Análise sintática

Função, interação com o compilador
Análise descendente e ascendente
Especificação e reconhecimento de cadeias de tokens válidas
Implementação
Tratamento de erros

Prof. Thiago A. S. Pardo

Análise sintática ascendente

- *Bottom-up*, ascendente ou redutiva
  - Analisadores de precedência de operadores
  - Analisadores LR
    - SLR: *Simple* LR
    - LR Canônico
    - *Look Ahead* LR: LALR

ASA: precedência de operadores

- Exercício: reconheça a expressão (id)

<E> ::= <E>/<T> | <T>
<T> ::= <T>&<F> | <F>
<F> ::= (<E>) | id

Pilha	Cadeia	Regra
$	(id)$	

Tabela sintática

	id	/	&	(	)	$
id		>	>		>	>
/	<	>	<	<	>	>
&	<	>	>	<	>	>
(	<	<	<	<	=	
)		>	>		>	>
$	<	<	<	<		

3

ASA: precedência de operadores

- Exercício: reconheça a expressão (id)

<E> ::= <E>/<T> | <T>
<T> ::= <T>&<F> | <F>
<F> ::= (<E>) | id

Pilha	Cadeia	Regra
$<	(id)$	empilha
$<(<	id)$	empilha
$<(<id>	)$	reduz
$<(=	)$	empilha
$<(=)>	$	reduz
$E	$	SUCESSO

Tabela sintática

	id	/	&	(	)	$
id		>	>		>	>
/	<	>	<	<	>	>
&	<	>	>	<	>	>
(	<	<	<	<	=	
)		>	>		>	>
$	<	<	<	<		

4

ASA: precedência de operadores

- Algorimo do ASA de precedência de operadores

Seja S o símbolo inicial da gramática, *a* o símbolo terminal mais ao topo da pilha e *b* o primeiro símbolo da cadeia de entrada

repita
    se ($S é o topo da pilha e $ é o primeiro símbolo da cadeia) então SUCESSO
    senão se (a<b ou a=b) então empilha b
    senão se (a>b) então
        desempilha até haver < entre o terminal do topo e o último desempilhado
    senão ERRO

5

ASA: precedência de operadores

- 2 métodos para construção da tabela sintática
  - Intuitivo: baseado no conhecimento da precedência e associatividade dos operadores
  - Mecânico: obtem-se a tabela diretamente da gramática

6

ASA: precedência de operadores

- Método intuitivo
  - Para 2 operadores quaisquer x e y

1. Se x tem maior precedência do que y, então tem-se x (na pilha) > y (na cadeia) e y (na pilha) < x (na cadeia)
   - Exemplo: como * tem maior precedência que +, então *>+ e +<*
2. Se x e y têm precedência igual (ou são iguais) e são associativos à esquerda, então tem-se x>y e y>x; se são associativos à direita, então tem-se x<y e y<x
   - Exemplo: como * e / têm a mesma precedência e são associativos à esquerda, tem-se *>/ e />*; como o operador de exponenciação ** é associativo à direita, tem-se **<**

7

ASA: precedência de operadores

3. As relações entre os operadores e os demais símbolos terminais (operandos e delimitadores) são fixas
   - Para qualquer operador x, tem-se x>$, $<x, x<id, id>x, x<(, (<x, x>) e)>x
4. As relações entre os operandos também são fixas
   - (<(,)>), id>), $<(, (=),)>$, id>$, $<id, (<id

8

ASA: precedência de operadores

- Exemplo: construir a tabela sintática para a gramática abaixo

  <E> ::= <E>+<E> | <E>*<E> | <E>**<E> | (<E>) | id

  sabendo-se que: ** tem maior precedência e é associativo à direita;
  * tem precedência intermediária e é associativo à esquerda; +
  tem menor precedência e é associativo à esquerda

	+	*	**	(	)	id	$
+							
*							
**							
(							
)							
id							
$							

9

ASA: precedência de operadores

- Exemplo: construir a tabela sintática para a gramática abaixo

  <E> ::= <E>+<E> | <E>*<E> | <E>**<E> | (<E>) | id

  sabendo-se que: ** tem maior precedência e é associativo à direita;
  * tem precedência intermediária e é associativo à esquerda; +
  tem menor precedência e é associativo à esquerda

	+	*	**	(	)	id	$
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
**	>	>	<	<	>	<	>
(	<	<	<	<	=	<	
)	>	>	>		>		>
id	>	>	>		>		>
$	<	<	<	<		<	OK

10

ASA: precedência de operadores

- Método mecânico: aplicável para gramáticas não ambíguas
  - Para os terminais *a* e *b*
    1. a=b se αaβbγ é lado direito de produção e β é λ ou um único símbolo não terminal
    2. a<b se αaXβ é lado direito de produção e X produz γbδ e γ é λ ou um único símbolo não terminal
    3. $<b se S produz γbδ e γ é λ ou um único símbolo não terminal
    4. a>b se αXbβ é lado direito de produção e X produz γaδ e δ é λ ou um único símbolo não terminal
    5. a>$ se S produz γaδ e δ é λ ou um único símbolo não terminal

ASA: precedência de operadores

- Em outras palavras
  - Um terminal *a* seguido imediatamente de um não terminal X tem precedência menor do que os primeiros símbolos terminais deriváveis a partir de X (precedidos de λ ou um não terminal)
  - Todos os últimos terminais que podem ser derivados a partir de um não terminal X (seguidos de λ ou um não terminal) têm precedência maior do que um terminal que segue imediatamente a X

ASA: precedência de operadores

- Exemplo: construir a tabela sintática para a gramática abaixo

  <E> ::= <E>+<E> | <E>*<E> | <E>**<E> | (<E>) | id

  Inicialmente, deve-se eliminar a ambiguidade da gramática (mantendo a precedência e a associatividade dos operadores)

  <E> ::= <E>+<T> | <T>
  <T> ::= <T>*<F> | <F>
  <F> ::= <P>**<F> | <P>
  <P> ::= id | (<E>)

13

ASA: precedência de operadores

Determinam-se, para cada não terminal, os primeiros e últimos terminais possíveis de ocorrerem em uma cadeia derivada a partir do não terminal

	Primeiros	**Últimos**
E	+ * ** (id	+ * **) id
T	* ** (id	* **) id
F	** (id	**) id
P	id (	id)

<E> ::= <E>+<T> | <T>
<T> ::= <T>*<F> | <F>
<F> ::= <P>**<F> | <P>
<P> ::= id | (<E>)

ASA: precedência de operadores

Para computar <, procurar pares aX nos lados direitos de produção; tem-se que a tem menor precedência do que qualquer primeiro terminal derivado a partir de X

Pares: +T *F **F (E

Relações: + < {*,**,(,id}
* < {**,(,id}
** < {**,(,id}
(< {+,*,**,(,id}

<E> ::= <E>+<T> | <T>
<T> ::= <T>*<F> | <F>
<F> ::= <P>**<F> | <P>
<P> ::= id | (<E>)

ASA: precedência de operadores

Para computar >, procurar pares Xb nos lados direitos de produção; tem-se que qualquer último terminal derivado de X tem precedência maior do que b

Pares: E+ T* P** E)

Relações: {+,*,**,),id} > +
{*,**,),id} > *
{),id} > **
{+,*,**,),id} >)

<E> ::= <E>+<T> | <T>
<T> ::= <T>*<F> | <F>
<F> ::= <P>**<F> | <P>
<P> ::= id | (<E>)

ASA: precedência de operadores

Para computar =, procurar aβb nos lados direitos das produções, onde β é λ ou um não terminal, e fazer a=b

Dada o lado direito (E), tem-se (=)

$ tem precedência menor do que todos os primeiros terminais deriváveis a partir do símbolo inicial da gramática

$ < {+,*,**,(,id

Todos os últimos terminais derivados a partir do símbolo inicial da gramática têm precedência maior do que $

{+,*,**,),id} > $

<E> ::= <E>+<T> | <T>
<T> ::= <T>*<F> | <F>
<F> ::= <P>**<F> | <P>
<P> ::= id | (<E>)

Exercício

- **Construir a tabela sintática** para a gramática abaixo pelo método mecânico e **reconhecer cadeia (a*b)**

S → (S O S) | a | b
O → + | *

18

Resposta

- Transformando a gramática

S → (S + S) | (S * S) | a | b

19

Resposta

- Primeiros e últimos

	Primeiros	Últimos
S	(a b	) a b

S → (S + S) | (S * S) | a | b

20

Resposta

- Relações <: construções do tipo aX
  - (S
  - + S
  - * S
    - (< {(,a,b}
    - + < {(,a,b}
    - * < {(,a,b}

	+	*	(	)	a	b	$
+			<		<	<	
*			<		<	<	
(			<		<	<	
)							
a							
b							
$							

	Primeiros	Últimos
S	(a b	) a b

S → (S + S) | (S * S) | a | b

21

Resposta

- Relações >: construções do tipo Xb
  - S +
  - S)
  - S *
    - {),a,b} > +
    - {),a,b} >)
    - {),a,b} > *

	+	*	(	)	a	b	$
+			<		<	<	
*			<		<	<	
(			<		<	<	
)	>	>		>			
a	>	>		>			
b	>	>		>			
$							

	Primeiros	Últimos
S	(a b	) a b

S → (S + S) | (S * S) | a | b

22

Resposta

- Relações =: construções do tipo aXb
  - (S +
  - + S)
  - (S *
  - * S)
    - (= +
    - + =)
    - (= *
    - * =)

	+	*	(	)	a	b	$
+			<	=	<	<	
*			<	=	<	<	
(	=	=	<		<	<	
)	>	>		>			
a	>	>		>			
b	>	>		>			
$							

	Primeiros	Últimos
S	(a b	) a b

S → (S + S) | (S * S) | a | b

23

Resposta

- Delimitadores
  - $ < primeiros de S
  - Últimos de S > $

	+	*	(	)	a	b	$
+			<	=	<	<	
*			<	=	<	<	
(	=	=	<		<	<	
)	>	>		>			>
a	>	>		>			>
b	>	>		>			>
$			<		<	<	

	Primeiros	Últimos
S	(a b	) a b

S → (S + S) | (S * S) | a | b

24

Resposta

- Reconhecendo cadeia **(a*b)**

S → (S + S) | (S * S) | a | b

	+	*	(	)	a	b	$
+			<	=	<	<	
*			<	=	<	<	
(	=	=	<		<	<	
)	>	>		>			>
a	>	>		>			>
b	>	>		>			>
$			<		<	<	

Pilha	Cadeia	Regra
$<	(a*b)$	empilha
$<(<	a*b)$	empilha
$<(<a>	*b)$	reduz
$<(=	*b)$	empilha
$<(=*<	b)$	empilha
$<(=*<b>	)$	reduz
$<(=*=	)$	empilha
$<(=*=)>	$	reduz
$S	$	OK

25

Questão

- E se a análise sintática pudesse ser modelada por autômatos de estados finitos?

26

Análise sintática ascendente

- *Bottom-up*, ascendente ou redutiva
  - Analisadores de precedência de operadores
  - Analisadores LR
    - SLR: *Simple* LR
    - LR Canônico
    - *Look Ahead* LR: LALR

27

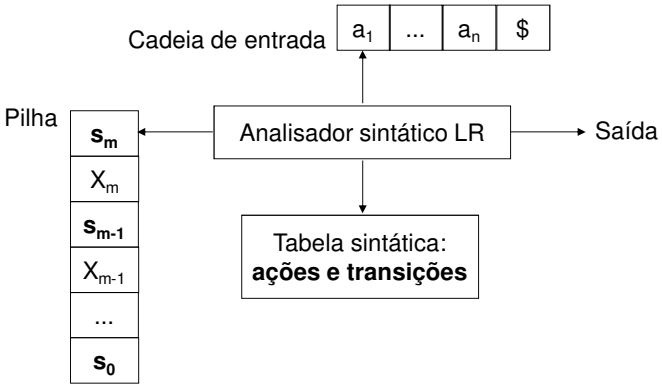
Analisadores LR

- LR(k): *Left to right, Rightmost derivation, with k lookahead symbols*
  - LR(1)
  - Ampla classe de gramáticas
- Método mais genérico e poderoso de análise
  - Podem reconhecer praticamente todas as construções possíveis em linguagens de programação
  - Implementação eficiente
  - Desvantagem: manipulação complexa da tabela sintática
- Yacc segue esta estratégia

28

Analisadores LR

- Esquema de um analisador LR
  - X_i são símbolos gramaticais
  - s_i são estados que sumarizam a informação contida abaixo na pilha



29

Analisadores LR

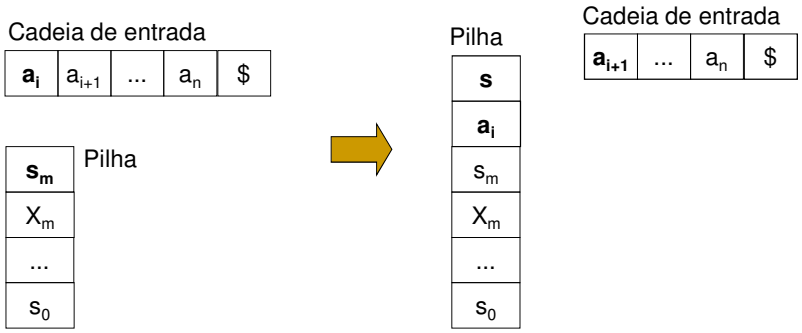
- A combinação do estado do topo da pilha e o primeiro símbolo da cadeia de entrada é utilizada para indexar a tabela sintática e determinar o que se faz (empilha ou reduz)

- Comportamento do analisador
  - Para o estado do topo da pilha s_m e o símbolo da entrada a_i, consuta-se ação[s_m,a_i] na tabela
    - Quatro possíveis valores
      - Empilhamento de um estado
      - Redução por uma regra gramatical
      - Aceitação da cadeia de entrada
      - Erro
  - Pela transição (ou "desvio") indicada na tabela, toma-se um estado e um símbolo gramatical e se produz um novo estado como saída

30

Analisadores LR

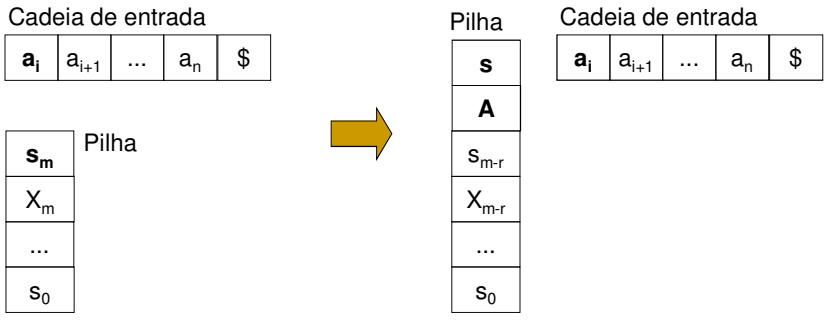
- Para o estado do topo da pilha s_m e o símbolo da entrada a_i, consuta-se ação[s_m,a_i]
  - Se ação[s_m,a_i]="empilhar s", então empilham-se a_i e s



31

Analisadores LR

- Se ação[s_m,a_i]="reduzir A→β", então
  - Sendo r o tamanho de β, desempilham-se 2r elementos (r estados e r símbolos gramaticais): o topo torna-se s_{m-r}
  - Empilha-se A (lado esquerdo da regra utilizada)
  - Empilha-se o estado s, indicado por transição[s_{m-r},A]



32

Analisadores LR

- Se ação[s_m,a_i]="aceitar", então a análise sintática tem sucesso e é encerrada
- Se ação[s_m,a_i]="erro", então o analisador descobriu um erro e deve tratá-lo

Analisadores LR

- Em outras palavras
  - Sempre deve haver um estado no topo da pilha
    - O estado diz tudo sobre a análise sintática
    - Símbolos gramaticais são dispensáveis na pilha, na realidade
  - Quando se empilha um símbolo gramatical, deve-se empilhar um estado em seguida
  - Ao se reduzir, (i) removem-se da pilha os símbolos gramaticais que correspondem ao lado direito do *handle* e os estados correspondentes e (ii) empilha-se o lado esquerdo do *handle* (i.e., um símbolo gramatical), (iii) gerando a necessidade de se empilhar mais um estado

Analisadores LR

- Algoritmo de análise LR

empilha-se o estado inicial s_0;
concatena-se o símbolo delimitador $ no final da cadeia de entrada
fazer ip apontar para o primeiro símbolo da cadeia
repetir
  seja s_n o estado no topo da pilha e a o símbolo apontado por ip
  se (ação[s_n,a]="empilhar s_{n+1}") então
    empilhar a;
    empilhar s_{n+1};
    avançar ip;
  senão se (ação[s_n,a]="reduzir A→β") então
    desempilhar 2*|β| elementos;
    empilhar A;
    empilhar o estado indicado por transição[$s_{n-2*|\beta|}$,A];
  senão se (ação[s_n,a]="aceitar") então SUCESSO
  senão ERRO;
fim-repetir;

35

Analisadores LR

- Reconhecer a cadeia id*id+id

(1) <E>::=<E>+<T>
(2) <E>::=<T>
(3) <T>::=<T>*<F>
(4) <T>::=<F>
(5) <F>::=(<E>)
(6) <F>::=id

Tabela sintática LR

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			

Na tabela, tem-se que:
- s*i* indica "empilhar *i*"
- r*i* indica "reduzir por regra *i*"

De onde vem esse 's'?

Analisadores LR

- Reconhecer a cadeia id*id+id

Pilha	Cadeia	Regra
0	id*id+id$	

37

Analisadores LR

- Reconhecer a cadeia id*id+id

Pilha	Cadeia	Regra
0	id*id+id$	s5
0id5	*id+id$	r6
0F3	*id+id$	r4
0T2	*id+id$	s7
0T2*7	id+id$	s5
0T2*7id5	+id$	r6
0T2*7F10	+id$	r3
0T2	+id$	r2
0E1	+id$	s6
0E1+6	id$	s5
0E1+6id5	$	r6
0E1+6F3	$	r4
0E1+6T9	$	r1
0E1	$	OK

38

Analisadores LR

- Exercício: reconhecer a cadeia (id)

Pilha	Cadeia	Regra
0	(id)$	

39

Analisadores LR

- Exercício: reconhecer a cadeia (id)

Pilha	Cadeia	Regra
0	(id)$	s4
0(4	id)$	s5
0(4id5	)$	r6
0(4F3	)$	r4
0(4T2	)$	r2
0(4E8	)$	s11
0(4E8)11	$	r5
0F3	$	r4
0T2	$	r2
0E1	$	OK

40

Analisadores LR

- Notem que
  - Transições são para símbolos não terminais
  - As transições para os símbolos terminais estão implícitas nas ações
  - O estado no topo da pilha oferece toda a informação sobre o *handle* encontrado
    - Não é preciso percorrer a pilha para encontrar o *handle*
    - Eficiência

41