,L$;	$	redução por $D \to T\,L$;
D	$	deslocamento
D $		aceitação

- A palavra à entrada foi reduzida ao símbolo inicial pelo que é aceite como pertencendo à linguagem

Análise sintática ascendente
Ilustração de um erro sintático

- Veja-se a reação deste procedimento a uma entrada errada, por exemplo a palavra `i v v ;`.

pilha	entrada	próxima ação
	`i v v ;` $\$$	deslocamento
`i`	`v v ;` $\$$	redução por $T \to$ `i`
T	`v v ;` $\$$	deslocamento
T `v`	`v ;` $\$$	redução por $L \to$ `v`
$T\,L$	`v ;` $\$$	deslocamento
$T\,L$ `v`	`;` $\$$	rejeição

- Rejeita porque L `v` não corresponde ao prefixo de uma produção da gramática
- Na realidade, o erro poderia ter sido detetado dois passos antes, aquando da segunda redução, porque `v` $\notin$ **`follow`**(L)
  - `v` corresponde ao símbolo à entrada
  - L é o símbolo que iria aparecer no topo da pilha se se fizesse a redução por $L \to$ `v`

Análise sintática ascendente
Ilustração de conflito entre deslocamento e redução

- Considere a gramática

$$S \to \texttt{i c } S \mid \texttt{i c } S \texttt{ e } S \mid \texttt{a}$$

e aplique-se o procedimento anterior à palavra `i c i c a e a`

pilha	entrada	próxima ação
	`i c i c a e a` $	deslocamento
`i`	`c i c a e a` $	deslocamento
`i c`	`i c a e a` $	deslocamento
`i c i`	`c a e a` $	deslocamento
`i c i c`	`a e a` $	deslocamento
`i c i c a`	`e a` $	redução por $S \to$ `a`
`i c i c` S	`e a` $	**conflito**:
		– redução por $S \to$ `i c` S
		– deslocamento para tentar $S \to$ `i c` S `e` S

- Esta gramática não faz lembrar uma estrutura típica em linguagens de programação?

Análise sintática ascendente
Ilustração de conflito entre reduções

- Considere a gramática

$$
\begin{array}{rcl}
S & \to & A \\
 & | & B \\
A & \to & \mathtt{c} \\
 & | & A\ \mathtt{a} \\
B & \to & \mathtt{c} \\
 & | & B\ \mathtt{b}
\end{array}
$$

e aplique-se o procedimento anterior à palavra $\mathtt{c}$

pilha	**entrada**	**próxima ação**
	$\mathtt{c}\ \$$	deslocamento
$\mathtt{c}$	$\$$	**conflito:**
		– redução usando $A \to \mathtt{c}$
		– redução usando $B \to \mathtt{c}$

Análise sintática ascendente
Ilustração de falso conflito

- Considere a gramática

$$
\begin{array}{rcl}
S & \rightarrow & \texttt{a} \\
 & | & < S > \\
 & | & \texttt{a}\ P \\
 & | & < S > S \\
P & \rightarrow & < S > \\
 & | & < S > S
\end{array}
$$

  e aplique-se o procedimento de reconhecimento à palavra $\texttt{a} < \texttt{a} > \texttt{a}$

pilha	**entrada**	**próxima ação**
	$\texttt{a} < \texttt{a} > \texttt{a}\ \$$	deslocamento
$\texttt{a}$	$< \texttt{a} > \texttt{a}\ \$$	falso conflito:
		– redução usando $S \rightarrow \texttt{a}$
		– deslocamento para tentar $S \rightarrow \texttt{a}\ P$

- Deslocamento, porque se se optasse pela redução no topo da pilha ficaria um S e $< \notin \texttt{follow}\,(S)$

Análise sintática ascendente
Ilustração de falso conflito (cont.)

- Optando pelo deslocamento e continuando...

pilha	entrada	próxima ação
	$\mathtt{a} < \mathtt{a} > \mathtt{a}\ \$$	deslocamento
$\mathtt{a}$	$< \mathtt{a} > \mathtt{a}\ \$$	deslocamento, porque $< \notin \mathbf{follow}(S)$
$\mathtt{a} <$	$\mathtt{a} > \mathtt{a}\ \$$	deslocamento
$\mathtt{a} < \mathtt{a}$	$> \mathtt{a}\ \$$	redução por $S \to \mathtt{a}$
$\mathtt{a} < S$	$> \mathtt{a}\ \$$	deslocamento
$\mathtt{a} < S >$	$\mathtt{a}\ \$$	deslocamento, porque $\mathtt{a} \notin \mathbf{follow}(P)$
$\mathtt{a} < S > \mathtt{a}$	$\$$	redução por $S \to \mathtt{a}$
$\mathtt{a} < S > S$	$\$$	redução por $P \to < S > S$
$\mathtt{a}\ P$	$\$$	redução por $S \to \mathtt{a}\ P$
S	$\$$	deslocamento
$S\ \$$		aceitação

Análise sintática ascendente
Eliminação de conflito

- Pode ser possível alterar uma gramática de modo a eliminar a fonte de conflito
- Considerando que se pretendia optar pelo deslocamento, a gramática da esquerda gera a mesma linguagem que a da direita e está isenta de conflitos.

$$
\begin{array}{rcl}
S & \rightarrow & \texttt{a} \\
 & | & \texttt{i c } S \\
 & | & \texttt{i c } S' \texttt{ e } S \\
S' & \rightarrow & \texttt{a} \\
 & | & \texttt{i c } S' \texttt{ e } S'
\end{array}
\qquad
\left|
\qquad
\begin{array}{rcl}
S & \rightarrow & \texttt{a} \\
 & | & \texttt{i c } S \\
 & | & \texttt{i c } S \texttt{ e } S
\end{array}
\right.
$$

Análise sintática ascendente
`if..then..else` sem conflitos

- Considere a gramática seguinte e processe-se a palavra `i c i c a e a`

$$S \to \texttt{a} \mid \texttt{i c } S \mid \texttt{i c } S' \texttt{ e } S$$
$$S' \to \texttt{a} \mid \texttt{i c } S' \texttt{ e } S'$$

pilha	entrada	próxima ação
	`i c i c a e a` $	deslocamento
`i`	`c i c a e a` $	deslocamento
`i c`	`i c a e a` $	deslocamento
`i c i`	`c a e a` $	deslocamento
`i c i c`	`a e a` $	deslocamento
`i c i c a`	`e a` $	redução por $S' \to$ `a`
`i c i c` S'	`e a` $	deslocamento
`i c i c` S' `e`	`a` $	deslocamento
`i c i c` S' `e a`	$	redução por $S \to$ `a`
`i c i c` S' `e` S	$	redução por $S \to$ `i c` S' `e` S
`i c` S	$	redução por $S \to$ `i c` S
S	$	deslocamento e aceitação

- Se houver capacidade de *lookahead*, pode optar-se por $S' \to$ `a`, porque `e` $\in$ **follow** (S') mas `e` $\notin$ **follow** (S)

Construção de um reconhecedor ascendente
Abordagem

- Como determinar de forma sistemática a ação a realizar (deslocamento, redução, aceitação, rejeição)?

pilha	**entrada**	**próxima ação**
	`i v v ; $`	deslocamento
`i`	`v v ; $`	redução por $T \rightarrow$ `i`
T	`v v ; $`	deslocamento
T `v`	`v ; $`	rejeição

- A ação a realizar em cada passo do procedimento de reconhecimento – deslocamento, redução, aceitação ou rejeição – depende da configuração em cada momento
- Uma **configuração** é formada pelo conteúdo da pilha mais a parte da entrada ainda não processada
- A pilha é conhecida – na realidade, é preenchida pelo procedimento de reconhecimento
- Da entrada, em cada momento, apenas se conhece o *lookahead*

Construção de um reconhecedor ascendente
Abordagem (cont.)

pilha	entrada	próxima ação
	i v v ; $	deslocamento
i	v v ; $	redução por $T \rightarrow$ i
T	v v ; $	deslocamento
T v	v ; $	rejeição

- Quantos símbolos da pilha usar?
- Poder-se-á usar apenas um?
- Se se quiser e puder construir um reconhecedor que apenas use o símbolo no topo, uma pilha onde se guardam os símbolos terminais e não terminais tem pouco interesse
- Mas pode definir-se um alfabeto adequado para a pilha
- Os símbolos a colocar na pilha devem representar estados no processo de deslocamento/redução/aceitação
- Por exemplo, um dado símbolo pode significar que, na produção $D \rightarrow T\ L$;, já se processou algo que corresponde ao $T\ L$, faltando o ;

Construção de um reconhecedor ascendente
Itens de uma gramática

- O alfabeto da pilha representa assim o conjunto de estados nesse processo de reconhecimento
- Cada estado representa um conjunto de itens
- Cada item representa o quanto de uma produção já foi processado e o quanto ainda falta processar
- A produção $A \rightarrow B_1 \ B_2 \ B_3$ introduz 4 itens:

$$A \rightarrow \cdot B_1 \ B_2 \ B_3$$
$$A \rightarrow B_1 \cdot B_2 \ B_3$$
$$A \rightarrow B_1 \ B_2 \cdot B_3$$
$$A \rightarrow B_1 \ B_2 \ B_3 \cdot$$

- A produção $A \rightarrow \varepsilon$ introduz um único item:

$$A \rightarrow \cdot$$

Conjunto dos conjuntos de itens
Ilustração com um exemplo

- Considere a gramática

$$S \rightarrow E$$
$$E \rightarrow \texttt{a} \mid \texttt{(}\ E\ \texttt{)}$$

- Reconhecer a palavra $u = u_1 u_2 \cdots u_n$, significa reduzir $u\$$ a $S\$$
- Então, o estado inicial pode ser definido por

$$Z_0 = \{S \rightarrow \cdot\, E\, \$\}$$

- O facto de o ponto ($\cdot$) se encontrar imediatamente à esquerda de um símbolo não terminal, significa que para se avançar no processo de reconhecimento é preciso obter esse símbolo
- Isso é considerado juntando ao conjunto Z_0 os itens iniciais das produções cuja cabeça é E

$$Z_0 = \{\ S \rightarrow \cdot\, E\, \$\ \} \cup \{\ E \rightarrow \cdot\, \texttt{a}\ ,\ E \rightarrow \cdot\, \texttt{(}\ E\ \texttt{)}\ \}$$

- Esta operação é designada de fecho (**closure**)

Conjunto dos conjuntos de itens
Ilustração com um exemplo (cont.)

- Evolução de Z_0:

$$Z_0 = \{ S \to \cdot\, E\, \$ \} \cup \{ E \to \cdot\, \mathtt{a} \,,\; E \to \cdot\, (\; E\;) \}$$

- O estado Z_0 pode evoluir por ocorrência de um E, um `a` ou um `(`, que correspondem aos símbolos que aparecem imediatamente à direita do ponto ($\cdot$)

$$Z_1 = \delta(Z_0, E) = \{ S \to E \cdot \$ \}$$
$$Z_2 = \delta(Z_0, \mathtt{a}) = \{ E \to \mathtt{a} \cdot \}$$
$$Z_3 = \delta(Z_0, \mathtt{(}) = \{ E \to (\; \cdot\, E\;) \}$$

- Z_3 tem de ser estendido pela função de fecho, uma vez que o ponto ($\cdot$) ficou imediatamente à esquerda de um símbolo não terminal (E)

$$Z_3 = \delta(Z_0, \mathtt{(}) = \{ E \to (\; \cdot\, E\;) \} \cup \{ E \to \cdot\, \mathtt{a} \,,\; E \to \cdot\, (\; E\;) \}$$

- Z_2 representa uma situação terminal, passível de redução pela regra $E \to \mathtt{a}$

Conjunto dos conjuntos de itens
Ilustração com um exemplo (cont.)

- Evolução de Z_1:

$$Z_1 = \delta(Z_0, E) = \{\, S \to E \cdot \$ \,\}$$

- Apenas evolui por ocorrência de um $\$$

$$Z_4 = \delta(Z_1, \$) = \{\, S \to E \,\$ \cdot \,\}$$

- Evolução de Z_3:

$$Z_3 = \delta(Z_0, \texttt{(}) = \{\, E \to \texttt{(} \cdot E \,\texttt{)} \,\} \cup \{\, E \to \cdot \texttt{a} \,,\; E \to \cdot \texttt{(}\; E \,\texttt{)} \,\}$$

- Pode evoluir por ocorrência de um E, um $\texttt{a}$ ou um $\texttt{(}$

$$Z_5 = \delta(Z_3, E) = \{\, E \to \texttt{(}\; E \cdot \texttt{)} \,\}$$
$$\delta(Z_3, \texttt{a}) = \{\, E \to \texttt{a} \cdot \,\} = Z_2$$
$$\delta(Z_3, \texttt{(}) = \{\, E \to \texttt{(} \cdot E \,\texttt{)} \,\} = Z_3$$

- A evolução de Z_3 com $\texttt{a}$ e $\texttt{(}$ dá origem a elementos já obtidos anteriormente

Conjunto dos conjuntos de itens
Ilustração com um exemplo (cont.)

- Z_4 representa uma situação terminal, correspondente à aceitação

$$Z_4 = \delta(Z_1, \$) = \{\ S \rightarrow E\,\$\,\cdot\ \}$$

- Evolução de Z_5

$$Z_5 = \delta(Z_3, E) = \{\ E \rightarrow (\ E\ \cdot\ )\ \}$$

- Apenas evolui por ocorrência de)

$$Z_6 = \delta(Z_4,)) = \{\ E \rightarrow (\ E\ )\ \cdot\ \}$$

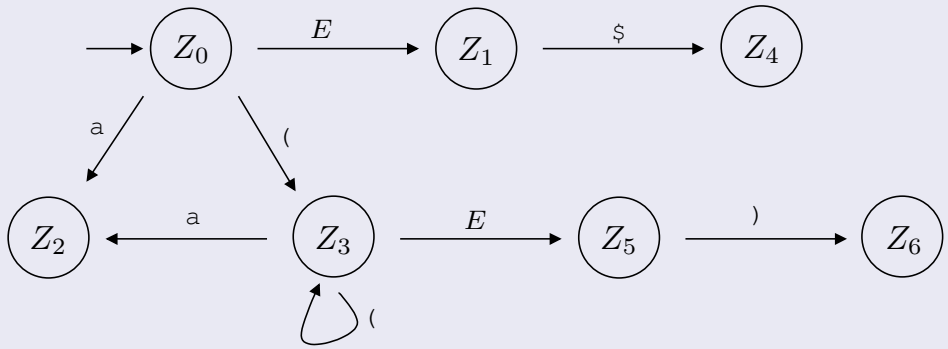
- Z_6 representa uma situação terminal, passível de redução pela regra $E \rightarrow (\ E\ )$

Conjunto dos conjuntos de itens
Ilustração com um exemplo (cont.)

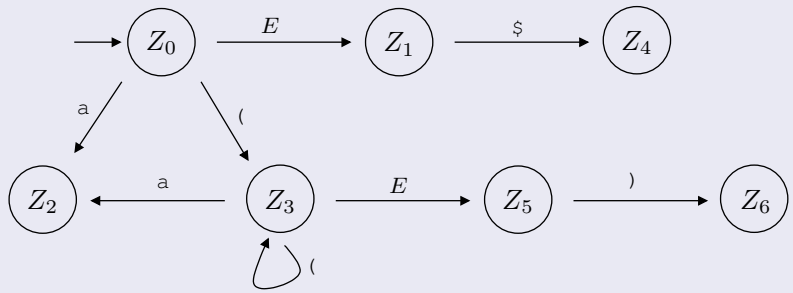
- Pondo tudo junto

$$
\begin{aligned}
Z_0 &= \{\, S \to \cdot\, E\,\$ \,\} \cup \{\, E \to \cdot\, \mathtt{a}\,,\; E \to \cdot\, (\; E\;) \,\} \\
Z_1 &= \delta(Z_0, E) = \{\, S \to E \cdot \$ \,\} \\
Z_2 &= \delta(Z_0, \mathtt{a}) = \{\, E \to \mathtt{a} \cdot \,\} \\
Z_3 &= \delta(Z_0, () = \{\, E \to (\; \cdot\, E\;) \,\} \cup \{\, E \to \cdot\, \mathtt{a}\,,\; E \to \cdot\, (\; E\;) \,\} \\
Z_4 &= \delta(Z_1, \$) = \{\, S \to E\,\$ \cdot \,\} \\
Z_5 &= \delta(Z_3, E) = \{\, E \to (\; E \cdot\;) \,\} \\
Z_6 &= \delta(Z_5,)) = \{\, E \to (\; E\;) \cdot \,\}
\end{aligned}
$$

- Representando na forma de um autómato, tem-se

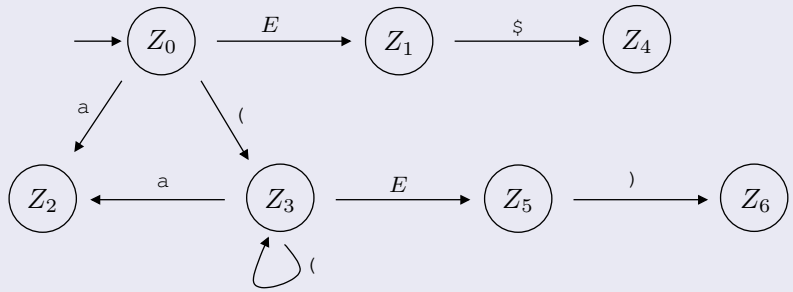


Conjunto dos conjuntos de itens
Ilustração com um exemplo (cont.)



- Neste autómato, os estados representam o alfabeto da pilha
- As transições representam operações de *push*
- As transições etiquetadas com símbolos terminais representam adicionalmente ações de deslocamento (*shift*)
- As ações de redução provocam operações de *pop*, em número igual ao número de elementos do corpo da produção
- As transições etiquetadas com símbolos não terminais ocorrem após as ações de redução
- Tudo isto representa o funcionamento de um autómato de pilha que permite fazer o reconhecimento da linguagem

Tabela de análise de um reconhecedor ascendente
Introdução



- O autómato de pilha pode ser implementado usando uma tabela de análise
- Esta tabela contém duas matrizes, ACTION e GOTO
  - as linhas de ambas são indexadas pelo alfabeto da pilha (conjunto de conjuntos de itens)
- A matriz ACTION representa ações
  - as colunas são indexadas pelos símbolos terminais da gramática, incluindo o marcador de fim de entrada ($)
  - As células contêm as ações *shift*, *reduce*, *accept* ou *error*
  - No caso de *shift*, também inclui o próximo símbolo a colocar na pilha
- A matriz GOTO representa a operação após uma redução
  - as colunas são indexadas pelos símbolos não terminais da gramática
  - As células indicam que valor colocar na *stack* após uma ação de redução

Tabela de análise de um reconhecedor ascendente
Exemplo

- Ao conjunto de conjunto de itens obtidos anteriormente

$$
\begin{aligned}
Z_0 &= \{\, S \to \cdot\, E\,\$ \,\} \cup \{\, E \to \cdot\, \mathtt{a}\ ,\ E \to \cdot\, (\ E\ ) \,\} \\
Z_1 &= \delta(Z_0, E) = \{\, S \to E \cdot \$ \,\} \\
Z_2 &= \delta(Z_0, \mathtt{a}) = \{\, E \to \mathtt{a} \cdot \,\} \\
Z_3 &= \delta(Z_0, () = \{\, E \to (\ \cdot\, E\ ) \,\} \cup \{\, E \to \cdot\, \mathtt{a}\ ,\ E \to \cdot\, (\ E\ ) \,\} \\
Z_4 &= \delta(Z_1, \$) = \{\, S \to E\,\$ \cdot \,\} \\
Z_5 &= \delta(Z_3, E) = \{\, E \to (\ E \cdot\ ) \,\} \\
Z_6 &= \delta(Z_5,)) = \{\, E \to (\ E\ ) \cdot \,\}
\end{aligned}
$$

- Corresponde a tabela

	a	(	)	$	E
Z_0	shift, Z_2	shift, Z_3			Z_1
Z_1				Z_4	
Z_2			reduce, $E \to \mathtt{a}$	reduce, $E \to \mathtt{a}$	
Z_3	shift, Z_2	shift, Z_3			Z_5
Z_4	accept				
Z_5			shift, Z_6		
Z_6			reduce, $E \to (\ E\ )$	reduce, $E \to (\ E\ )$	

- As células vazias representam situações de erro sintático

Reconhecedor ascendente
Algoritmo de reconhecimento

- Com base na tabela de análise, o procedimento de reconhecimento pode ser implementado pelo seguinte algoritmo

```
push(Z_0)
forever
    if top() == Z_4
        ACCEPT
    action = table[top(),lookahead()]
    if action is (shift, Z_i)
        adv(); push(Z_i);
    else if action is (reduce A → α)
        pop |α| símbolos; push(table[top(),A]);
    else
        REJECT
```

Tabela de análise de um reconhecedor ascendente
Exemplo #2

	a	(	)	$	E
Z_0	shift, Z_2	shift, Z_3			Z_1
Z_1				Z_4	
Z_2			reduce, $E \to$ a	reduce, $E \to$ a	
Z_3	shift, Z_2	shift, Z_3			Z_5
Z_4	accept				
Z_5			shift, Z_6		
Z_6			reduce, $E \to$ (E)	reduce, $E \to$ (E)	

- Aplicando este algoritmo à palavra ((a))

pilha	entrada	próxima ação
Z_0	((a)) $	shift, Z_3
$Z_0\ Z_3$	(a)) $	shift, Z_3
$Z_0\ Z_3\ Z_3$	a)) $	shift, Z_2
$Z_0\ Z_3\ Z_3\ Z_2$	)) $	reduce $E \to$ a (1 *pop*)
$Z_0\ Z_3\ Z_3$		*push* Z_5
$Z_0\ Z_3\ Z_3\ Z_5$	)) $	shift, Z_6
$Z_0\ Z_3\ Z_3\ Z_4\ Z_6$	) $	reduce $E \to$ (E) (3 *pops*)
$Z_0\ Z_3$		*push* Z_5
$Z_0\ Z_3\ Z_5$	) $	shift, Z_6
$Z_0\ Z_3\ Z_4\ Z_6$	$	reduce $E \to$ (E) (3 *pops*)
Z_0		*push* Z_1
$Z_0\ Z_1$	$	shift, Z_4
$Z_0\ Z_1\ Z_4$		accept

Tabela de análise de um reconhecedor ascendente
Exemplo #3

- Determine-se a tabela de análise para um reconhecedor ascendente com *lookahead* 1 da gramática seguinte

$$
\begin{aligned}
S &\rightarrow \mathtt{a} \mid (\ S\ ) \mid \mathtt{a}\,P \mid (\ S\ )\ S \\
P &\rightarrow (\ S\ ) \mid (\ S\ )\ S
\end{aligned}
$$

- O primeiro passo corresponde a alterar a gramática de modo ao símbolo inicial não aparecer do lado direito

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow \mathtt{a} \mid (\ S\ ) \mid \mathtt{a}\,P \mid (\ S\ )\ S \\
P &\rightarrow (\ S\ ) \mid (\ S\ )\ S
\end{aligned}
$$

Tabela de análise de um reconhecedor ascendente
Exemplo #3 (cont.)

- O passo seguinte corresponde a calcular o conjunto de conjunto de itens

$$
\begin{aligned}
Z_0 &= \{S_0 \to \cdot\, S\,\$\,\} \\
&\cup \{S \to \cdot\,\mathtt{a}\,, S \to \cdot\,(\,S\,)\,, S \to \cdot\,\mathtt{a}\,P\,, S \to \cdot\,(\,S\,)\,S\,\} \\
Z_1 &= \delta(Z_0, \mathtt{a}) = \{S \to \mathtt{a}\,\cdot\,, S \to \mathtt{a}\,\cdot\,P\,\} \\
&\cup \{P \to \cdot\,(\,S\,)\,, P \to \cdot\,(\,S\,)\,S\,\} \\
Z_2 &= \delta(Z_0, () = \{S \to (\,\cdot\,S\,)\,, S \to (\,\cdot\,S\,)\,S\,\} \\
&\cup \{S \to \cdot\,\mathtt{a}\,, S \to \cdot\,(\,S\,)\,, S \to \cdot\,\mathtt{a}\,P\,, S \to \cdot\,(\,S\,)\,S\,\} \\
Z_3 &= \delta(Z_0, S) = \{S_0 \to S\,\cdot\,\$\,\} \\
Z_4 &= \delta(Z_1, () = \{P \to (\,\cdot\,S\,)\,, P \to (\,\cdot\,S\,)\,S\,\} \\
&\cup \{S \to \cdot\,\mathtt{a}\,, S \to \cdot\,(\,S\,)\,, S \to \cdot\,\mathtt{a}\,P\,, S \to \cdot\,(\,S\,)\,S\,\} \\
Z_5 &= \delta(Z_1, P) = \{S \to \mathtt{a}\,P\,\cdot\,\} \\
Z_6 &= \delta(Z_2, S) = \{S \to (\,S\,\cdot\,)\,, S \to (\,S\,\cdot\,)\,S\,\} \\
Z_7 &= \delta(Z_3, \$) = \{S_0 \to S\,\$\,\cdot\,\} \\
Z_8 &= \delta(Z_4, S) = \{P \to (\,S\,\cdot\,)\,, P \to (\,S\,\cdot\,)\,S\,\} \\
Z_9 &= \delta(Z_6,)) = \{S \to (\,S\,)\cdot\,, S \to (\,S\,)\,\cdot\,S\,\} \\
&\cup \{S \to \cdot\,\mathtt{a}\,, S \to \cdot\,(\,S\,)\,, S \to \cdot\,\mathtt{a}\,P\,, S \to \cdot\,(\,S\,)\,S\,\} \\
Z_{10} &= \delta(Z_8,)) = \{P \to (\,S\,)\cdot\,, P \to (\,S\,)\,\cdot\,S\,\} \\
&\cup \{S \to \cdot\,\mathtt{a}\,, S \to \cdot\,(\,S\,)\,, S \to \cdot\,\mathtt{a}\,P\,, S \to \cdot\,(\,S\,)\,S\,\} \\
Z_{11} &= \delta(Z_9, S) = \{S \to (\,S\,)\,S\,\cdot\,\} \\
Z_{12} &= \delta(Z_{10}, S) = \{P \to (\,S\,)\,S\,\cdot\,\}
\end{aligned}
$$

- Note que $\delta(Z_2, \mathtt{a}) = \delta(Z_4, \mathtt{a}) = \delta(Z_9, \mathtt{a}) = \delta(Z_{10}, \mathtt{a}) = Z_1$ e $\delta(Z_2, () = \delta(Z_4, () = \delta(Z_9, () = \delta(Z_{10}, () = Z_2$

Tabela de análise de um reconhecedor ascendente
Exemplo #3 (cont.)

- E finalmente a tabela de análise

	a	(	)	$	S	P
Z_0	*shift*, Z_1	*shift*, Z_2			Z_3	
Z_1		*shift*, Z_4	*reduce* $S \rightarrow$ a	*reduce* $S \rightarrow$ a		Z_5
Z_2	*shift*, Z_1	*shift*, Z_2			Z_6	
Z_3				*shift*, Z_7		
Z_4	*shift*, Z_1	*shift*, Z_2			Z_8	
Z_5			*reduce* $S \rightarrow$ a P	*reduce* $S \rightarrow$ a P		
Z_6			*shift*, Z_9			
Z_7	*accept*	*accept*	*accept*	*accept*		
Z_8			*shift*, Z_{10}			
Z_9	*shift*, Z_1	*shift*, Z_2	*reduce* $S \rightarrow$ (S)	*reduce* $S \rightarrow$ (S)	Z_{11}	
Z_{10}	*shift*, Z_1	*shift*, Z_2	*reduce* $P \rightarrow$ (S)	*reduce* $P \rightarrow$ (S)	Z_{12}	
Z_{11}			*reduce* $S \rightarrow$ (S) S	*reduce* $S \rightarrow$ (S) S		
Z_{12}			*reduce* $P \rightarrow$ (S) S	*reduce* $P \rightarrow$ (S) S		

Tabela de análise de um reconhecedor ascendente
Exercício

- Determine-se a tabela de análise para um reconhecedor ascendente com *lookahead* 1 da gramática seguinte

$$S \rightarrow \varepsilon \mid S\ B\ \mathtt{a} \mid S\ A\ \mathtt{b}$$
$$A \rightarrow \mathtt{a} \mid A\ A\ \mathtt{b}$$
$$B \rightarrow B\ B\ \mathtt{a} \mid \mathtt{b}$$