Construção de Compiladores

Prof. Dr. Daniel Lucrédio
DC - Departamento de Computação
UFSCar - Universidade Federal de São Carlos

Tópico 06 - Análise Sintática Conclusão

Referências bibliográficas

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Compiladores: Princípios, Técnicas e Ferramentas (2a. edição). Pearson, 2008.

Kenneth C. Louden. Compiladores: Princípios E Práticas (1a. edição). Cengage Learning, 2004.

Terence Parr. The Definitive Antlr 4 Reference (2a. edição). Pragmatic Bookshelf, 2013.

ASA

- Melhor tipo de ASA: LALR
- Vantagens
  - Técnica poderosa
  - Gramáticas simples e elegantes
  - Eficiente
- Desvantagens
  - Exige manipulação complexa da tabela sintática
  - Construção trabalhosa (manual) de um analisador sintático LR para uma gramática típica de uma linguagem de programação
  - Conflitos reduce/reduce (pedra no sapato)
  - Resolução de conflitos é mais trabalhosa

ASD

- Melhor tipo de ASD: ALL(*)
- Vantagens: a técnica ALL(*) supera praticamente todas as limitações dos analisadores LL(1), LL(k) e LL(*)
  - Facilidade de escrita de regras
  - Facilidade no semântico
  - Facilidade de depuração
  - Praticamente qualquer gramática
  - Desempenho excelente
- Desvantagens:
  - Recursão indireta à esquerda não é resolvida

ALL(*) = ANTLR

- ANother Tool for Language Recognition
- Suporte a múltiplas linguagens
- Gera os tipos de analisadores mais fáceis de compreender e depurar
  (Analisador Sintático Preditivo de Descendência Recursiva)
- Um dos mais utilizados

Vamos vê-lo em prática?

Analisador sintático preditivo de descendência recursiva

Analisador sintático preditivo de descendência recursiva

- É o tipo mais simples
- É o preferido quando se constrói à mão

- Usa funções recursivas

Uma função para cada não-terminal

- Cada função é um espelho das regras de produção

Faz o “casamento” dos terminais
E chamadas para outros não-terminais

Analisador sintático preditivo de descendência recursiva

- Exemplo:

S → c A d

A → a b A | c

```
void match(token) {
// Testa se o símbolo atual
// casa com o token
// Se sim, avança a leitura
// Se não, acusa erro
}

token prox() {
// Retorna o próximo token
// sem avançar a leitura
// Ou seja, dá apenas uma
// "olhadinha" à frente
}
```

```
void S() {
   match("c");
   A();
   match("d");
}

void A() {
   if(prox() == "a") {
      match("a");
      match("b");
      A();
   }
   else if(prox() == "c") {
      match("c");
   }
   else { // erro sintático
   }
}
```

Analisador sintático preditivo de descendência recursiva

Vamos tentar implementar a linguagem ALGUMA

1. Manualmente
2. Usando o ANTLR

Fim