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

Questão

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

Análise sintática ascendente

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

3

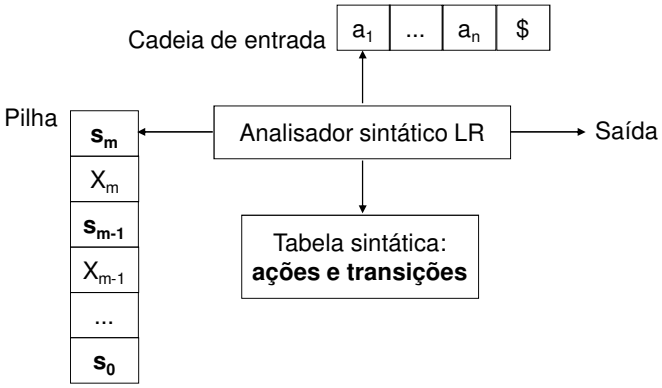
Analisadores LR

- 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

4

Analisadores LR

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



5

Analisadores LR

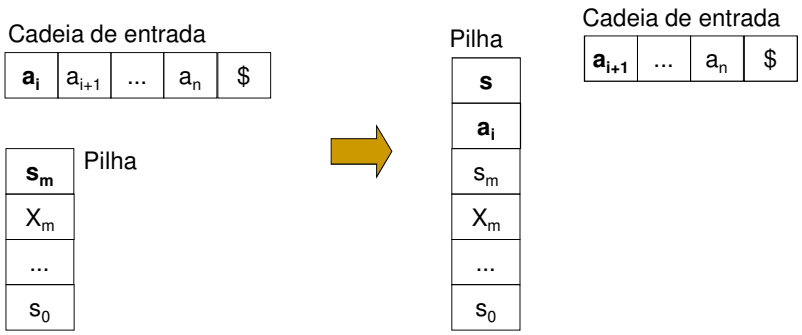
- A combinação do estado do topo da pilha e o primeiro símbolo da cadeia de entrada é utilizada para indexar a tabela sintática e determinar o que se faz (empilha ou reduz)

- 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

6

Analisadores LR

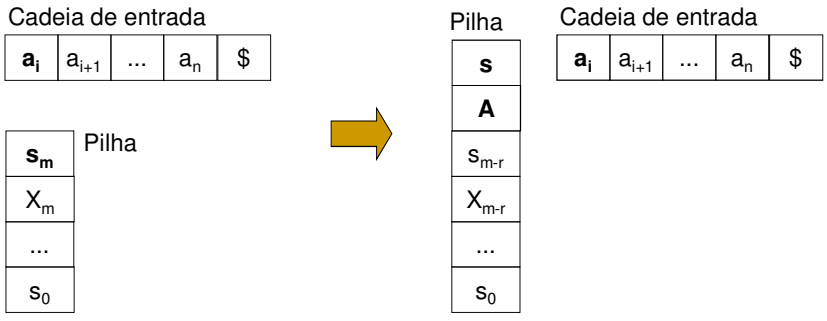
- Para o estado do topo da pilha s_m e o símbolo da entrada a_i, consuta-se ação[s_m,a_i]
  - Se ação[s_m,a_i]="empilhar s", então empilham-se a_i e s



7

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]



8

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;

11

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$	

13

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

14

Analisadores LR

- Exercício: reconhecer a cadeia (id)

Pilha	Cadeia	Regra
0	(id)$	

15

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

16

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

17

Analisadores LR

- Três técnicas para construir tabelas sintáticas para gramáticas LR
  - *Simple* LR (SLR)
    - Mais fácil de implementar, mas o menos poderoso
  - LR canônico
    - Mais complexo, mas mais poderoso
  - *Look Ahead* LR (LALR)
    - Complexidade e poder intermediários
- Tabelas possivelmente distintas para cada técnica, determinando o poder do analisador

18

Analisadores LR

- Gramáticas LR
  - Uma gramática é LR se é possível construir uma tabela sintática LR para ela
    - Não pode haver ambigüidade
- LL(k) vs. LR(k)
  - LL(k): decide-se por uma produção olhando-se apenas os k primeiros símbolos da cadeia de entrada
  - LR(k): reconhece-se o lado direito de uma produção tendo visto tudo que foi derivado a partir desse lado direito (na pilha) mais o esquadrinhamento antecipado de k símbolos (da cadeia de entrada)
    - Mais poderoso do que LL(k): pode descrever mais linguagens

19

Análise SLR

- A análise sintática por meio de uma tabela SLR é chamada análise sintática SLR
- Uma gramática é SLR se for possível construir uma tabela SLR para ela

20

Análise SLR

- A construção da tabela SLR se baseia no *conjunto canônico de itens LR(0)*
  - LR(0): não se olha nenhum símbolo a frente
- Um item para uma gramática G é uma regra de produção com alguma indicação do que já foi derivado/consumido na regra durante a análise sintática
  - Exemplo: A→XYZ
    - A→**.**XYZ
    - A→X**.**YZ
    - A→XY**.**Z
    - A→XYZ**.**
  - Regras do tipo A→λ geram somente um item A→**.**

Análise SLR

- Construção do conjunto canônico de itens
  - Duas operações
    1. Acrescentar à gramática a produção S'→S (em que S é o símbolo inicial da gramática)
       - Permite a identificação do fim da análise, mais especificamente, S'→S**.**
    2. Computar as funções *fechamento* e *transição* para a nova gramática

Análise SLR

- Função *fechamento*
  - Seja I um conjunto de itens LR(0)
    1. Todo item em I pertence ao fechamento(I)
    2. Se A→α.Xβ está em fechamento(I) e X→γ é uma produção, então adiciona-se X→.γ ao conjunto
  - Em outras palavras
    - Inicializa-se o conjunto I com as regras iniciais da gramática, colocando-se o indicador (.) no início de cada regra
    - Para cada regra no conjunto, adicionam-se as regras dos não terminais que aparecem precedidos pelo indicador (.)

Análise SLR

- Exemplo

S' → S
S → a | [L]
L → L;S | S

I={S→[.L]}

fechamento(I)=

Análise SLR

- Exemplo

  S' → S
  S → a | [L]
  L → L;S | S

  I={S→[.L]}

  fechamento(I)={S→[.L], L→.L;S, L→.S, S→.a, S→.[L]}

Análise SLR

- Função *transição*
  - *transição*(I,X): consiste avançar o indicador (.) através do símbolo gramatical X das produções correspondentes em I e calcular a função *fechamento* para o novo conjunto
- Exemplo

  I={S→[L.], L→L.;S}

  transição(I,;)=

Análise SLR

- Função *transição*
  - *transição*(I,X): consiste avançar o indicador (.) através do símbolo gramatical X das produções correspondentes em I e calcular a função *fechamento* para o novo conjunto
- Exemplo

  I={S→[L.], L→L.;S}

  transição(I,;)={L→L;.S, S→.a, S→.[L]}

27

Análise SLR

- Algoritmo para obter o conjunto canônico de itens LR(0)

  C:={I_0=fechamento({S'→S})}
  repita
    para cada conjunto I em C e X símbolo de G, tal que transição(I,X)≠λ
      adicione transição(I,X) a C
  até que todos os conjuntos tenham sido adicionados a C

28

Análise SLR

- Exemplo

0) S' → S
1) S → a
2) S → [L]
3) L → L;S
4) L → S

Análise SLR

- Exemplo

0) S' → S
1) S → a
2) S → [L]
3) L → L;S
4) L → S

Conjunto de itens

I_0 = {S'→.S, S→.a, S→.[L]}
transição(I_0,S) = {S'→S.} = I_1
transição(I_0,a) = {S→a.} = I_2
transição(I_0,[) = {S→[.L], L→.L;S, L→.S, S→.a, S→.[L]} = I_3
transição(I_3,L) = {S→[L.], L→L.;S} = I_4
transição(I_3,S) = {L→S.} = I_5
transição(I_3,a) = {S→a.} = I_2
transição(I_3,[) = {S→[.L], L→.L;S, L→.S, S→.a, S→.[L]} = I_3
transição(I_4,]) = {S→[L].} = I_6
transição(I_4,;) = {L→L;.S, S→.a, S→.[L]} = I_7
transição(I_7,S) = {L→L;S.} = I_8
transição(I_7,a) = {S→a.} = I_2
transição(I_7,[) = {S→[.L], L→.L;S, L→.S, S→.a, S→.[L]} = I_3

Análise SLR

- Exercício em duplas para entregar: construir o conjunto de itens para a gramática abaixo

S→if E then C | C
E→a
C→b