

Construção de compiladores

Profs. Mário César San Felice (e Helena Caseli,
Murilo Naldi, Daniel Lucrédio)

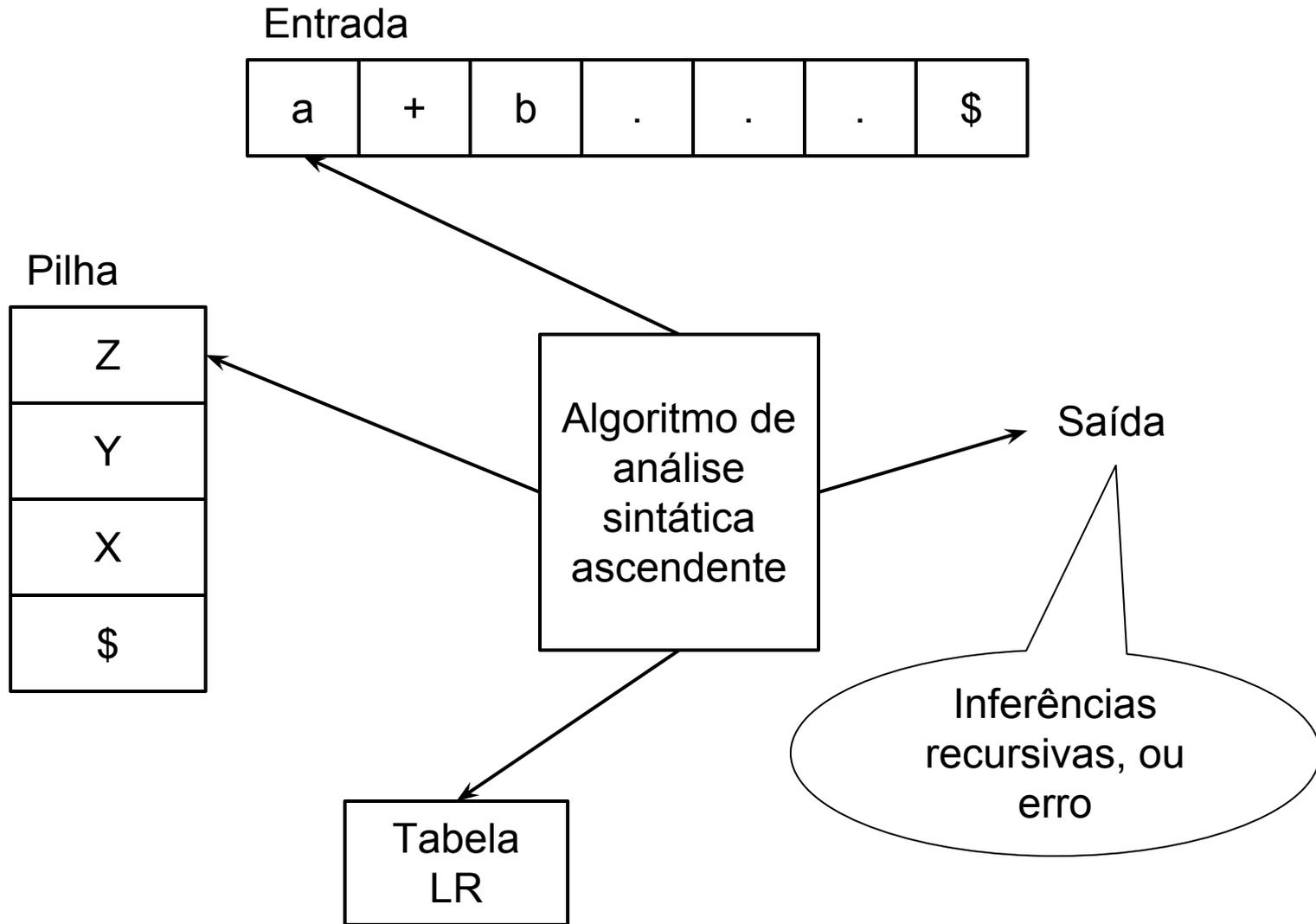
Departamento de Computação - UFSCar

1º semestre / 2018

Tópico 6 - Análise LR

Análise sintática ascendente LR

ASA



ASA LR

- Analisador LR (k)
 - Left to right
 - Lê a sentença em análise da esquerda para a direita
 - Rightmost derivation
 - Produz uma derivação mais à direita ao reverso
 - Inferência recursiva
 - Considerando-se k símbolos na cadeia de entrada

Tabela de análise LR

- Existem diferentes tipos de tabelas LR
 - Cada uma com vantagens/desvantagens
- A tabela LR é dividida em duas
 - Ação
 - Transição
- A tabela é construída diretamente a partir da gramática
- Estados = armazenam a situação atual de leitura
 - Permitem detectar o aparecimento de um “gancho”

Exemplo de tabela de análise LR

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Tabela de análise LR

- Os códigos para as ações são:
 - $si = \textit{shift } i$
 - “avança na entrada e empilha o estado i na pilha”
 - $rj = \textit{reduce } j$
 - “reduz segundo a produção de número j ”
 - OK
 - “aceita a entrada”
 - Entrada em branco
 - Erro sintático

Algoritmo de análise LR

- ENTRADAS:
 - Uma cadeia de entrada w
 - Uma tabela de análise LR com as ações e transições definidas para uma gramática G
- SAÍDA:
 - Se w está em $L(G)$, a saída são os passos de inferência recursiva (análise ascendente para w)
 - Caso contrário, a saída é uma indicação de erro
- CONDIÇÕES INICIAIS:
 - $w\$$ no buffer de entrada
 - estado 0 na pilha (estado inicial)

Algoritmo de análise LR

```
a := primeiro símbolo de w$;
while(1) { /* repita indefinidamente */
  s := estado no topo da pilha;
  if(ACAO[s,a] = "shift t") {
    empilha t;
    a := próximo símbolo da entrada;
  } else if (ACAO[s,a] = "reduce A -->  $\beta$ ") {
    desempilha  $|\beta|$  itens;
    t := topo da pilha;
    empilha TRANSICAO[t,A];
    imprima "A -->  $\beta$ "
  } else if (ACAO[s,a] = "OK") pare; /* fim */
  else erro;
}
```


Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0	T	*id+id\$	

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0	E	+id\$	

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6	E + F	\$	

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6 3	E + F	\$	r4

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6 3	E + F	\$	r4
0 1 6	E + T	\$	

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6 3	E + F	\$	r4
0 1 6 9	E + T	\$	r1

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6 3	E + F	\$	r4
0 1 6 9	E + T	\$	r1
0	E	\$	

Exemplo

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * id + id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*id+id\$	s5
0 5	id	*id+id\$	r6
0 3	F	*id+id\$	r4
0 2	T	*id+id\$	s7
0 2 7	T *	id+id\$	s5
0 2 7 5	T * id	+id\$	r6
0 2 7 10	T * F	+id\$	r3
0 2	T	+id\$	r2
0 1	E	+id\$	s6
0 1 6	E +	id\$	s5
0 1 6 5	E + id	\$	r6
0 1 6 3	E + F	\$	r4
0 1 6 9	E + T	\$	r1
0 1	E	\$	OK

Exercício

Entrada = id * (id + id)

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*(id+id)\$	s5
0 5	id	*(id+id)\$	r6
0 3	F	*(id+id)\$	r4
0 2	T	*(id+id)\$	s7
0 2 7	T *	(id+id)\$	s4
0 2 7 4	T * (id+id)\$	s5
0 2 7 4 5	T * (id	+id)\$	r6
0 2 7 4 3	T * (F	+id)\$	r4
0 2 7 4 2	T * (T	+id)\$	r2
0 2 7 4 8	T * (E	+id)\$	s6
0 2 7 4 8 6	T * (E +	id)\$	s5
0 2 7 4 8 6 5	T * (E + id)\$	r6
0 2 7 4 8 6 3	T * (E + F)\$	r4
0 2 7 4 8 6 9	T * (E + T)\$	r1
0 2 7 4 8	T * (E)\$	s11
0 2 7 4 8 11	T * (E)	\$	r5
0 2 7 10	T * F	\$	r3
0 2	T	\$	r2
0 1	E	\$	OK

Exercício

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Entrada = id * (id

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Pilha	Símbolos	Entrada	Ação
0		id*(id\$	s5
0 5	id	*(id\$	r6
0 3	F	*(id\$	r4
0 2	T	*(id\$	s7
0 2 7	T *	(id\$	s4
0 2 7 4	T * (id\$	s5
0 2 7 4 5	T * (id	\$	r6
0 2 7 4 3	T * (F	\$	r4
0 2 7 4 2	T * (T	\$	r2
0 2 7 4 8	T * (E	\$	erro

Análise LR

- 3 tipos de tabela (o algoritmo é o mesmo):
 - Simple LR (SLR)
 - Fácil de implementar
 - Aplicável a uma classe mais restrita de gramáticas
 - Look Ahead LR (LALR)
 - Nível intermediário e implementação eficiente
 - Funciona para a maioria das linguagens de programação
 - LR Canônico
 - Mais poderoso e complexo
 - Pode ser aplicado a um grande número de gramáticas

Tabela SLR

- A construção da tabela SLR se baseia na coleção canônica de conjuntos de itens LR(0)
 - 0 porque não se olha qualquer símbolo a frente
- Um item LR(0) para uma gramática G é uma produção com alguma indicação (.) de até onde essa produção já foi analisada no processo de reconhecimento

Tabela SLR

- Ex: a produção $A \rightarrow XYZ$ dá origem a 4 itens LR(0).
 - $A \rightarrow \cdot XYZ$
 - $A \rightarrow X \cdot YZ$
 - $A \rightarrow XY \cdot Z$
 - $A \rightarrow XYZ \cdot$
- Produções do tipo $A \rightarrow \varepsilon$ geram somente um item $A \rightarrow \cdot$.

Tabela SLR

- Ex: a produção $A \rightarrow XYZ$ dá origem a 4 itens LR(0).

- $A \rightarrow .XYZ$

- $A \rightarrow X.YZ$

- $A \rightarrow XY.Z$

- $A \rightarrow XYZ.$

**Item
inicial**

**Item
completo**

- Produções do tipo $A \rightarrow .$ somente um item $A \rightarrow .$

Itens LR(0)

- Exercícios:
 - encontre os itens LR(0) para as seguintes gramáticas

- **G1:**

$$S' \rightarrow S$$

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

- **G2:**

$$E' \rightarrow E$$

$$E \rightarrow E + n \mid n$$

Itens LR(0)

- Respostas:

- G1:

$$S' \rightarrow S$$

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

$$S' \rightarrow \cdot S$$

$$S' \rightarrow S \cdot$$

$$S \rightarrow \cdot (S) S$$

$$S \rightarrow (\cdot S) S$$

$$S \rightarrow (S \cdot) S$$

$$S \rightarrow (S) \cdot S$$

$$S \rightarrow (S) S \cdot$$

$$S \rightarrow \cdot$$

- G2:

$$E' \rightarrow E$$

$$E \rightarrow E + n \mid n$$

$$E' \rightarrow \cdot E$$

$$E' \rightarrow E \cdot$$

$$E \rightarrow \cdot E + n$$

$$E \rightarrow E \cdot + n$$

$$E \rightarrow E + \cdot n$$

$$E \rightarrow E + n \cdot$$

$$E \rightarrow \cdot n$$

$$E \rightarrow n \cdot$$

Fecho de conjuntos de itens

- Função CLOSURE(I)
 - I é um conjunto de itens para uma gramática G
- Regras:
 1. Inicialmente, acrescente todo item de I em CLOSURE(I)
 2. Se $A \rightarrow \alpha . B \beta$ está em CLOSURE(I) e $B \rightarrow \gamma$ é uma produção, então adicione o item $B \rightarrow . \gamma$ em CLOSURE(I), se ainda não estiver lá
 - Aplique essa regra até que nenhum outro item possa ser incluído no CLOSURE(I)

Fecho de conjuntos de itens

- Algoritmo:

```
SetOfItems CLOSURE (I) {  
    J := I;  
    repeat  
        for (cada item "A→α.Bβ" em J)  
            for (cada produção "B→γ" de G)  
                if ("B→.γ" não está em J)  
                    adicione "B→.γ" em J;  
    until nenhum item seja adicionado a  
        J em um passo do loop;  
    return J;  
}
```

Fecho de conjuntos de itens

- Exemplo:

$E' \rightarrow E$

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

- $CLOSURE(\{ [E' \rightarrow \cdot E] \}) = \{ [E' \rightarrow \cdot E], \dots$

Fecho de conjuntos de itens

- Exemplo:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $CLOSURE(\{ [E' \rightarrow \cdot E] \}) = \{ [E' \rightarrow \cdot E],$
 $[E \rightarrow \cdot E+T], [E \rightarrow \cdot T],$
 $[T \rightarrow \cdot T*F], [T \rightarrow \cdot F],$
 $[F \rightarrow \cdot (E)], [F \rightarrow \cdot id]$
 $\}$

Fecho de conjuntos de itens

- Exercício:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T^*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $CLOSURE(\{ [E \rightarrow \cdot E+T] \}) =$

Fecho de conjuntos de itens

- Exercício:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $CLOSURE(\{ [E \rightarrow \cdot E+T] \}) = \{ [E \rightarrow \cdot E+T],$
 $[E \rightarrow \cdot T], [T \rightarrow \cdot T*F],$
 $[T \rightarrow \cdot F], [F \rightarrow \cdot (E)],$
 $[F \rightarrow \cdot id]$
 $\}$

Fecho de conjuntos de itens

- Exercício :

$$E' \rightarrow E$$

$$E \rightarrow E+T \quad | \quad T$$

$$T \rightarrow T*F \quad | \quad F$$

$$F \rightarrow (E) \quad | \quad id$$

- $CLOSURE(\{ [T \rightarrow T \cdot *F] \}) =$

Fecho de conjuntos de itens

- Exercício :

$$E' \rightarrow E$$
$$E \rightarrow E+T \quad | \quad T$$
$$T \rightarrow T*F \quad | \quad F$$
$$F \rightarrow (E) \quad | \quad id$$

- $CLOSURE(\{ [T \rightarrow T \cdot *F] \}) = \{ [T \rightarrow T \cdot *F] \}$

Fecho de conjuntos de itens

- Exercício :

$$E' \rightarrow E$$
$$E \rightarrow E+T \quad | \quad T$$
$$T \rightarrow T^*F \quad | \quad F$$
$$F \rightarrow (E) \quad | \quad id$$

- $CLOSURE(\{ [T \rightarrow T^* \cdot F] \}) =$

Fecho de conjuntos de itens

- Exercício :

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T^*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $CLOSURE(\{ [T \rightarrow T^* \cdot F] \}) = \{ [T \rightarrow T^* \cdot F],$
 $[F \rightarrow \cdot (E)],$
 $[F \rightarrow \cdot id]$
 $\}$

Função de transição

- Função GOTO(I,X)
 - I é um conjunto de itens
 - X é um símbolo da gramática
- Regra
 - GOTO(I,X) é o fecho do conjunto de todos os itens $[A \rightarrow \alpha X \cdot \beta]$ tais que $[A \rightarrow \alpha \cdot X \beta]$ está em I

Função de transição

- Regra
 - $GOTO(I, X)$ é o fecho do conjunto de todos os itens $[A \rightarrow \alpha X \cdot \beta]$ tais que $[A \rightarrow \alpha \cdot X \beta]$ está em I
- Em outras palavras:
 1. Examinamos os itens em I com ponto imediatamente à esquerda de X
 2. Para cada item encontrado no passo 1
 - a. Movemos o ponto para imediatamente à direita de X
 - b. Calculamos o fecho desse conjunto

Função de transição

- Exemplo:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $I = \{ [E' \rightarrow E \cdot], [E \rightarrow E \cdot + T] \}$

- $GOTO(I, +) =$

Função de transição

- Exemplo:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $I = \{ [E' \rightarrow E \cdot], [E \rightarrow E \cdot +T] \}$

- $GOTO(I, +) = \{ [E \rightarrow E+ \cdot T], [T \rightarrow \cdot T*F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot id] \}$

Função de transição

- Exercício:

$$E' \rightarrow E$$

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow (E) \mid id$$

- $I = \{ [F \rightarrow \cdot (E)], [F \rightarrow (E \cdot)], [F \rightarrow \cdot id] \}$

- $GOTO(I,) =$

- $GOTO(I, id) =$

- $GOTO(I, () =$

Função de transição

- Exercício:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

- $I = \{ [F \rightarrow \cdot (E)], [F \rightarrow (E \cdot)], [F \rightarrow \cdot id] \}$
- $GOTO(I,) = \{ [F \rightarrow (E) \cdot] \}$
- $GOTO(I, id) = \{ [F \rightarrow id \cdot] \}$
- $GOTO(I, () = \{ [F \rightarrow (\cdot E)], [E \rightarrow \cdot E+T], [E \rightarrow \cdot T], [T \rightarrow \cdot T*F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot id] \}$

Coleção canônica de conjuntos de itens LR(0)

- Veremos agora um algoritmo para, dada uma gramática G :
 - Obter itens LR(0) para G
 - Agrupar os itens em conjuntos
 - Agrupar os conjuntos em uma coleção
- Essa coleção será utilizada para construir
 - Um autômato LR(0)
- Esse autômato é a base do analisador sintático SLR
 - A tabela SLR é montada a partir desse autômato

Coleção canônica de conjuntos de itens LR(0)

- Passo 1:
 - Dada uma gramática G
 - Criar uma gramática aumentada G'
 - G' consiste de G com uma produção adicional inicial $S' \rightarrow S$
 - onde S é o símbolo inicial de G
 - S' não ocorre em G
- Exemplo:

$$\begin{array}{l} E \rightarrow E+T \quad | \quad T \\ T \rightarrow T * F \quad | \quad F \\ F \rightarrow (E) \quad | \quad id \end{array}$$

Coleção canônica de conjuntos de itens LR(0)

- Passo 1:
 - Dada uma gramática G
 - Criar uma gramática aumentada G'
 - G' consiste de G com uma produção adicional inicial $S' \rightarrow S$
 - onde S é o símbolo inicial de G
 - S' não ocorre em G
- Exemplo:

$E' \rightarrow E$

$E \rightarrow E+T \quad | \quad T$

$T \rightarrow T * F \quad | \quad F$

$F \rightarrow (E) \quad | \quad id$

**E' é o novo
símbolo
inicial**

Coleção canônica de conjuntos de itens LR(0)

- Passo 1:
 - Dada uma gramática G
 - Criar uma gramática aumentada G'
 - G' consiste de G com uma nova produção inicial $S' \rightarrow S$
 - onde S é o símbolo inicial
 - S' não ocorre em G
- Exemplo:

$E' \rightarrow E$

$E \rightarrow E+T \quad | \quad T$
 $T \rightarrow T * F \quad | \quad F$
 $F \rightarrow (E) \quad | \quad id$

Na gramática aumentada tem-se a certeza de que o símbolo inicial tem somente uma produção associada

Dessa forma é mais fácil detectar o fim da análise sintática. Mais precisamente, o item $s' \rightarrow s .$ marca o fim da análise

Coleção canônica de conjuntos de itens LR(0)

- Passo 2 (algoritmo)

```
void itens(G') {  
    C := { CLOSURE ({ [S' → .S] }) };  
    repeat  
        (a) for (cada conjunto de itens I em C)  
            (b) for (cada símbolo de gramática X)  
                if (GOTO (I, X) não é vazio e não  
                    está em C)  
                    adicione GOTO (I, X) em C;  
    until nenhum novo conjunto de itens  
        é adicionado a C em uma rodada;  
}
```

Coleção canônica de conjuntos de itens LR(0)

- Exemplo:

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

$$C = \text{CLOSURE}(\{[E' \rightarrow \cdot E]\}) = \left\{ \begin{array}{l} [E' \rightarrow \cdot E], \\ [E \rightarrow \cdot E+T], \\ [E \rightarrow \cdot T], \\ [T \rightarrow \cdot T*F], \\ [T \rightarrow \cdot F], \\ [F \rightarrow \cdot (E)], \\ [F \rightarrow \cdot id] \end{array} \right\} = I_0$$

Coleção canônica de conjuntos de itens LR(0)

$$C = \{ I_0 \}$$

$$a: I_0 = \left\{ \begin{array}{l} [E' \rightarrow \cdot E], \\ [E \rightarrow \cdot E + T], \\ [E \rightarrow \cdot T], \\ [T \rightarrow \cdot T * F], \\ [T \rightarrow \cdot F], \\ [F \rightarrow \cdot (E)], \\ [F \rightarrow \cdot id] \end{array} \right\}$$

$$b: \begin{array}{l} \text{GOTO}(I_0, E') = \emptyset \\ \text{GOTO}(I_0, E) = \{ [E' \rightarrow E \cdot], [E \rightarrow E \cdot + T] \} = I_1 \\ \text{GOTO}(I_0, T) = \{ [E \rightarrow T \cdot], [T \rightarrow T \cdot * F] \} = I_2 \\ \text{GOTO}(I_0, F) = \{ [T \rightarrow F \cdot] \} = I_3 \\ \text{GOTO}(I_0, +) = \emptyset \\ \text{GOTO}(I_0, *) = \emptyset \\ \text{GOTO}(I_0, () = \{ [F \rightarrow (\cdot E)], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], \\ [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], \\ [F \rightarrow \cdot id] \} = I_4 \end{array}$$

$$\text{GOTO}(I_0,) = \emptyset$$

$$\text{GOTO}(I_0, id) = \{ [F \rightarrow id \cdot] \} = I_5$$

Coleção canônica de conjuntos de itens LR(0)

$$C = \{ I_0, I_1, I_2, I_3, I_4, I_5 \}$$

$$a: I_1 = \left\{ \begin{array}{l} [E' \rightarrow E.] \\ [E \rightarrow E.+T] \end{array} \right\}$$

$$b: \begin{array}{l} \text{GOTO}(I_1, E') = \emptyset \\ \text{GOTO}(I_1, E) = \emptyset \\ \text{GOTO}(I_1, T) = \emptyset \\ \text{GOTO}(I_1, F) = \emptyset \\ \text{GOTO}(I_1, +) = \{ [E \rightarrow E+.T], [T \rightarrow .T \times F], \\ [T \rightarrow .F], [F \rightarrow .(E)], \\ [F \rightarrow .id] \} = I_6 \end{array}$$

$$\text{GOTO}(I_1, x) = \emptyset$$

$$\text{GOTO}(I_1, () = \emptyset$$

$$\text{GOTO}(I_1,) = \emptyset$$

$$\text{GOTO}(I_1, id) = \emptyset$$

Coleção canônica de conjuntos de itens LR(0)

$$a: I_2 = \left\{ \begin{array}{l} [E \rightarrow T.] \\ [T \rightarrow T.*F] \end{array} \right\}$$

$$b: \text{GOTO}(I_2, *) = \left\{ \begin{array}{l} [T \rightarrow T*.F], \\ [F \rightarrow .(E)], \\ [F \rightarrow .id] \end{array} \right\} = I_7$$

$$a: I_3 = \{ [T \rightarrow F.] \}$$

b: todos GOTO são vazios

Coleção canônica de conjuntos de itens LR(0)

a: $I_4 = \{$
 $[F \rightarrow (.E)],$
 $[E \rightarrow .E+T],$
 $[E \rightarrow .T],$
 $[T \rightarrow .T * F],$
 $[T \rightarrow .F],$
 $[F \rightarrow .(E)],$
 $[F \rightarrow .id]$
 $\}$

b: $GOTO(I_4, E) = \{ [F \rightarrow (E.)], [E \rightarrow E.+T] \} = I_8$

$GOTO(I_4, T) = \{ [E \rightarrow T.], [T \rightarrow T.*F] \} = I_2$

$GOTO(I_4, F) = \{ [T \rightarrow F.] \} = I_3$

$GOTO(I_4, () = \{ [F \rightarrow (.E)], [E \rightarrow .E+T], [E \rightarrow .T],$
 $[T \rightarrow .T * F], [T \rightarrow .F], [F \rightarrow .(E)],$
 $[F \rightarrow .id] \} = I_4$

$GOTO(I_4, id) = \{ [F \rightarrow id.] \} = I_5$

Coleção canônica de conjuntos de itens LR(0)

a: $I_5 = \{ [F \rightarrow id.] \}$

b: todos GOTO são vazios

Agora, $C = \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8 \}$

Coleção canônica de conjuntos de itens LR(0)

$$a: I_6 = \left\{ \begin{array}{l} [E \rightarrow E + \cdot T], \\ [T \rightarrow \cdot T * F], \\ [T \rightarrow \cdot F], \\ [F \rightarrow \cdot (E)], \\ [F \rightarrow \cdot id] \end{array} \right\}$$

$$b: \text{GOTO}(I_6, T) = \left\{ [E \rightarrow E + T \cdot], [T \rightarrow T \cdot * F] \right\} = I_9$$

$$\text{GOTO}(I_6, F) = \left\{ [T \rightarrow F \cdot] \right\} = I_3$$

$$\text{GOTO}(I_6, () = \left\{ [F \rightarrow (\cdot E)], \dots \right\} = I_4$$

$$\text{GOTO}(I_6, id) = \left\{ [F \rightarrow id \cdot] \right\} = I_5$$

Coleção canônica de conjuntos de itens LR(0)

$$a: I_7 = \left\{ \begin{array}{l} [T \rightarrow T * . F], \\ [F \rightarrow . (E)], \\ [F \rightarrow . id] \end{array} \right\}$$

$$b: \text{GOTO}(I_7, F) = \{ [T \rightarrow T * F .] \} = I_{10}$$
$$\text{GOTO}(I_7, () = \{ [F \rightarrow (. E)], \dots \} = I_4$$
$$\text{GOTO}(I_7, id) = \{ [F \rightarrow id .] \} = I_5$$

Coleção canônica de conjuntos de itens LR(0)

$$a: I_8 = \left\{ \begin{array}{l} [F \rightarrow (E.)], \\ [E \rightarrow E.+T] \end{array} \right\}$$

$$b: \text{GOTO}(I_8, ()) = \{ [F \rightarrow (E).] \} = I_{11}$$

$$\text{GOTO}(I_8, +) = \{ [E \rightarrow E+.T], \dots \} = I_6$$

$$\text{Agora, } C = \{ I_0, \dots, I_8, I_9, I_{10}, I_{11} \}$$

Coleção canônica de conjuntos de itens LR(0)

$$a: I_9 = \left\{ \begin{array}{l} [E \rightarrow E + T.] \\ [T \rightarrow T * F] \end{array} \right\}$$

$$b: \text{GOTO}(I_9, *) = \{ [T \rightarrow T * F.] \dots \} = I_7$$

$$a: I_{10} = \{ [T \rightarrow T * F.] \}$$

b: todos GOTO são vazios

$$a: I_{11} = \{ [F \rightarrow (E).] \}$$

b: todos GOTO são vazios

Coleção canônica de conjuntos de itens LR(0)

- Neste momento, nenhum novo conjunto de itens é adicionado
 - Fim do algoritmo
- Podemos agora montar o autômato LR(0)
 - Esse autômato captura as possíveis transições entre estados de reconhecimento parcial
- Cada conjunto de itens LR(0) é um estado
- As transições são dadas pela função GOTO

Autômato LR(0)

- Coleção canônica

$$I_0 = \{ [E' \rightarrow \cdot E], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot id] \}$$

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

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

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

$$I_4 = \{ [F \rightarrow (\cdot E)], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot id] \}$$

$$I_5 = \{ [F \rightarrow id \cdot] \}$$

$$I_6 = \{ [E \rightarrow E + \cdot T], [T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)], [F \rightarrow \cdot id] \}$$

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

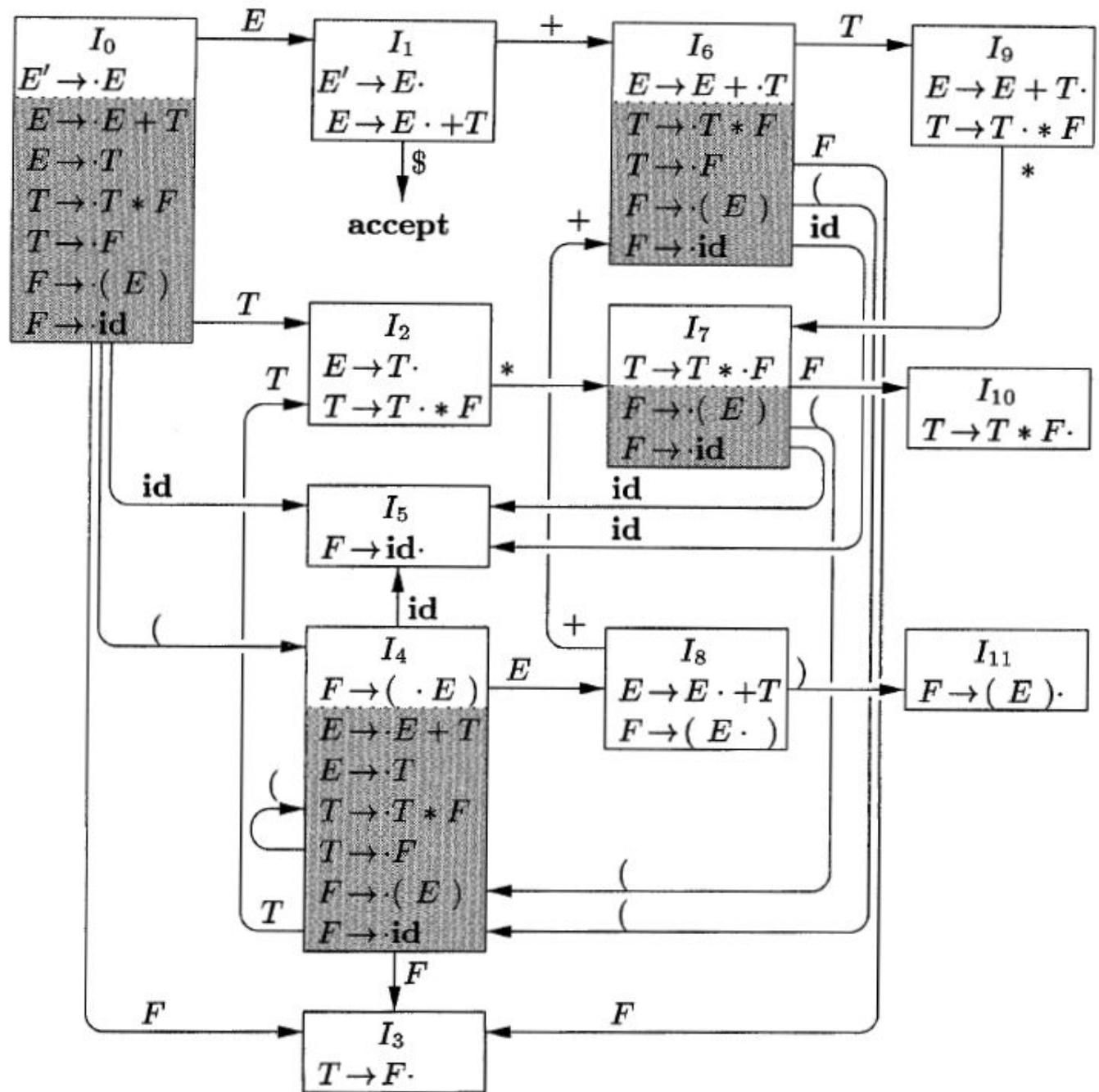
$$I_8 = \{ [F \rightarrow (E \cdot)], [E \rightarrow E \cdot + T] \}$$

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

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

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

Autômato LR(0)



Do autômato LR(0) para a tabela SLR

- O autômato LR(0) mostra o que acontece quando
 - Algum símbolo da entrada é consumido
 - Representam as ações do tipo “shift”
 - Estados da pilha são reduzidos
 - Correspondem aos estados da tabela Transições
- Para o analisador SLR, é necessário definir também os movimentos do tipo “reduce”
 - Existe um outro algoritmo para isso
 - Mas é preciso conhecer o conjunto $SEGUIDORES(A)$ para cada não-terminal A de uma gramática

Do autômato LR(0) para a tabela SLR

- Exemplo (relembrando):

$$E' \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

primeiros (E') = { (, id }

primeiros (E) = { (, id }

primeiros (T) = { (, id }

primeiros (F) = { (, id }

seguidores (E') = { \$ }

seguidores (E) = { +,), \$ }

seguidores (T) = { +,), *, \$ }

seguidores (F) = { +,), *, \$ }

Do autômato LR(0) para a tabela SLR

- Em seguida, vamos atribuir um índice para cada regra de produção (com exceção da regra aumentada)

$$E' \rightarrow E$$

$$(1) E \rightarrow E+T$$

$$(2) E \rightarrow T$$

$$(3) T \rightarrow T * F$$

$$(4) T \rightarrow F$$

$$(5) F \rightarrow (E)$$

$$(6) F \rightarrow id$$

Do autômato LR(0) para a tabela SLR

- Agora podemos ver o algoritmo completo
- ENTRADA:
 - Uma gramática aumentada G'
- SAÍDA:
 - As funções AÇÃO e TRANSIÇÃO da tabela de análise SLR para G'

Do autômato LR(0) para a tabela SLR

- Algoritmo:

1. Construa $C = \{I_0, I_1, \dots, I_n\}$, a coleção de conjuntos de itens LR(0) para G'
2. O estado i é construído a partir de I_i . As ações para o estado são determinadas da seguinte forma:
 - (a) Se o item $[A \rightarrow \alpha.a\beta]$ está em I_i e $\text{GOTO}(I_i, a) = I_j$
então **ACAO** $[i, a] := \text{shift } j$ // *a deve ser um terminal*
 - (b) Se o item $[A \rightarrow \alpha.]$ está em I_i
então **ACAO** $[i, a] := \text{reduce } A \rightarrow \alpha$ para todo **a**
em **seguidores(A)** // *A não pode ser S'*
 - (c) Se o item $[S' \rightarrow S.]$ está em I_i
então **ACAO** $[i, \$] := \text{OK}$ // *aceita a cadeia*
3. Se $\text{GOTO}(I_i, A) = I_j$
então **TRANSICAO** $[i, A] := j$ para todo não terminal **A**
4. As entradas não definidas nas regras 2 e 3 caracterizam **erro**
5. O estado inicial é aquele construído a partir do conjunto de itens contendo $[S' \rightarrow .S]$

Do autômato LR(0) para a tabela SLR

- Algoritmo:

1. Construa $C = \{I_0, I_1, \dots, I_n\}$, a coleção de itens
2. O estado i é construído a partir de I_0 e para o estado i são determinadas as ações seguintes:
 - (a) Se o item $[A \rightarrow \alpha.a\beta]$ está em I_i e $\text{GOTO}(I_i, a) = I_j$
então **ACAO** $[i, a] := \text{shift } j$ // *a deve ser um terminal*
 - (b) Se o item $[A \rightarrow \alpha.]$ está em I_i
então **ACAO** $[i, a] := \text{reduce } A \rightarrow \alpha$ para todo a
em **seguidores**(A) // *A não pode S'*
 - (c) Se o item $[S' \rightarrow S.]$ está em I_i
então **ACAO** $[i, \$] := \text{OK}$ // *aceita a cadeia*
3. Se $\text{GOTO}(I_i, A) = I_j$
então **TRANSICAO** $[i, A] := j$ para todo não terminal A
4. As entradas não definidas nas regras 2 e 3 caracterizam **erro**
5. O estado inicial é aquele construído a partir do conjunto de itens contendo $[S' \rightarrow .S]$

Basta copiar da
tabela do
autômato LR(0),
escolhendo a
ação "shift"

Do autômato LR(0) para a tabela SLR

- Algoritmo para a construção da tabela de ações para o autômato LR(0) para o estado I_i são determinados a partir dos itens em I_i e os terminais em Σ .
 1. Constrói-se a tabela de ações para o estado I_i para o estado I_i são determinados a partir dos itens em I_i e os terminais em Σ .
 2. O algoritmo para a construção da tabela de ações para o estado I_i é o seguinte:
 - (a) Se o item $[A \rightarrow \alpha.a\beta]$ está em I_i e $\text{FOLLOW}(I_i, a) = I_j$ então **ACAO** $[i, a] := \text{shift } j$ // *a deve ser um terminal*
 - (b) Se o item $[A \rightarrow \alpha.]$ está em I_i então **ACAO** $[i, a] := \text{reduce } A \rightarrow \alpha$ para todo a em **seguidores(A)** // *A não pode S'*
 - (c) Se o item $[S' \rightarrow S.]$ está em I_i então **ACAO** $[i, a] := \text{OK}$ // *aceita a cadeia*
 3. Procura-se as colunas com terminais que estão em **seguidores(A)**.
 - (a) Se $a \in \text{seguidores}(A)$ então $\text{ACAO}[i, a] = j$ para j o índice da produção $A \rightarrow \alpha$ das na I_j que contém o item $[A \rightarrow \alpha.a\beta]$.
 5. Procura-se os itens contendo $[S' \rightarrow .S]$ e coloca rn nas células correspondentes (onde n é o índice da produção).

Tabela SLR

(1) $E \rightarrow E+T$

(2) $E \rightarrow T$

(3) $T \rightarrow T * F$

(4) $T \rightarrow F$

(5) $F \rightarrow (E)$

(6) $F \rightarrow id$

Linha 11:

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

Item completo:

$[F \rightarrow (E) \cdot]$

$seg(F) = \{ *, +,), \$ \}$

Índice $F \rightarrow (E) = 5$

Ação = r5

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4					
1		s6							
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4					
5		r6	r6		r6	r6			
6	s5			s4					
7	s5			s4					
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Do autômato LR(0) para a tabela SLR

- Algoritmo:

1. Construa $C = \{I_0, I_1, \dots, I_n\}$, a coleção de conjuntos de itens LR(0) para G'

2. O estado i é construído a partir de I_i . As ações para o estado são determinadas na seguinte forma:

(a) Se o item $[A \rightarrow \alpha.a\beta]$ está em I_i , então **ACAO** $[i, a] := \text{shift}$

(b) Se o item $[A \rightarrow \alpha.]$ está em I_i , então **ACAO** $[i, a] := \text{reduce}$

para todo a em seguidores(A) // A não pode S'

(c) Se o item $[S' \rightarrow S.]$ está em I_i , então **ACAO** $[i, \$] := \text{OK}$

3. Se $\text{GOTO}(I_i, A) = I_j$, então **TRANSICAO** $[i, A] := j$ para todo A

4. As entradas não definidas nas regras anteriores são marcadas como erro

5. O estado inicial é aquele construído a partir de itens contendo $[S' \rightarrow .S]$

Equivalente à regra b, mas com o símbolo inicial, escolhendo ação "OK"

O "truque" é achar a linha com um item completo envolvendo S'

Tabela SLR

Linha 1:

```
I1={ [E'→E.],
      [E→E.+T]
    }
```

Item completo:

[E'→E.]

Ação = OK

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4					
1		s6				OK			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4					
5		r6	r6		r6	r6			
6	s5			s4					
7	s5			s4					
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Do autômato LR(0) para a tabela SLR

- Algoritmo:

1. Construa $C = \{I_0, I_1, \dots, I_n\}$, a coleção de conjuntos de itens LR(0) para G'
2. O estado i é construído a partir de I_i . As ações para o estado são determinadas da seguinte forma:
 - (a) Se o item $[A \rightarrow \alpha.a\beta]$ está em I_i e $\text{GOTO}(I_i, a) = I_j$
então **ACAO** $[i, a] := \text{shift } j$ // *a deve ser um terminal*
 - (b) Se o item $[A \rightarrow \alpha.]$ está em I_i
então **ACAO** $[i, a] := \text{reduce}$ em **seg** *de S'*
 - (c) Se o item $[S' \rightarrow S.]$ está em I_i
então **ACAO** $[i, \$] := \text{OK}$ *a cadeia*
3. Se $\text{GOTO}(I_i, A) = I_j$
então **TRANSICAO** $[i, A] := j$ para todo não terminal A
4. As entradas não definidas nas regras 2 e 3 caracterizam **erro**
5. O estado inicial é aquele construído a partir do conjunto de itens contendo $[S' \rightarrow .S]$

Basta copiar da
tabela do
autômato LR(0)

Tabela SLR

	id	+	*	()	E	T	F
I ₀	I ₅			I ₄	I ₁	I ₂	I ₃	
I ₁		I ₆						
I ₂			I ₇					
I ₃								
I ₄	I ₅			I ₄	I ₈	I ₂	I ₃	
I ₅								
I ₆	I ₅			I ₄		I ₉	I ₃	
I ₇	I ₅			I ₄			I ₁₀	
I ₈		I ₆			I ₁₁			
I ₉			I ₇					
I ₁₀								
I ₁₁								

Estados	Ações						Transições		
	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				OK			
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	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Exercício Tabela SLR

- Construa a tabela SLR para a gramática a seguir

$S \rightarrow a \mid (L)$

$L \rightarrow L ; S \mid S$

Resposta Tabela SLR

- Passo 1 – aumentar a gramática

$$S' \rightarrow S$$
$$S \rightarrow a \mid (L)$$
$$L \rightarrow L ; S \mid S$$

Resposta Tabela SLR

$$\begin{array}{l} S' \rightarrow S \\ S \rightarrow a \mid (L) \\ L \rightarrow L ; S \mid S \end{array}$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

$$C = \text{CLOSURE}(S' \rightarrow \cdot S) = \{ [S' \rightarrow \cdot S], [S \rightarrow \cdot a], [S \rightarrow \cdot (L)] \} = I_0$$

$$a: I_0 = \left\{ \begin{array}{l} [S' \rightarrow \cdot S], \\ [S \rightarrow \cdot a], \\ [S \rightarrow \cdot (L)] \end{array} \right\}$$

$$b: \text{GOTO}(I_0, S) = \{ [S' \rightarrow S \cdot] \} = I_1$$

$$\text{GOTO}(I_0, a) = \{ [S \rightarrow a \cdot] \} = I_2$$

$$\text{GOTO}(I_0, () = \{ [S \rightarrow (\cdot L)], [L \rightarrow \cdot L; S], [L \rightarrow \cdot S], [S \rightarrow \cdot a], [S \rightarrow \cdot (L)] \} = I_3$$

Resposta Tabela SLR

$$S' \rightarrow S$$
$$S \rightarrow a \mid (L)$$
$$L \rightarrow L ; S \mid S$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

a: $I_1 = \{ [S' \rightarrow S.] \}$

b: todos GOTO são
vazios

a: $I_2 = \{ [S \rightarrow a.] \}$

b: todos GOTO são
vazios

Resposta Tabela SLR

$$S' \rightarrow S$$

$$S \rightarrow a \mid (L)$$

$$L \rightarrow L ; S \mid S$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

$$a: I_3 = \left\{ \begin{array}{l} [S \rightarrow (.L)], \\ [L \rightarrow .L;S], \\ [L \rightarrow .S], \\ [S \rightarrow .a], \\ [S \rightarrow .(L)] \end{array} \right\}$$

$$b: \begin{array}{l} \text{GOTO}(I_3, L) = \{ [S \rightarrow (L.)], [L \rightarrow L.;S] \} = I_4 \\ \text{GOTO}(I_3, S) = \{ [L \rightarrow S.] \} = I_5 \\ \text{GOTO}(I_3, a) = \{ [S \rightarrow a.] \} = I_2 \\ \text{GOTO}(I_3, () = \{ [S \rightarrow (.L)], [L \rightarrow .L;S], [L \rightarrow .S], \\ [S \rightarrow .a], [S \rightarrow .(L)] \} = I_3 \end{array}$$

Resposta Tabela SLR

$$S' \rightarrow S$$

$$S \rightarrow a \mid (L)$$

$$L \rightarrow L ; S \mid S$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

$$\begin{aligned} a: I_4: & \{ [S \rightarrow (L.)], \\ & [L \rightarrow L.; S] \} \\ b: \text{GOTO}(I_4,) &= \{ [S \rightarrow (L).] \} = I_6 \\ \text{GOTO}(I_4, ;) &= \{ [L \rightarrow L;.S], \\ & [S \rightarrow .a], \\ & [S \rightarrow .(L)] \} = I_7 \end{aligned}$$

Resposta Tabela SLR

$$S' \rightarrow S$$

$$S \rightarrow a \mid (L)$$

$$L \rightarrow L ; S \mid S$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

$$a: I_5 = \{ [L \rightarrow S.] \}$$

b: todos GOTO são

vazios

$$a: I_6 = \{ [S \rightarrow (L).] \}$$

b: todos GOTO são

vazios

Resposta Tabela SLR

 $S' \rightarrow S$ $S \rightarrow a \mid (L)$ $L \rightarrow L ; S \mid S$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

$$\text{a: } I_7 = \left\{ \begin{array}{l} [L \rightarrow L; . S], \\ [S \rightarrow . a], \\ [S \rightarrow . (L)] \end{array} \right\}$$

$$\begin{aligned} \text{b: } \text{GOTO}(I_7, S) &= \{ [L \rightarrow L; S.] \} = I_8 \\ \text{GOTO}(I_7, a) &= \{ [S \rightarrow a.] \} = I_2 \\ \text{GOTO}(I_7, () &= \{ [S \rightarrow (. L)], \dots \} = I_3 \end{aligned}$$

Resposta Tabela SLR

$$S' \rightarrow S$$
$$S \rightarrow a \mid (L)$$
$$L \rightarrow L ; S \mid S$$

- Passo 2 – calcular a coleção canônica de conjuntos de itens LR(0)

a: $I_8 = \{ [L \rightarrow L; S.] \}$

b: todos GOTO são

vazios

Resposta Tabela SLR

$$\begin{array}{l} S' \rightarrow S \\ S \rightarrow a \mid (L) \\ L \rightarrow L ; S \mid S \end{array}$$

- Conjunto canônico:

$$I_0 = \{ [S' \rightarrow \cdot S], [S \rightarrow \cdot a], [S \rightarrow \cdot (L)] \}$$
$$I_1 = \{ [S' \rightarrow S \cdot] \}$$
$$I_2 = \{ [S \rightarrow a \cdot] \}$$
$$I_3 = \{ [S \rightarrow (\cdot L)], [L \rightarrow \cdot L; S], [L \rightarrow \cdot S], \\ [S \rightarrow \cdot a], [S \rightarrow \cdot (L)] \}$$
$$I_4 = \{ [S \rightarrow (L \cdot)], [L \rightarrow L \cdot ; S] \}$$
$$I_5 = \{ [L \rightarrow S \cdot] \}$$
$$I_6 = \{ [S \rightarrow (L) \cdot] \}$$
$$I_7 = \{ [L \rightarrow L ; \cdot S], [S \rightarrow \cdot a], [S \rightarrow \cdot (L)] \}$$
$$I_8 = \{ [L \rightarrow L ; S \cdot] \}$$

Resposta

Tabela SLR

- Autômato LR(0)
 - Função GOTO

$S' \rightarrow S$

$S \rightarrow a \mid (L)$

$L \rightarrow L ; S \mid S$

GOTO	a	;	()	S	L
I ₀	I ₂		I ₃		I ₁	
I ₁						
I ₂						
I ₃	I ₂		I ₃		I ₅	I ₄
I ₄		I ₇		I ₆		
I ₅						
I ₆						
I ₇	I ₂		I ₃		I ₈	
I ₈						

Resposta Tabela SLR

- Calculando conjuntos seguidores

$S' \rightarrow S$

$S \rightarrow a \mid (L)$

$L \rightarrow L ; S \mid S$

primeiros (S') = {a, (}

primeiros (S) = {a, (}

Primeiros (L) = {a, (}

seguidores (S') = {\$}

seguidores (S) = {\$,), , ; }

seguidores (L) = {), , ; }

Resposta Tabela SLR

- Indexando a gramática

$S' \rightarrow S$

(1) $S \rightarrow a$

(2) $S \rightarrow (L)$

(3) $L \rightarrow L ; S$

(4) $L \rightarrow S$

Resposta Tabela SLR

- Movimentos “shift”

GOTO	a	;	()	\$	L
I ₀	I ₂	I ₃	I ₁			
I ₁						
I ₂						
I ₃	I ₂	I ₃	I ₅	I ₄		
I ₄	I ₇	I ₆				
I ₅						
I ₆						
I ₇	I ₂	I ₃	I ₈			
I ₈						

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2							
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 0:

```
I0 = { [S' → .S],
       [S → .a],
       [S → .(L)]
     }
```

Nenhum item completo!

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2							
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 1:

$I_1 = \{ [S' \rightarrow S.] \}$

Item completo:

$[S' \rightarrow S.]$

Mas envolve S' ,
portanto ignora

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2							
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 2:

$I_2 = \{ [S \rightarrow a \cdot] \}$

Item completo:

$[S \rightarrow a \cdot]$

$\text{seg}(S) = \{ \$, , , ; \}$

Índice $S \rightarrow a = 1$

Ação = r1

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 3:

```
I3 = { [S → (.L)],  
      [L → .L; S],  
      [L → .S],  
      [S → .a],  
      [S → .(L)]  
    }
```

Nenhum item completo!

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 4:

$I_4 = \{ [S \rightarrow (L.)], [L \rightarrow L.;S] \}$

Nenhum item completo!

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5							
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 5:

$I_5 = \{ [L \rightarrow S \cdot] \}$

Item completo:

$[L \rightarrow S \cdot]$

$\text{seg}(L) = \{) , ; \}$

Índice $L \rightarrow S = 4$

Ação = r4

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5		r4		r4			
6							
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 6:

$I_6 = \{ [S \rightarrow (L) \cdot] \}$

Item completo:

$[S \rightarrow (L) \cdot]$

$\text{seg}(S) = \{ \$,), , ; \}$

Índice $S \rightarrow (L) = 2$

Ação = r2

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 7:

```
I7 = { [L→L; .S],  
       [S→.a],  
       [S→.(L)]  
 }
```

Nenhum item
completo!

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3				
8							

Resposta Tabela SLR

- Movimentos “reduce”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 8:

$I_8 = \{ [L \rightarrow L ; S.] \}$

Item completo:

$[L \rightarrow L ; S.]$

$\text{seg}(L) = \{ \text{), , ;} \}$

Índice $L \rightarrow L ; S = 3$

Ação = r3

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1							
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3				
8		r3		r3			

Resposta Tabela SLR

- Movimentos “OK”

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

Linha 1:

$I1 = \{ [S' \rightarrow S.] \}$

Item completo:

$[S' \rightarrow S.]$

Ação = OK

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3				
1					OK		
2		r1		r1	r1		
3	s2		s3				
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3				
8		r3		r3			

Resposta Tabela SLR

- Transições

- (1) $S \rightarrow a$
- (2) $S \rightarrow (L)$
- (3) $L \rightarrow L ; S$
- (4) $L \rightarrow S$

GOTO

	a	;	()	\$	L
I ₀	I ₂	I ₃	I ₁			
I ₁						
I ₂						
I ₃	I ₂	I ₃	I ₅	I ₄		
I ₄		I ₇	I ₆			
I ₅						
I ₆						
I ₇	I ₂	I ₃	I ₈			
I ₈						

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3			1	
1					OK		
2		r1		r1	r1		
3	s2		s3			5	4
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3			8	
8		r3		r3			

Tabela SLR

- Resposta!

Estados	Ações					Transições	
	a	;	()	\$	S	L
0	s2		s3			1	
1					OK		
2		r1		r1	r1		
3	s2		s3			5	4
4		s7		s6			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3			8	
8		r3		r3			

Gramática SLR(1)

- Uma gramática é SLR(1) sse, para qualquer estado s , as duas condições a seguir são satisfeitas:
 1. Para qualquer item $[A \rightarrow \alpha . X \beta]$ em s em que X é um terminal
 - Não existe um item completo $[B \rightarrow \gamma .]$ em s com X em seguidores(B)
 2. Para quaisquer dois itens completos $[A \rightarrow \alpha .]$ e $[B \rightarrow \beta .]$ em s
 - Temos seguidores(A) \cap seguidores(B) vazio

Conflitos

- Violações nessas condições geram conflitos, respectivamente:
 1. Empilha-reduz (*shift-reduce*)
 2. Reduz-reduz (*reduce-reduce*)
- Na tabela, isso aparece na forma de duas ou mais ações em uma mesma célula

Exemplo conflito empilha-reduz

`decl → decl-if | 'outra'`

`decl-if → 'if' '(' exp ')' decl`

`| 'if' '(' exp ')' decl 'else' decl`

`exp → '0' | '1'`

- Na gramática acima, haverá um conflito na transição

`[decl-if → 'if' '(' exp ')' decl.]`

`[decl-if → 'if' '(' exp ')' decl.
'else' decl]`

- Trata-se de um conflito empilha-reduz, pois
 - O item completo indica uma redução
 - O outro item indica que o 'else' deve ser empilhado

Exemplo conflito reduz-reduz

`decl` → `ativa-decl` | `atrib-decl`

`ativa-decl` → `ID`

`atrib-decl` → `var` `' := '` `exp`

`var` → `ID`

`exp` → `var` | `NUM`

- Na gramática acima, há um estado no qual estão presentes os seguintes itens

[`ativa-decl`→`ID`.]

[`var`→`ID`.]

- Ocorre um conflito reduz-reduz, já que
 - seguidores(`ativa-decl`) = {`$`}
 - seguidores(`var`) = {`' := '`, `$`}

Resolução de conflitos

- Empilha-reduz
 - Opção default (sempre empilha)
 - Resolve a ambiguidade do “else sobrando”
 - Um else sempre deve ser agrupado com o if mais próximo

- Ex:

`if (1) then if (2) then outra else outra`

- Decisão errada:

Pilha	Entrada	Ação
if (exp) then if(exp) then outra	else outra \$	reduce
if (exp) then decl-if	else outra \$	shift
if (exp) then decl-if else	outra \$	shift
if (exp) then decl-if else outra	\$	reduce
decl-if	\$	

Resolução de conflitos

- Empilha-reduz
 - Opção default (sempre empilha)
 - Resolve a ambiguidade do “else sobrando”
 - Um else sempre deve ser agrupado com o if mais próximo

- Ex:

`if (1) then if (2) then outra else outra`

- Decisão certa:

Pilha	Entrada	Ação
if (exp) then if(exp) then outra	else outra \$	shift
if (exp) then if(exp) then outra else	outra \$	shift
if (exp) then if(exp) then outra else outra	\$	reduce
if (exp) then decl-if	\$	reduce
decl-if	\$	

Resolução de conflitos

- Empilha-reduz
 - Associatividade/precedência
 - Resolve ambiguidades de expressões
- Consiste em definir explicitamente a relação de precedência/associatividade entre os terminais
 - Ex: $* > +$, $+ > +$, etc
- No momento da dúvida:
 - a = o terminal mais à direita na pilha
 - b = a entrada atual
 - Se $a > b$, reduz
 - Se $a < b$, empilha

Conflitos

- **Ex:**

`id+id*id`

Momento de dúvida, pois existe um estado com os seguintes itens:

$[E \rightarrow E + E \cdot]$

$[E \rightarrow E \cdot * E]$

e^* está em seguidores(E)

Pilha	Entrada	Ação
E + E	* id \$	shift
E + E *	id \$	shift
E + E * id	\$	reduce
E + E * E	\$	reduce
E + E	\$	reduce
E	\$...

Neste exemplo

$a = +$

$b = *$

$+ < *$, portanto empilha

Conflitos

- **Ex:**

id+id+id

Momento de dúvida, pois existe um estado com os seguintes itens:

[E → E + E .]

[E → E . + E]

e + está em seguidores(E)

Pilha	Entrada	Ação
E + E	+ id \$	reduce
E	+ id \$	shift
E +	id \$	shift
E + id	\$	reduce
E + E	\$	reduce
E	\$...

Neste exemplo

a = +

b = +

+ == +, portanto reduz
(Por que?)

Resolução de conflitos

- Empilha-reduz
 - Modificar a gramática

- Ex:
 - De:

$$E \rightarrow E+E \mid E * E \mid id$$

- Para:

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow id$$

Resolução de conflitos

- Reduz-reduz
 - Normalmente indica ambiguidade ou outro problema no projeto da gramática
 - Precisa re-escrever a gramática
 - Não existe regra para isso
 - É uma das desvantagens da análise LR
- Mas na análise SLR existem muitos conflitos que surgem porque o método não é suficientemente poderoso
 - Existem outros métodos de análise LR melhores
 - Análise LR(1) canônica
 - Análise LALR

Análise LR(1) canônica

- Similar à SLR, mas considerando um conjunto maior de itens
 - Itens LR(1)
- Consiste em estender a coleção de itens LR(0), para incluir um segundo componente
 - Um símbolo de lookahead
 - LR(1) = itens com 1 símbolo lookahead
- Formato dos itens:
 - $[A \rightarrow \alpha \cdot \beta, a]$
 - onde $A \rightarrow \alpha \beta$ é uma produção e a é um terminal ou $\$$

Análise LR(1) canônica

- Os itens LR(1) servem para restringir a decisão pela redução, ou seja,
 - Um item $[A \rightarrow \alpha . , a]$ indica que a redução $A \rightarrow \alpha$ só ocorre se o próximo símbolo de entrada for a
- A construção da coleção de itens LR(1) é feita da mesma forma
 - Mas mudam as funções CLOSURE e GOTO

Fecho de conjuntos de itens LR(1)

- Algoritmo:

```
SetOfItems CLOSURE(I) {
```

```
  J := I;
```

```
  repeat
```

```
    for (cada item: "[A→α.Bβ, a]" em J)
```

```
      for (cada produção "B→γ" de G)
```

```
        for (cada terminal b em primeiros(βa))
```

```
          if ("B→.γ" não está em J)
```

```
            adicione "[B→.γ, b]" em J;
```

```
  until nenhum item seja adicionado a
```

```
    J em um passo do loop;
```

```
  return J;
```

```
}
```

 Mudanças

Função GOTO para itens LR(1)

- Nada muda, apenas é necessário analisar os itens considerando também o componente lookahead
- As transições serão exatamente as mesmas, exceto que só é possível fazer a transição entre itens com o mesmo componente lookahead
- Ex:
 - $GOTO(\{[E.+E, \$]\}, +) = \text{Closure}(\{[E+.E, \$]\})$
 $= \{[E+.E, \$], \dots\}$

Tabela de análise LR(1) canônica

- Algoritmo:

1. Construa $C' = \{I_0, I_1, \dots, I_n\}$, a coleção de conjuntos de itens LR(1) para G'

2. O estado i é construído a partir de I_i . As ações para o estado são determinadas da seguinte forma:

(a) Se o item $[A \rightarrow \alpha \cdot a \beta, b]$ está em I_i e $\text{GOTO}(I_i, a) = I_j$
então **ACAO** $[i, a] := \text{shift } j$ // *a deve ser um terminal*

(b) Se o item $[A \rightarrow \alpha \cdot, a]$ está em I_i
então **ACAO** $[i, a] := \text{reduce } A \rightarrow \alpha$ ~~para todo a em seguidores(A)~~

(c) Se o item $[S' \rightarrow S \cdot, \$]$ está em I_i
então **ACAO** $[i, \$] := \text{OK}$ // *aceita a cadeia*

3. Se $\text{GOTO}(I_i, A) = I_j$
então **TRANSICAO** $[i, A] := j$ para todo não-terminal A

4. As entradas não definidas nas regras 2 e 3 caracterizam **erro**

5. O estado inicial é aquele construído a partir do conjunto de itens contendo $[S' \rightarrow \cdot S, \$]$

 Mudanças

Exemplo análise LR(1) canônica

- **Conflito reduz-reduz**

`decl → ativa-decl | atrib-decl`

`ativa-decl → ID`

`atrib-decl → var ' := ' exp`

`var → ID`

`exp → var | NUM`

- **Na gramática acima, para o conflito anterior teríamos**

`[ativa-decl→ID., $]`

`[var→ID., ' := ']`

Exemplo análise LR(1) canônica

- Na gramática acima, para o conflito anterior teríamos
[ativa-decl→ID., \$]
[var→ID., ' := ']
- O conflito é solucionado já que
 - Os itens LR(1) diferenciam as reduções com base nas suas verificações à frente:
 - Reduzir usando a primeira opção se o símbolo à frente for '\$'
 - Reduzir usando a segunda se for ' := '

Análise LR(1) canônica

- A tabela LR(1) canônica inclui todas as possibilidades de movimentos considerando-se um símbolo à frente
 - Ou seja, os itens incluem toda combinação de movimentos empilha/reduz para TODO símbolo terminal
 - Isso resulta em tabelas muito grandes

Análise LALR (*Lookahead* LR)

- Frequentemente usado na prática
 - Tabelas consideravelmente menores do que as LR canônicas
 - Gramáticas LALR são capazes de expressar a maioria das construções sintáticas comuns das linguagens de programação
 - Não computa o conjunto completo de itens LR(1) na prática

Análise LALR (*Lookahead* LR)

- Vantagens
 - Preserva alguns dos benefícios da análise sintática SLR(1)
 - Também preserva o menor tamanho do autômato de itens LR(0)
 - Porém utiliza a capacidade lookahead de (alguns) itens LR(1)

Análise LALR (*Lookahead LR*)

- Outra característica interessante:
 - Analisadores LR canônicos e LALR se comportam da mesma forma se a entrada for correta
 - Significa que farão a mesma sequência de shifts e reduces

Análise LALR (*Lookahead* LR)

- Quando houver uma entrada errada:
 - O analisador LALR pode tentar fazer umas reduções a mais antes de detectar o erro
 - Ou seja, o analisador LR detecta o erro antes
 - Mas o LALR vai eventualmente encontrar o erro
 - Na verdade, antes de fazer o próximo shift
- Em resumo: LALR é fortemente indicado!
 - Por isso é o mais usado na prática para análise bottom-up

Recuperação de erros na análise LR

- Erros são detectados ao ler a tabela AÇÃO
 - Nunca ao ler a tabela TRANSIÇÃO
- Duas estratégias de recuperação:
 - Modo pânico
 - Recuperação em nível de frase

Recuperação em modo pânico

- Procure na pilha (de cima para baixo) um estado que tenha um GOTO para um não-terminal A
 - Ou seja, ache o momento em que A começou a ser analisado
- Joga fora todos os estados da pilha, inclusive esse encontrado
- Então basta ignorar os símbolos de entrada, até achar um que possa vir legitimamente após A
- Essa estratégia tenta eliminar a frase contendo o erro sintático. O analisador “finge” que reconheceu A corretamente, e continua a análise

Recuperação em nível de frase

- Examina-se cada entrada de erro na tabela e decide-se, com base na linguagem, qual erro do programador poderia gerar esse erro
 - Pode-se projetar uma rotina de recuperação
 - Cada entrada vazia da tabela pode ser um ponteiro para uma rotina
 - As ações da rotina podem incluir inserção/remoção de símbolos da pilha/entrada
- Estratégia é normalmente ad hoc

ASD vs ASA

- Vantagens da análise sintática ascendente LR
 - Os métodos de análise sintática ascendente são mais poderosos
 - Capazes de lidar com um número maior de gramáticas
 - As verificações à frente na pilha são mais simples do que as verificações à frente na entrada

LL(k) vs LR(k)

- LL(k)
 - A produção é selecionada olhando-se apenas os k primeiros símbolos que seu lado direito pode derivar
- LR(k)
 - O lado direito de uma produção é reconhecido conhecendo-se tudo que foi derivado a partir desse lado direito (na pilha)
 - Mais o esquadramento antecipado (lookahead) de k símbolos (da cadeia de entrada)
- LR é mais poderoso do que LL

ASD vs ASA

- Desvantagens da análise sintática ascendente LR
 - Exige manipulação complexa da tabela sintática
 - Construção trabalhosa
 - Conflitos reduce/reduce (pedra no sapato)
 - Resolução de conflitos é mais trabalhosa

ASD vs ASA

- ASA
 - Particularmente boa para expressões
 - Mais flexibilidade nas regras
 - Pior para depurar e resolver conflitos
 - Mais “declarativa”
- ASD (em particular o analisador recursivo)
 - Exige mais trabalho para definir a gramática
 - Não tolera recursão à esquerda
 - Às vezes requer fatoração
 - Problema reduzido com EBNF e técnica LL(*)
 - Mas depois é mais fácil depurar e resolver não-determinismos
 - Mais “imperativa”

ASA vs ASD

- A técnica ALL(*) supera praticamente todas as limitações
 - Facilidade de escrita de regras
 - Facilidade de depuração
 - Praticamente qualquer gramática
 - Desempenho excelente
- Recomendação: ANTLR

Fim