

Análise LR

Compiladores I

Gerson Geraldo Homrich Cavalheiro

Objetivo

- Caracterizar um método de análise bottom-up.
- Compreender os mecanismos básicos envolvidos no processo de análise LR.
- Oferecer as bases para utilização de uma ferramenta de geração de analisadores LR.

Gerson Geraldo H. Cavalheiro

Análise LR

2 / 30

Programa da aula

- Análise Bottom-Up
 - Revisão
- A Análise LR
 - Introdução
 - Estrutura geral e algoritmo básico
 - Gramática LR
 - Construção de tabelas sintáticas SLR
- Prática: Yacc

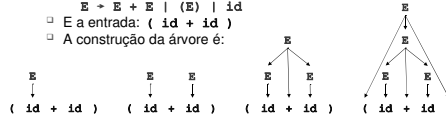
Gerson Geraldo H. Cavalheiro

Análise LR

3 / 30

A Análise Bottom-Up

- Princípio:
 - Construir o nó mais à esquerda de uma árvore de derivação que ainda não foi construído mas que possui seus filhos já construídos.
- Exemplo:
 - Para a gramática:
 $E \rightarrow E + E \mid (E) \mid id$
 - E a entrada: $(id + id)$
 - A construção da árvore é:



Gerson Geraldo H. Cavalheiro

Análise LR

4 / 30

A Análise Bottom-Up

- Possui diversas técnicas
 - Precedência de operadores
 - Utilizado em analisadores simples quando aplicado em situações que tenham semelhança com expressões aritméticas.
 - $BC(k,n)$
 - Popular nos anos 70, atualmente em desuso.
 - LR e suas variantes
 - Gramáticas LR têm sido as mais empregadas, em particular a sua variante LALR.

(Adaptado de Grune 2001)

Gerson Geraldo H. Cavalheiro

Análise LR

5 / 30

A Análise LR

Introdução

- LR – Analisador redutivo bottom-up
 - L: *left-to-right*
 - Varredura da entrada: da esquerda para a direita.
 - R: *rightmost derivation*
 - Produz a derivação mais à direita ao reverso, i.e., percorrendo as reduções de traz para frente, tem-se uma derivação mais à direita da cadeia de entrada.
- Aplicação
 - Gramáticas livres de contexto.
 - Geral e eficiente.
 - Ferramenta popular: Yacc.
 - Dada uma GLC, gera um analisador sintático LALR.

Gerson Geraldo H. Cavalheiro

Análise LR

(Adaptado de Grune 2001)

6 / 30

A Análise LR

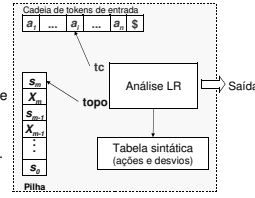
Introdução

- Métodos LR:
 - SLR – Simple LR;
 - LR Canônico;
 - LALR – Lookahead LR.
- SLR
 - Permite o estudo dos mecanismos empregados.
- Prática através do uso do Yacc (LALR).

A Análise LR

Estrutura geral e algoritmo básico

- Fita de entrada
- Pilha:
 - s_i : estado
 - X_i : símbolo da gramática
- Tabela sintática:
 - Duas partes: função de ações e função de desvio
 - Dado um par (topo, tc) indica a próxima ação.
- Programa diretor
- Saída



A Análise LR

Estrutura geral e algoritmo básico

- Tabela sintática
 - Linhas: estados
 - Colunas: terminais, não terminais
- | Estados | Ações | | Desvios |
|---------|-----------|---------------|---------|
| | terminais | não-terminais | |
| 0 | | | D |
| 1 | A | | S |
| 2 | C | | S |
| . | O | | V |
| . | E | | I |
| . | S | | O |
| . | | | S |
- (Adaptado de Price 2001)
- Endereçamento: $M[\text{topo}, \text{tc}]$
 - Resulta em uma AÇÃO, pode implicar em um DESVIO.

A Análise LR

Estrutura geral e algoritmo básico

- Comportamento do programa diretor
 - De acordo com:
 - tc : Token corrente (em curso de tratamento) e
 - Topo : Estado no topo da pilha
 - Consulta a tabela de sintática e determina a ação correspondente, a qual pode ser:
 - Empilhar
 - Reduzir
 - Aceitar
 - Erro
 - A execução destas ações modifica a configuração do analisador.

A Análise LR

Estrutura geral e algoritmo básico

- Configuração de um analisador LR é um par:

$$(s_0 X_1 s_1 \dots X_m s_m, a_1 a_{i+1} \dots an \$)$$
- Mudança de configuração:
 - Mapeamento do par (s_m, a_i) na tabela sintática identifica a ação.
- Considere que a configuração corrente seja:

$$(s_0 X_1 s_1 \dots X_m s_m, a_1 a_{i+1} \dots an \$)$$

as mudanças de configurações atingíveis são determinadas pelas ações executadas.

A Análise LR

Estrutura geral e algoritmo básico

- $(s_0 X_1 s_1 \dots X_m s_m, a_1 a_{i+1} \dots an \$)$
- Se $M[s_m, a_i] = \text{EMPILHAR } s$
Então novo estado é:

$$(s_0 X_1 s_1 \dots X_m s_m a_i s, a_1 a_{i+1} \dots an \$)$$

A Análise LR

Gramática LR

- Gramática para qual é possível construir uma tabela sintática.
- Existem GLC que não são LR, mas que podem ser evitadas nas construções típicas de linguagens de programação.
- Para uma GLC ser LR deve existir um analisador sintático de empilhar e reduzir que processando a entrada da esquerda para a direita seja capaz de reconhecer as produções do lado direito desta gramática, quando as mesmas surgirem no topo da pilha.
- Um analisador LR não precisa varrer toda a pilha para saber quando o lado direito de uma produção surge no topo: o símbolo de estado no topo da pilha contém a informação necessária.
- Em uma gramática LR(k), k símbolos de entrada (*lookahead*) podem ser utilizados para auxiliar o reconhecimento de uma produção.

A Análise LR

Construção de tabelas sintáticas SLR

- Baseia-se em um conjunto canônico de itens LR(0).
- Definição de item LR(0), ou simplesmente *item*:
 - Um item de uma gramática G é uma produção de G com um ponto em uma de suas posições do lado direito.
 - Para $A \rightarrow \epsilon$, $A \rightarrow \cdot$.
- Exemplo:
 - Produção: $A \rightarrow XYZ$
 - Itens: $A \rightarrow \cdot XY$
 - $A \rightarrow X \cdot Y$
 - $A \rightarrow XY \cdot$
 - $A \rightarrow X \cdot Z$

A Análise LR

Construção de tabelas sintáticas SLR

- Construção do conjunto canônico de itens LR(0) requer duas operações:
 - Estender a gramática com a produção: $S' \rightarrow S$
 - Computar as funções:
 - closure
 - gotosobre o conjunto de itens.

A Análise LR

Construção de tabelas sintáticas SLR

- closure(I)
 - Se I é um conjunto de itens de uma gramática G, o conjunto de itens closure(I) é construído a partir das regras:
 1. Todo item de I pertence a closure(I)
 2. Se $A \rightarrow \alpha X \beta$ pertence à closure(I) e $X \rightarrow \gamma$ é uma produção, então adicionar em closure(I) $X \rightarrow \gamma$
 - Exemplo:
 - $S \rightarrow a | [L]$
 - $L \rightarrow L ; S | S$
- Para $I = \{S \rightarrow [L]\}$,
closure(I) = $\{S \rightarrow [L], L \rightarrow .L ; S, L \rightarrow .S, S \rightarrow .a, S \rightarrow .[L]\}$

A Análise LR

Construção de tabelas sintáticas SLR

- goto(I,X) é formado:
 - pela produção que representa o avanço do ponto sobre X em I, e
 - pela função closure do conjunto resultante.
 - Para X terminal ou não terminal, goto(I,X), onde $A \rightarrow \alpha X \beta$ pertence a I, é dada pela função closure dos itens $A \rightarrow \alpha X \beta$.
 - Exemplo:
 - $S \rightarrow a | [L]$
 - $L \rightarrow L ; S | S$
- $I = \{S \rightarrow [L], L \rightarrow .L ; S\}$
goto(I,;) = $\{L \rightarrow L ; S, S \rightarrow .a, S \rightarrow .[L]\}$

A Análise LR

Construção de tabelas sintáticas SLR

- Algoritmo para obter o conjunto canônico LR(0)
 - Inicialização:
 - $C = \{I_0 = \text{closure}(\{S' \rightarrow \cdot S\})\}$
 - Repita:
 - Para todo I em C e todo X em G, tq goto(I,X) $\neq \emptyset$ adicione goto(I,X) à C.
 - Até que todos conjuntos tenham sido adicionados em C

A Análise LR

Construção de tabelas sintáticas SLR

Entrada: Conjunto $C = \{l_0, l_1, \dots, l_n\}$ para G' , e l_0 estado inicial

Saída: Tabela de análise SLR para G'

Algoritmo:

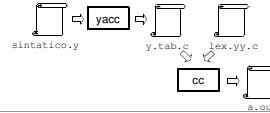
```
Para cada estado i
  Se goto(i,a) == l_j
    Então: AÇÃO[i,a] = EMPILHA j
  Se l contém A → α.
    Então Para todo Seguinte(A)
      AÇÃO[i,a] = REDUZ n // n == número da produção A → α.
  Se l contém S' → S
    Então AÇÃO[i,$] = ACEITA
  Se goto(i,A) = l_j
    Então DESVIO[i,A] = j
```

A Análise LR

Prática: Yacc

- Yacc: Yet Another Compiler Compiler
 - Início da década de 1970
 - S. C. Johnson
 - Popular nos ambientes Unix
 - Conhecidos como bison nas distribuições Linux atuais
 - Faz par com o Lex

- Fluxo de desenvolvimento com Yacc:



A Análise LR

Prática: Yacc

- Formato do arquivo

```
%{
Definições C
%}
Declarações de tokens
%%
Regras de tradução
%%
Rotinas de suporte
```

A Análise LR

Prática: Yacc

- Formato do arquivo – Exemplo (adaptado de Aho, 1995)

```
%(
#include <stdio.h>
#define YYSTYPE double
%)
%token NUMERO
left '+' '-'
left '*' '/'
%%
linhas : linhas expr '\n' { printf("%g\n", $2); }
        ; linhas '\n'
        ;
expr : expr '+' expr { $$ = $1 + $3; }
     ; expr '-' expr { $$ = $1 - $3; }
     ; expr '*' expr { $$ = $1 * $3; }
     ; expr '/' expr { $$ = $1 / $3; }
     ; NUMERO
%%
#include "yy.lex.c"
```

A Análise LR

Prática: Yacc

Exercícios

2. Estenda o exemplo de uso do Yacc para uma calculadora permitindo o uso de expressões com parênteses.
3. Incorpore o analisador léxico desenvolvido com Lex para reconhecimento de expressões aritméticas desenvolvido no módulo Análise Sintática com a calculadora desenvolvida no exercício 1.
4. Rastreie os movimentos que seriam feitos pela calculadora acima para reconhecimento de expressões de forma poder mostrar o processo de análise.
5. Implemente em Yacc uma gramática capaz de reconhecer comandos que possuam expressões do tipo if-then-else, considerando que pode ocorrer o aninhamento de expressões.

A Análise LR

Bibliografia

- Aho, A. V.; Sethi, R.; Ullman, J. D. **Compiladores: Princípios, Técnicas e Ferramentas**. Rio de Janeiro: LTC, 1995. 344p.
 - Referência sobre o assunto. Apresenta com detalhes os algoritmos trabalhados e uma extensa lista de exercícios. Apresenta uma introdução ao uso do Yacc. A maior parte deste material foi composta a partir desta referência.
- Price, A. M. A.; Toscani, S. S. **Implementação de Linguagens de Programação: Compiladores**. 2ª ed. Porto Alegre: Instituto de Informática da UFRGS/Sagra-Luzzatto, 2001. 196p.
 - Apresenta uma visão introdutória. Os algoritmos são apresentados de forma superficial. É um resumo suficiente da matéria.
- Grune, D.; Bal, H. E.; Jacobs, C. J. H.; Langendoen, K. G. **Projeto Moderno de Compiladores: Implementação e Aplicações**. Rio de Janeiro: Campus, 2001. 671p.
 - Encontra-se organizado de forma não usual. Seu ponto forte é ser bastante visual e pode apoiar a compreensão dos métodos.