TÉCNICO LISBOA

Compiladores[1]

Análise Sintáctica Ascendente

Capítulo 4.5 e 4.6 - "Compilers: Principles, Techniques and Tools"

Prof. Alberto Abad

IST - Universidade de Lisboa

2021/2022

[1] Slides adaptados de Prof. Pedro T. Monteiro (2017/2018)

Algoritmos de parsing

Gramáticas livres de contexto divididas em:

- **LL(k)** parsing
  - Top-down - análise descendente
  - Considera sempre derivações mais à esquerda
  - Pilha:
    - pop do símbolo não terminal da cabeça da regra
    - push do corpo da regra

k - indica o número de símbolos de LOOKAHEAD

Algoritmos de parsing

Gramáticas livres de contexto divididas em:

- **LL(k)** parsing
  - Top-down - análise descendente
  - Considera sempre derivações mais à esquerda
  - Pilha:
    - pop do símbolo não terminal da cabeça da regra
    - push do corpo da regra
- **LR(k)** parsing
  - Bottom-up - análise ascendente
  - Considera sempre derivações mais à direita
  - Pilha:
    - pop do corpo da regra
    - push do símbolo não terminal da cabeça da regra

k - indica o número de símbolos de LOOKAHEAD

Análise ascendente

Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Análise ascendente

Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

$$
\begin{aligned}
E &\rightarrow T \\
T &\rightarrow T * F \\
T &\rightarrow F \\
F &\rightarrow id
\end{aligned}
$$

Árvore para a entrada **id * id**:

Análise ascendente

Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.
Exemplo:

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

Árvore para a entrada **id * id**:

Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

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

Árvore para a entrada **id * id**:

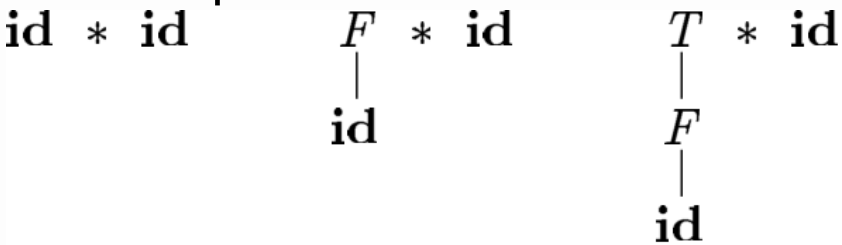
Análise ascendente

Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

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

Árvore para a entrada **id * id**:

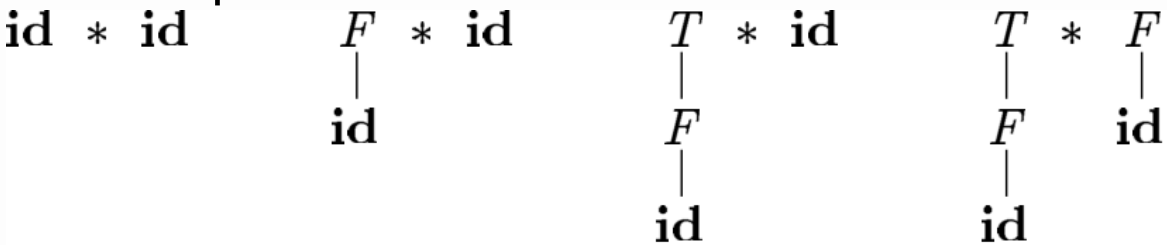


Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

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

Árvore para a entrada **id * id**:

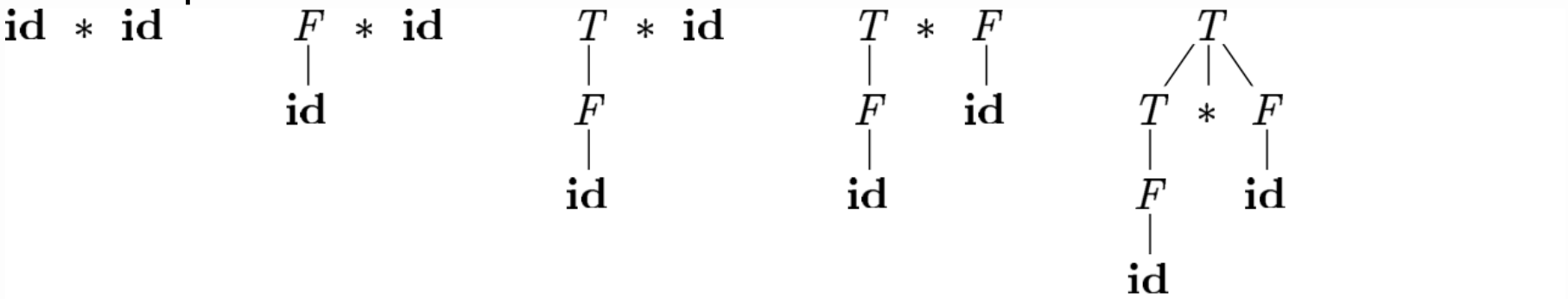


Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

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

Árvore para a entrada **id * id**:

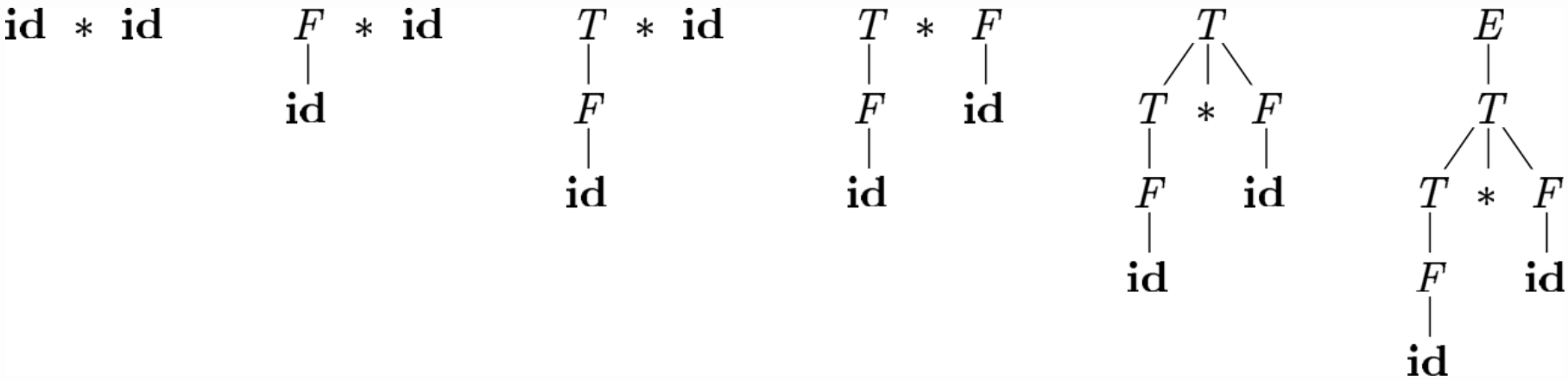


Análise ascendente: corresponde à construção da árvore de *parsing* para uma entrada a partir das folhas até à raíz.

Exemplo:

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

Árvore para a entrada **id * id**:



Acções possíveis dos parsers shift-reduce (e.g., parsers LR):

- Acção shift
  - avançar o *token* actual para o topo da pilha
  - avançar o ponteiro para o próximo *token*

Acções possíveis dos parsers shift-reduce (e.g., parsers LR):

- Acção shift
  - avançar o *token* actual para o topo da pilha
  - avançar o ponteiro para o próximo *token*
- Acção reduce
  - substituir o corpo da regra pela sua cabeça no topo da pilha
    "todo o corpo foi processado $\rightarrow$ subir um nível na árvore"

Acções possíveis dos parsers shift-reduce (e.g., parsers LR):

- Acção shift
  - avançar o *token* actual para o topo da pilha
  - avançar o ponteiro para o próximo *token*
- Acção reduce
  - substituir o corpo da regra pela sua cabeça no topo da pilha
    "todo o corpo foi processado $\rightarrow$ subir um nível na árvore"
- Acção accept
  - a pilha contêm o símbolo inicial e a entrada esta vazia

Acções possíveis dos parsers shift-reduce (e.g., parsers LR):

- Acção shift
  - avançar o *token* actual para o topo da pilha
  - avançar o ponteiro para o próximo *token*
- Acção reduce
  - substituir o corpo da regra pela sua cabeça no topo da pilha
    "todo o corpo foi processado $\rightarrow$ subir um nível na árvore"
- Acção accept
  - a pilha contêm o símbolo inicial e a entrada esta vazia
- Acção error
  - erro de sintaxe (entrada vazia na tabela de parsing)

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
$	id * id $	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
$	id * id $	shift
$ id	* id $	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
$	id * id $	shift
$ id	* id $	reduce by $F \rightarrow id$
$ F	* id $	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	shift
\$ T * id	\$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	shift
\$ T * id	\$	reduce by $F \rightarrow id$
\$ T * F	\$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	shift
\$ T * id	\$	reduce by $F \rightarrow id$
\$ T * F	\$	reduce by $T \rightarrow T * F$
\$ T	\$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	shift
\$ T * id	\$	reduce by $F \rightarrow id$
\$ T * F	\$	reduce by $T \rightarrow T * F$
\$ T	\$	reduce by $E \rightarrow T$
\$ E	\$	

Análise ascendente

Shift-Reduce parsing

Pilha	Entrada	Acções
\$	id * id \$	shift
\$ id	* id \$	reduce by $F \rightarrow id$
\$ F	* id \$	reduce by $T \rightarrow F$
\$ T	* id \$	shift
\$ T *	id \$	shift
\$ T * id	\$	reduce by $F \rightarrow id$
\$ T * F	\$	reduce by $T \rightarrow T * F$
\$ T	\$	reduce by $E \rightarrow T$
\$ E	\$	accept

- Problema: Saber quando shiftar e quando reduzir.
- Solução: LR parsers!

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

Items LR(0):

$$T \rightarrow \bullet\, T * F$$
$$T \rightarrow T \bullet * F$$
$$T \rightarrow T * \bullet F$$
$$T \rightarrow T * F \bullet$$

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

Items LR(0):

$$T \rightarrow \bullet\, T * F$$
$$T \rightarrow T \bullet * F$$
$$T \rightarrow T * \bullet F$$
$$T \rightarrow T * F \bullet$$
$$T \rightarrow \bullet F$$
$$T \rightarrow F \bullet$$

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

Items LR(0):

$$T \rightarrow \bullet\, T * F$$
$$T \rightarrow T \bullet * F$$
$$T \rightarrow T * \bullet F$$
$$T \rightarrow T * F \bullet$$
$$T \rightarrow \bullet\, F$$
$$T \rightarrow F \bullet$$
$$F \rightarrow \bullet\, id$$
$$F \rightarrow id \bullet$$

Items LR(0):

- são produções da gramática aumentadas de ponto (•)
- representam a parte da regra que já foi vista pelo parser
- existem tantos items LR(0) quantas as posições para o ponto (•)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

Items LR(0):

$$T \rightarrow \bullet\, T * F$$
$$T \rightarrow T \bullet * F$$
$$T \rightarrow T * \bullet F$$
$$T \rightarrow T * F \bullet$$
$$T \rightarrow \bullet\, F$$
$$T \rightarrow F \bullet$$
$$F \rightarrow \bullet\, id$$
$$F \rightarrow id \bullet$$

Nota: A produção $A \rightarrow \epsilon$ gera apenas um único item LR(0): $A \rightarrow \bullet$

Cada vez que o ponto (•) avança uma posição, corresponde a uma acção:

Cada vez que o ponto (•) avança uma posição, corresponde a uma acção:

- shift - quando o ponto (•) avança sobre um símbolo terminal
  $T \rightarrow T \bullet * F$
  $T \rightarrow T * \bullet F$

Cada vez que o ponto (•) avança uma posição, corresponde a uma acção:

- shift - quando o ponto (•) avança sobre um símbolo terminal
  $T \rightarrow T \bullet * F$
  $T \rightarrow T * \bullet F$
- reduce - quando o ponto (•) avança sobre um símbolo não terminal
  $T \rightarrow \bullet T * F$
  $T \rightarrow T \bullet * F$

Ideia geral:

- Reconhecer a string de entrada
- Construção do automato NFA para representar o estado do parser
  - considerar todos os items LR(0)
  - considerar todas as possíveis transições
- Determinizar o NFA e construir o DFA correspondente
- Minimizar o DFA

Ideia geral:

- Reconhecer a string de entrada
- Construção do automato NFA para representar o estado do parser
  - considerar todos os items LR(0)
  - considerar todas as possíveis transições
- Determinizar o NFA e construir o DFA correspondente
- Minimizar o DFA

No entanto...

Ideia geral:

- Reconhecer a string de entrada
- Construção do automato NFA para representar o estado do parser
  - considerar todos os items LR(0)
  - considerar todas as possíveis transições
- Determinizar o NFA e construir o DFA correspondente
- Minimizar o DFA

No entanto... a construção directa do DFA é possível!

Tal como nos analizadores léxicos, duas operações primitivas são necessárias para o DFA:

- ϵ-closure - permite adicionar items LR(0) que são possíveis/equivalentes
- goto - semelhante ao move, permite transitar de estado

Dado um conjunto I de items LR(0):

- Todos os elementos de I estão no ϵ-closure(I)
- Se $A \rightarrow \alpha \bullet B\, \beta$ pertence a ϵ-closure(I) e $B \rightarrow \gamma$ é uma produção:
  - Adicionar $B \rightarrow \bullet\, \gamma$ a ϵ-closure(I)

Dado um conjunto I de items LR(0):

- Todos os elementos de I estão no ϵ-closure(I)
- Se $A \rightarrow \alpha \bullet B\,\beta$ pertence a ϵ-closure(I) e $B \rightarrow \gamma$ é uma produção:
  - Adicionar $B \rightarrow \bullet\,\gamma$ a ϵ-closure(I)

Gramática exemplo:

$$T \rightarrow T * F$$
$$T \rightarrow F$$
$$F \rightarrow id$$

$$\mathrm{I} = \{[T \rightarrow T * \bullet F]\}$$
$$\epsilon\text{-closure(I)} = \{\ [T \rightarrow T * \bullet F],\ [F \rightarrow \bullet\, id]\ \}$$

Dado um conjunto I de items LR(0), e um símbolo da gramática X:

- goto(I,X) = ϵ-closure($\{[A \rightarrow \alpha\, X \bullet \beta]$: $\forall [A \rightarrow \alpha \bullet X\, \beta] \in I\}$)

Dado um conjunto I de items LR(0), e um símbolo da gramática X:

- goto(I,X) = ϵ-closure($\{[A \rightarrow \alpha\, X \bullet \beta]$: $\forall [A \rightarrow \alpha \bullet X\, \beta] \in I\}$)

Gramática exemplo:

$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow id$

I = $\{[T \rightarrow T \bullet * F]\}$
goto(I, *) =
ϵ-closure($\{[T \rightarrow T * \bullet F]\}$) = {
$[T \rightarrow T * \bullet F]$, $[F \rightarrow \bullet id]$ }

1º passo – Aumentar a gramática

Regra adicional $S' \rightarrow S\$$, onde S é o símbolo inicial da gramática

1º passo – Aumentar a gramática
Regra adicional $S' \rightarrow S\$$, onde S é o símbolo inicial da gramática

A regra terá associado dois items LR(0):

- $S' \rightarrow \bullet\, S\, \$$
- $S' \rightarrow S\, \bullet\, \$$

1º passo – Aumentar a gramática
Regra adicional $S' \rightarrow S\$$, onde S é o símbolo inicial da gramática

A regra terá associado dois items LR(0):

- $S' \rightarrow \bullet\, S\, \$$
- $S' \rightarrow S\, \bullet\, \$$

Quando o ponto ($\bullet$) reduz o símbolo inicial da gramática S:

- significa que a entrada foi aceite

O DFA é constituído por estados caracterizados por conjuntos de produções únicas (núcleo e outras), e transições entre estados pelos símbolos da gramática

O DFA é constituído por estados caracterizados por conjuntos de produções únicas (núcleo e outras), e transições entre estados pelos símbolos da gramática

Algoritmo:

- Iniciar o estado 0 com ϵ-closure($\{[S' \rightarrow \bullet\, S\, \$]\}$)
- Para cada estado I:
  - Para cada símbolo X da gramática
    - Calcular novo estado do DFA por $goto(I, X)$ (se não vazio)
- Repetir até nenhum estado novo ser adicionado
- O estado aceitador contém $\{\ [S' \rightarrow S\ \bullet\ \$]\ \}$

Os items LR(0) de um dado estado subdividem-se em:

- items do núcleo (ou kernel) - items que caracterizam o estado, não pudendo ser derivados a partir de outros items do mesmo estado
  - no estado inicial: $S \rightarrow \bullet\, S\,\$$
  - nos outros estados: todos os items LR(0) sem ponto ($\bullet$) no início do corpo da produção
- outros - restantes

Os items LR(0) de um dado estado subdividem-se em:

- items do núcleo (ou kernel) - items que caracterizam o estado, não pudendo ser derivados a partir de outros items do mesmo estado
  - no estado inicial: $S \rightarrow \bullet\, S\,\$$
  - nos outros estados: todos os items LR(0) sem ponto ($\bullet$) no início do corpo da produção
- outros - restantes

No exemplo anterior:

- item do núcleo: $\{\ [T \rightarrow T * \bullet\, F]\ \}$
- outros: $\{[F \rightarrow \bullet\, id]\}$

Gramática exemplo:

$$
\begin{aligned}
E &\rightarrow E + T \\
E &\rightarrow T \\
T &\rightarrow T * F \\
T &\rightarrow F \\
F &\rightarrow (E) \\
F &\rightarrow id
\end{aligned}
$$

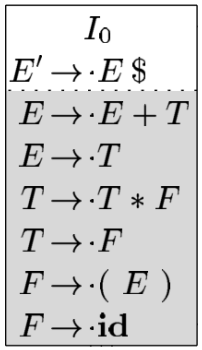
I_0
$E' \to \cdot E\,\$$

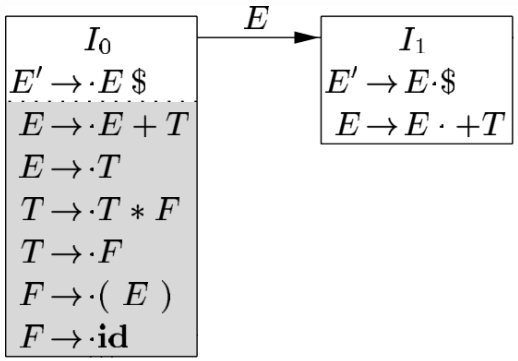
Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\ E\ )$$
$$F \to id$$



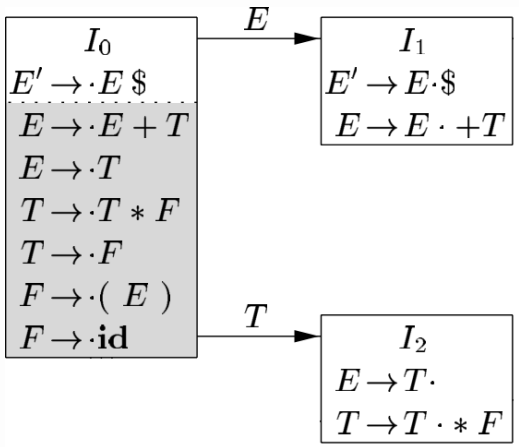


Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\ E\ )$$
$$F \to id$$

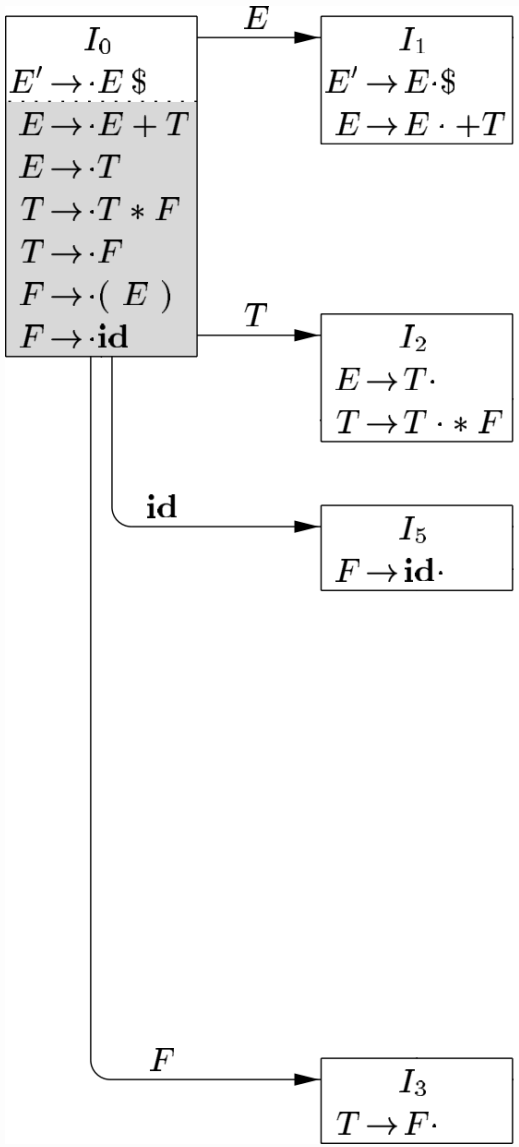
Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$



Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

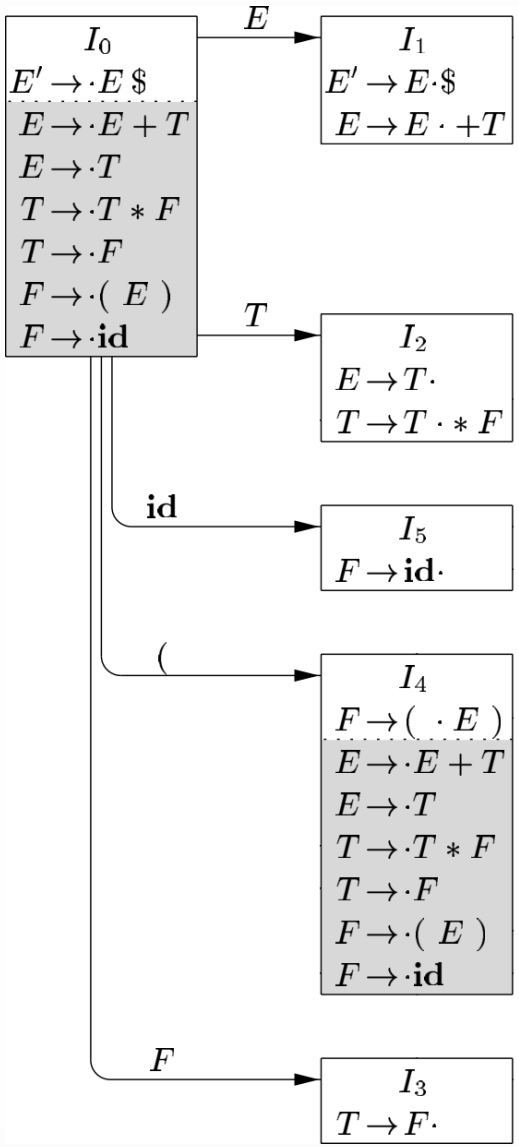


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\ E\ )$$
$$F \to id$$

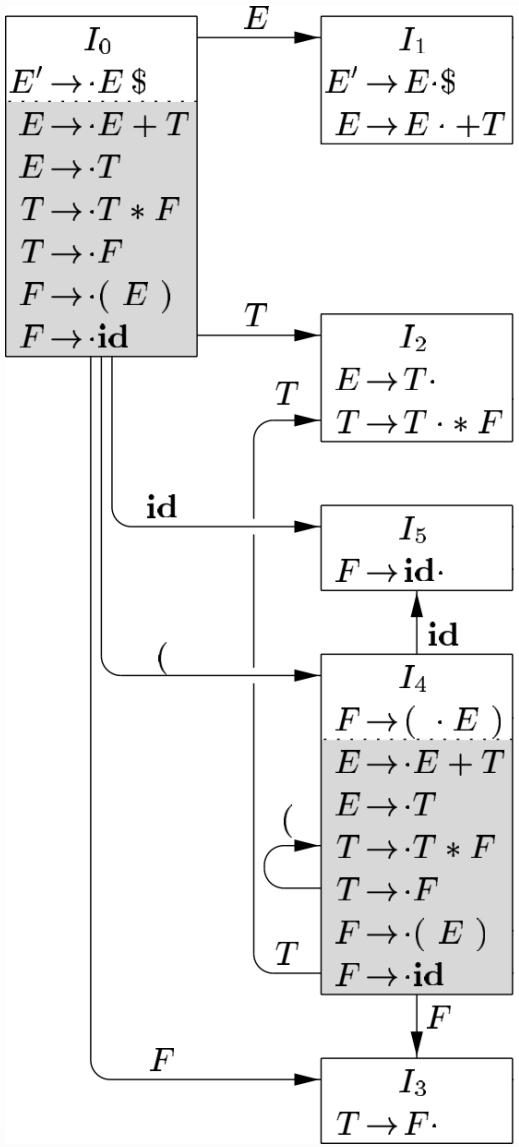


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

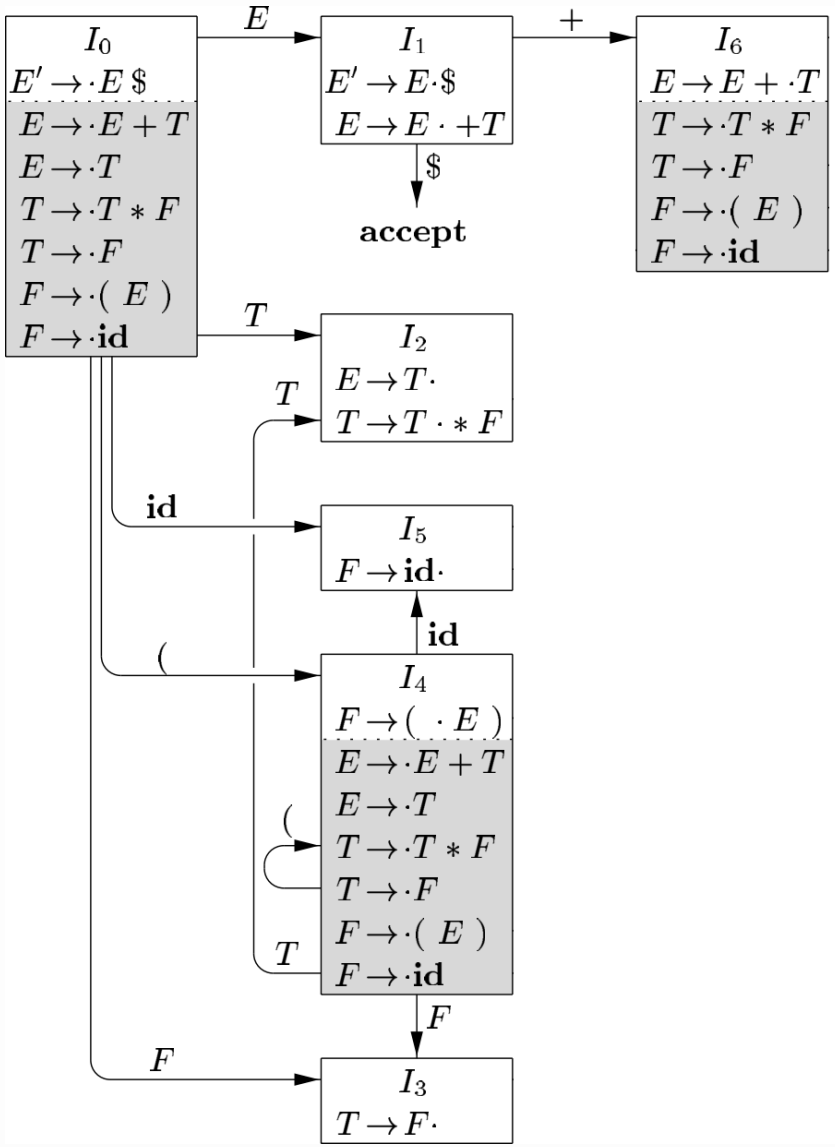


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

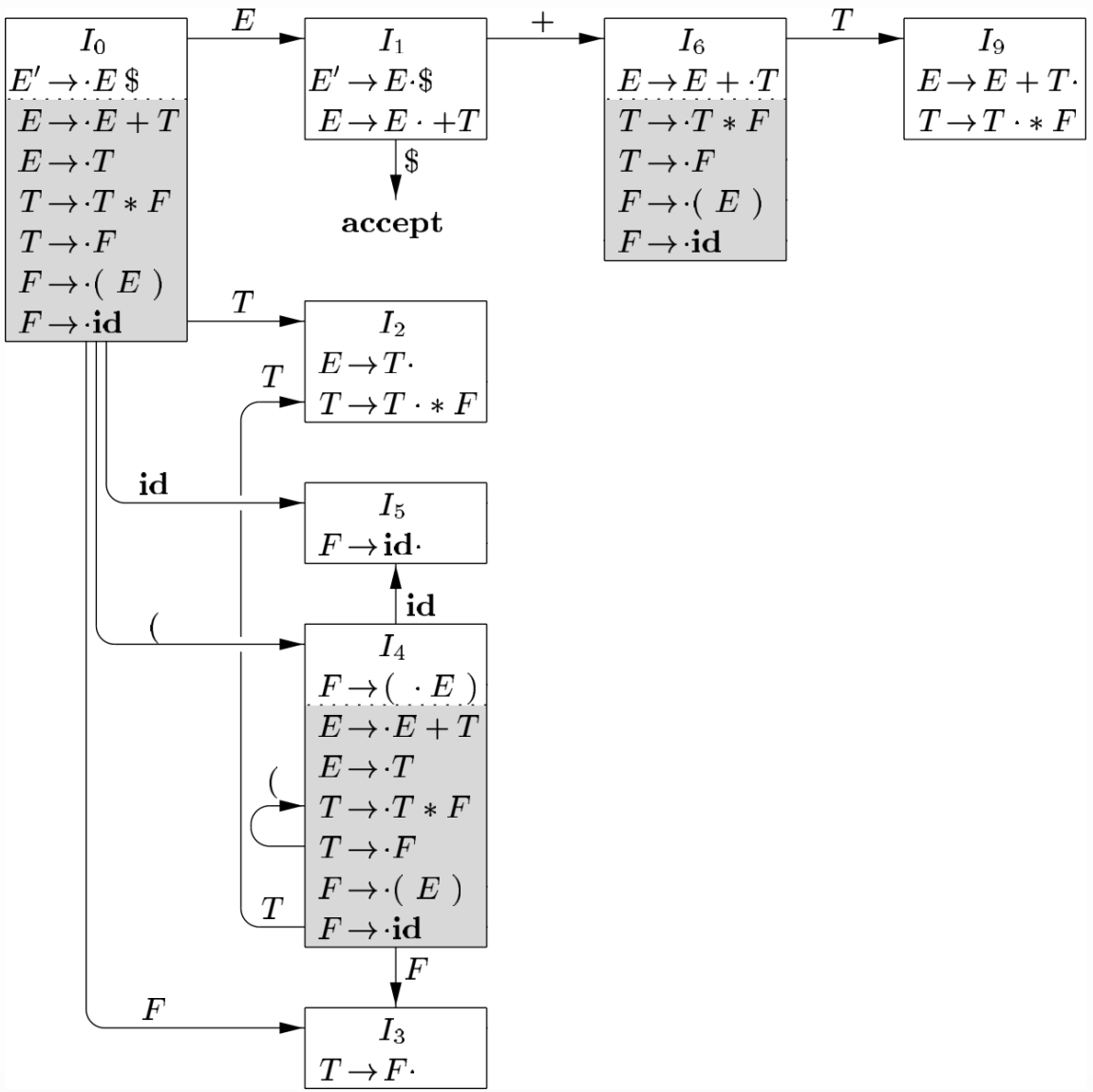


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

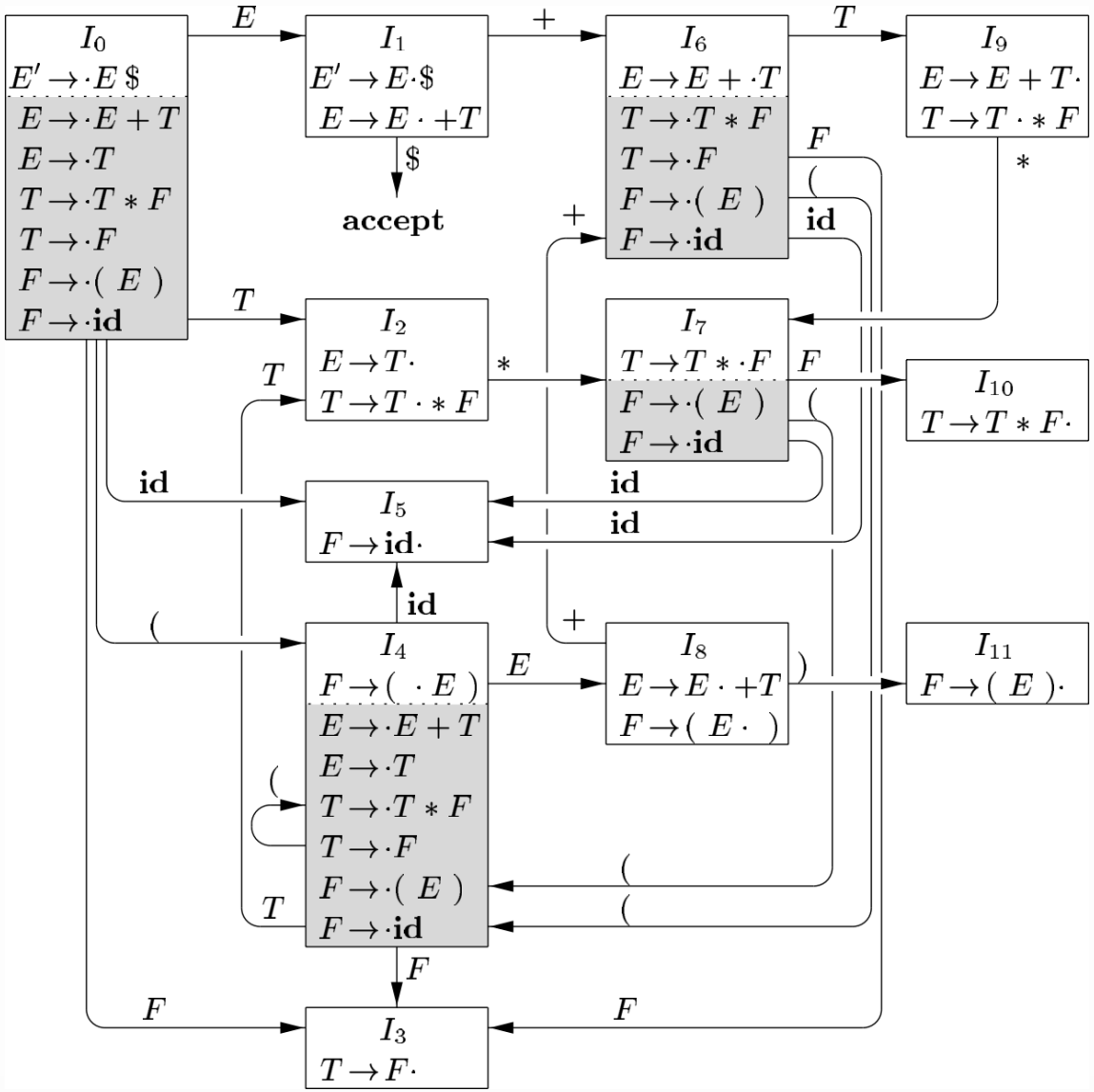


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

$$E' \to E\,\$$$
$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (\,E\,)$$
$$F \to id$$

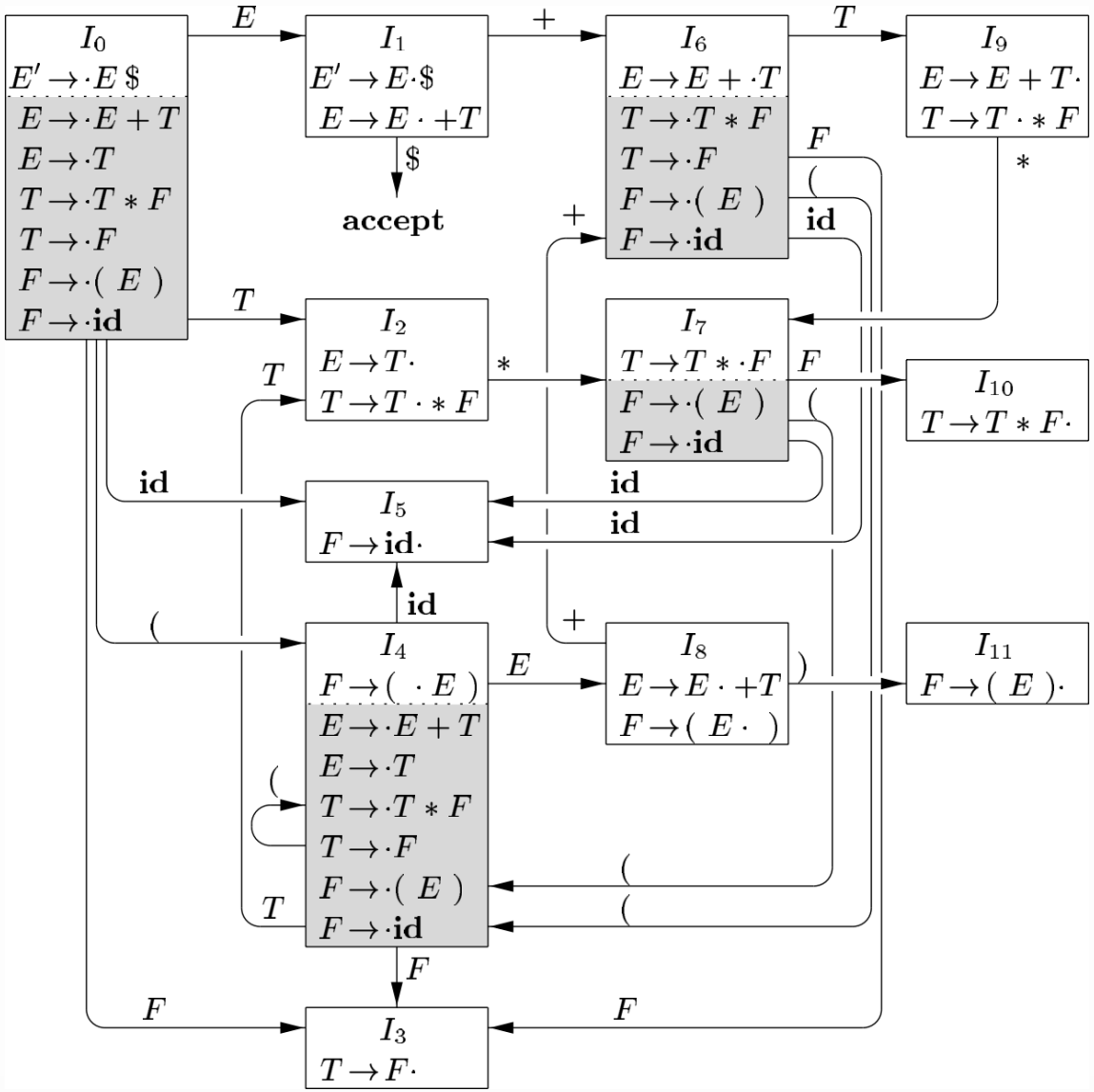


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

0 $E' \to E\,\$$
1 $E \to E + T$
2 $E \to T$
3 $T \to T * F$
4 $T \to F$
5 $F \to (\ E\ )$
6 $F \to id$

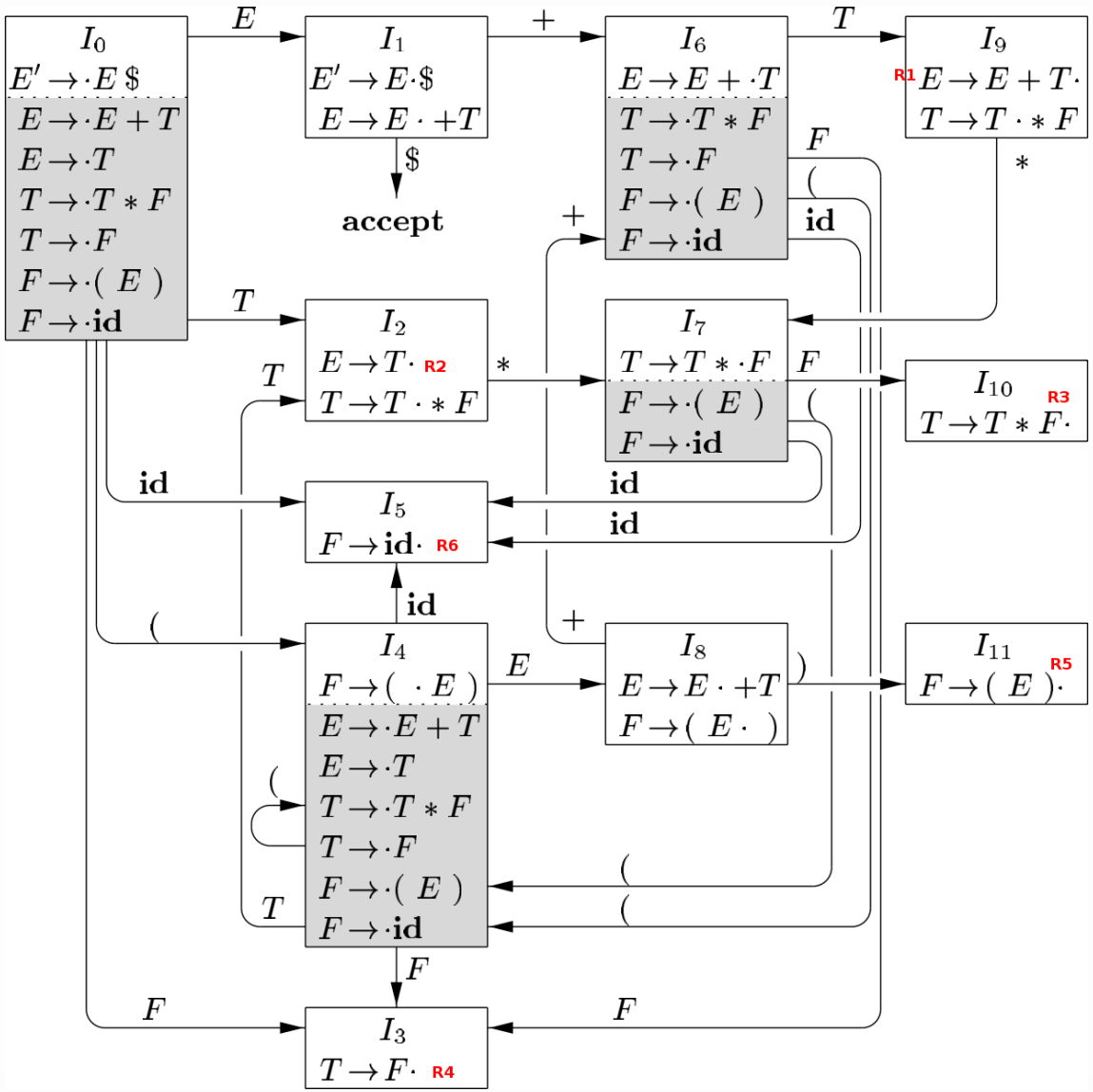


Processo de análise

Construção do DFA – Exemplo

Gramática exemplo:

0 $E' \rightarrow E\,\$$
1 $E \rightarrow E + T$
2 $E \rightarrow T$
3 $T \rightarrow T * F$
4 $T \rightarrow F$
5 $F \rightarrow (\ E\ )$
6 $F \rightarrow id$



Processo de análise

Construção da tabela de parsing

Tabela de parsing tem duas zonas onde associa estados e símbolos:

- acções (shifts, reduces, accept) definido sobre símbolos terminais
- goto definido sobre símbolos não terminais

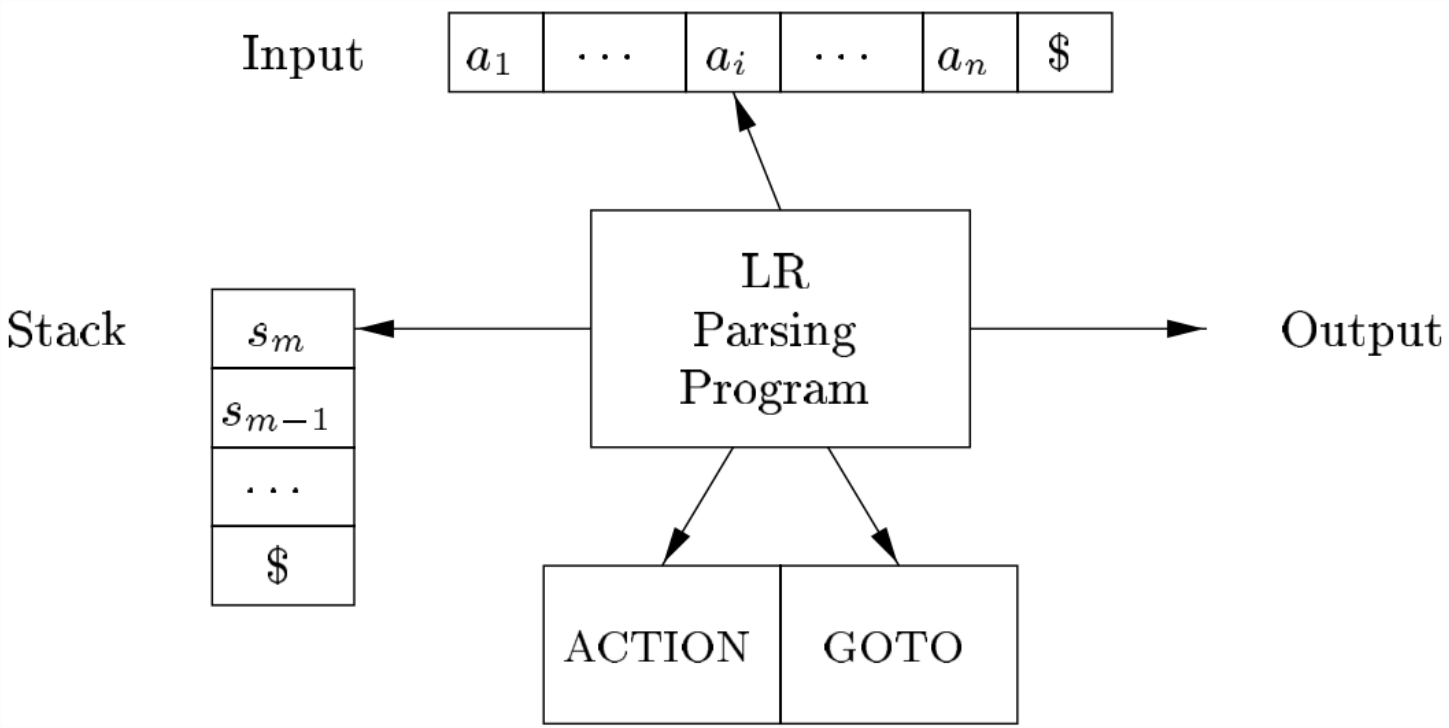


Tabela de parsing – Parser LR(0)

	+	*	(	)	id	$	E	T	F
0									

Tabela de parsing – Parser LR(0)

	+	*	(	)	id	$	E	T	F
0			4		5		1	2	3
1	6					Acc			
2	R2	7/R2	R2	R2	R2	R2			
3	R4	R4	R4	R4	R4	R4			
4			4		5		8	2	3
5	R6	R6	R6	R6	R6	R6			
6			4		5			9	3
7			4		5				10
8	6			11					
9	R1	7/R1	R1	R1	R1	R1			
10	R3	R3	R3	R3	R3	R3			
11	R5	R5	R5	R5	R5	R5			

Ambiguidade/Conflicto:
Entrada na tabela de parsing com mais de uma acção

- reduce/reduce
  caso duas regras possam ser reduzidas no mesmo estado
- shift/reduce
  caso possa reduzir uma regra e fazer shift por um símbolo terminal

Processo de análise

Construção da tabela de parsing – SLR(1)

O **parser SLR(1)** "Simple LR" constroi a tabela de parsing baseado:

- transições do DFA
- conjuntos FOLLOW dos símbolos não-terminais

O **parser SLR(1)** "Simple LR" constroi a tabela de parsing baseado:

- transições do DFA
- conjuntos FOLLOW dos símbolos não-terminais

Gramática exemplo:

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

	FIRST	FOLLOW
E		
T		
F		

O **parser SLR(1)** "Simple LR" constroi a tabela de parsing baseado:

- transições do DFA
- conjuntos FOLLOW dos símbolos não-terminais

Gramática exemplo:

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

	FIRST	FOLLOW
E	{(, id}	
T	{(, id}	
F	{(, id}	

O **parser SLR(1)** "Simple LR" constroi a tabela de parsing baseado:

- transições do DFA
- conjuntos FOLLOW dos símbolos não-terminais

Gramática exemplo:

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

	FIRST	FOLLOW
E	{(, id}	{), +, $}
T	{(, id}	{), +, *, $}
F	{(, id}	{), +, *, $}

Tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		5		1	2	3
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		5		8	2	3
5	R6	R6		R6		R6			
6			4		5			9	3
7			4		5				10
8	6			11					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Existem situações em o autómato realiza operações desnecessárias na pilha:

- transita para um estado (shift/goto)
  - Operação: shift da entrada para a pilha
- nesse estado apenas pode fazer uma redução
  - Operação: pop da pilha

Processo de análise

Compactação da tabela de parsing I: Eliminação estados

Existem situações em o autómato realiza operações desnecessárias na pilha:

- transita para um estado (shift/goto)
  - Operação: shift da entrada para a pilha
- nesse estado apenas pode fazer uma redução
  - Operação: pop da pilha

Solução: Compactar a tabela com propagação de reduções:

- eliminar estados onde só é possível fazer uma redução:
  - shift ao estado eliminado $\rightarrow$ shift composto (shift + reduce, sem mudança de estado)
  - goto ao estado eliminado $\rightarrow$ aplicar segunda redução (sem mudança de estado)

Compactação da tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		5		1	2	3
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		5		8	2	3
5	R6	R6		R6		R6			
6			4		5			9	3
7			4		5				10
8	6			11					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Compactação da tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		5		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		5		8	2	L4
5	R6	R6		R6		R6			
6			4		5			9	L4
7			4		5				10
8	6			11					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Compactação da tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		L6		8	2	L4
5	R6	R6		R6		R6			
6			4		L6			9	L4
7			4		L6				10
8	6			11					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Compactação da tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		L6		8	2	L4
5	R6	R6		R6		R6			
6			4		L6			9	L4
7			4		L6				L3
8	6			11					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Compactação da tabela de parsing – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
3	R4	R4		R4		R4			
4			4		L6		8	2	L4
5	R6	R6		R6		R6			
6			4		L6			9	L4
7			4		L6				L3
8	6			L5					
9	R1	7		R1		R1			
10	R3	R3		R3		R3			
11	R5	R5		R5		R5			

Tabela de parsing compactada – Parser SLR(1)

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
4			4		L6		8	2	L4
6			4		L6			9	L4
7			4		L6				L3
8	6			L5					
9	R1	7		R1		R1			

Eliminar entradas frequentes de redução em um determinado estado, definindo uma ação por omissão:

- A redução mais frequente passa para a coluna de comportamento por omissão:
  - Entradas diferentes da redução são mantidas inalteradas
  - Entradas de erro desaparecem
- Permite obter tabelas menos densas, reduzindo espaço.
- A aplicação de uma redução onde deveria haver um erro (símbolo que não é do follow), acabará por dar erro nos seguintes passos

Processo de análise

Compactação da tabela de parsing

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
4			4		L6		8	2	L4
6			4		L6			9	L4
7			4		L6				L3
8	6			L5					
9	R1	7		R1		R1			

Processo de análise

Compactação da tabela de parsing

	+	*	(	)	id	$	E	T	F
0			4		L6		1	2	L4
1	6					Acc			
2	R2	7		R2		R2			
4			4		L6		8	2	L4
6			4		L6			9	L4
7			4		L6				L3
8	6			L5					
9	R1	7		R1		R1			

Com coluna de comportamento por omissão:

	+	*	(	)	id	$	•	E	T	F
0			4		L6			1	2	L4
1	6					Acc				
2		7					R2			
4			4		L6			8	2	L4
6			4		L6				9	L4
7			4		L6					L3
8	6			L5						
9		7					R1			

Processamento da entrada:

- O automato começa com o estado 0 no topo da pilha (antes do símbolo inicial)
- Avança até não haver mais entrada
- No final, tem de ter o símbolo inicial no topo da pilha

Nota: A recursão pode causar um aumento da pilha.
No limite, toda a entrada está na pilha num dado momento.

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	L3
\$0 T	\$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	L3
\$0 T	\$	goto 2
\$0 T2	\$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	L3
\$0 T	\$	goto 2
\$0 T2	\$	R2
\$0 E	\$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	L3
\$0 T	\$	goto 2
\$0 T2	\$	R2
\$0 E	\$	goto 1
\$0 E1	\$	

Processo de análise

Processamento da entrada – Exemplo

Pilha	Entrada	Acções
\$0	id * (id + id) \$	L6
\$0 F	* (id + id) \$	L4
\$0 T	* (id + id) \$	goto 2
\$0 T2	* (id + id) \$	shift 7
\$0 T2 *7	(id + id) \$	shift 4
\$0 T2 *7 (4	id + id) \$	L6
\$0 T2 *7 (4 F	+ id) \$	L4
\$0 T2 *7 (4 T	+ id) \$	goto 2
\$0 T2 *7 (4 T2	+ id) \$	R2
\$0 T2 *7 (4 E	+ id) \$	goto 8
\$0 T2 *7 (4 E8	+ id) \$	shift 6
\$0 T2 *7 (4 E8 +6	id) \$	L6
\$0 T2 *7 (4 E8 +6 F	) \$	L4
\$0 T2 *7 (4 E8 +6 T	) \$	goto 9
\$0 T2 *7 (4 E8 +6 T9	) \$	R1
\$0 T2 *7 (4 E	) \$	goto 8
\$0 T2 *7 (4 E8	) \$	L5
\$0 T2 *7 F	\$	L3
\$0 T	\$	goto 2
\$0 T2	\$	R2
\$0 E	\$	goto 1
\$0 E1	\$	ACCEPT

Gramática:

$$S \rightarrow b\,S\,b$$
$$S \rightarrow b\,X\,a$$
$$S \rightarrow c$$
$$X \rightarrow c$$
$$X \rightarrow S\,b$$
$$X \rightarrow X\,c$$

Exercício:

- Construa o DFA para a gramática aumentada, usando os items LR(0)

Gramática:

$$S \rightarrow b\,S\,b$$
$$S \rightarrow b\,X\,a$$
$$S \rightarrow c$$
$$X \rightarrow c$$
$$X \rightarrow S\,b$$
$$X \rightarrow X\,c$$

Exercício:

- Construa o DFA para a gramática aumentada, usando os items LR(0)
- Construa a tabela de análise compactada

Gramática:

$$S \rightarrow b\,S\,b$$
$$S \rightarrow b\,X\,a$$
$$S \rightarrow c$$
$$X \rightarrow c$$
$$X \rightarrow S\,b$$
$$X \rightarrow X\,c$$

Exercício:

- Construa o DFA para a gramática aumentada, usando os items LR(0)
- Construa a tabela de análise compactada
- A gramática é passível de ser processada por um parser LR(0) ?

Gramática:

$$S \rightarrow b\,S\,b$$
$$S \rightarrow b\,X\,a$$
$$S \rightarrow c$$
$$X \rightarrow c$$
$$X \rightarrow S\,b$$
$$X \rightarrow X\,c$$

Exercício:

- Construa o DFA para a gramática aumentada, usando os items LR(0)
- Construa a tabela de análise compactada
- A gramática é passível de ser processada por um parser LR(0) ?
- E por um parser SLR(1) ?

Gramática:

$$S \rightarrow b\,S\,b$$
$$S \rightarrow b\,X\,a$$
$$S \rightarrow c$$
$$X \rightarrow c$$
$$X \rightarrow S\,b$$
$$X \rightarrow X\,c$$

Exercício:

- Construa o DFA para a gramática aumentada, usando os items LR(0)
- Construa a tabela de análise compactada
- A gramática é passível de ser processada por um parser LR(0) ?
- E por um parser SLR(1) ?
- Processe a entrada `bcca`

Questões?

Dúvidas?