Compiladores

Análise Sintática

Bruno Lopes

Instituto de Computação
UFF

Parser bottom-up (ascendente)

Shift: Avança pela lista de *tokens*

Reduce: Aplica uma produção inversa

Instituto de Computação

```
0. E -> E + T
1. E -> E - T
2. E -> T
3. T -> T * F
4. T -> F
5. F -> -F
6. F -> int
7. F -> ( E )
```

- (int + int) - int

Instituto de Computação

Implementação

Uma pilha

- Topo da pilha é em função dos *tokens* de entrada
- Avança empilha o próximo token e incrementa a posição
- reduz desempilha símbolos e empilha não terminal

Instituto de Computação

Conflitos

- Técnicas de análise ascendente diferentes usam estratégias distintas para escolher entre as ações de avançar e reduzir
- Problemas na gramática ou na técnica podem levar a conflitos entre as ações a serem escolhidas

Conflito de avançar ou reduzir

Analisador não tem como decidir entre ações de avançar ou reduzir

Conflito de redução

Analisador não tem como decidir entre duas ou mais produções que podem ser usadas na redução

Poda do *Handle*!

Conflitos

- Técnicas de análise ascendente diferentes usam estratégias distintas para escolher entre as ações de avançar e reduzir
- Problemas na gramática ou na técnica podem levar a conflitos entre as ações a serem escolhidas

Conflito de avançar ou reduzir

Analisador não tem como decidir entre ações de avançar ou reduzir

Conflito de redução

Analisador não tem como decidir entre duas ou mais produções que podem ser usadas na redução

Poda do *Handle*!

Conflitos

- Técnicas de análise ascendente diferentes usam estratégias distintas para escolher entre as ações de avançar e reduzir
- Problemas na gramática ou na técnica podem levar a conflitos entre as ações a serem escolhidas

Conflito de avançar ou reduzir

Analisador não tem como decidir entre ações de avançar ou reduzir

Conflito de redução

Analisador não tem como decidir entre duas ou mais produções que podem ser usadas na redução

Poda do *Handle*!

Conflitos

- Técnicas de análise ascendente diferentes usam estratégias distintas para escolher entre as ações de avançar e reduzir
- Problemas na gramática ou na técnica podem levar a conflitos entre as ações a serem escolhidas

Conflito de avançar ou reduzir

Analisador não tem como decidir entre ações de avançar ou reduzir

Conflito de redução

Analisador não tem como decidir entre duas ou mais produções que podem ser usadas na redução

Poda do *Handle*!

```
token <- next_token()
repita até que o topo da pilha == Goal e token == EOF
  se o topo da pilha é um handle A→w
    // reduz w para A
    desempilha |w| símbolos
    empilha A
  senão se (token <> EOF)
    //avança
    empilha token
    token -> next_token()
  else reporta um erro
```

Instituto de Computação

```
0. Goal -> Expr
1. Expr -> Expr + Term
2. Expr -> Expr - Term
3. Expr -> Term
4. Term -> Term * Factor
5. Term -> Term / Factor
6. Term -> Factor
7. Factor -> number
8. Factor -> id
9. Factor -> ( Expr )
```

id - num * id

Instituto de Computação

Identificando *handles*

- Nenhum algoritmo eficiente conhecido consegue reconhecer *handles* no topo da pilha
- Solução: usar heurísticas para tentar adivinhar que topos de pilha são *handles* para certas classes de gramáticas
- A maioria examina a pilha e o próximo token de entrada (*lookahead*)

Instituto de Computação

Prefixos viáveis

- Prefixos de *handles*
- Enquanto o *parser* tiver um prefixo viável na pilha, ainda não foi detectado um erro de sintaxe
- O conjunto de prefixos viáveis de uma gramática é uma linguagem regular
- Podem-se utilizar Autômatos Finitos

Instituto de Computação

1. $S' \rightarrow S$
2. $S \rightarrow (S)S$
3. $S \rightarrow \epsilon$

Instituto de Computação

1. $E \rightarrow E + n$
2. $E \rightarrow n$

Instituto de Computação