Revisão de GLC e Analisadores Descendentes

Marcelo Johann

Conteúdo da aula

1. Exemplos de Gramáticas
2. Propriedades: Ambíguas, sem ciclos, ε-livres, fatoradas à esquerda, recursivas à esquerda, simplificadas
3. Transformações: Eliminação de produções vazias, de recursividade à esquerda (direta, indireta), fatoração
4. Analisadores Descendentes
   - Recursivos com Retrocesso
   - Recursivos Preditivos
   - Conjunto FIRST e Implementação

Exemplos

Gramática para Comandos

cmd → if expr then cmd else cmd | print
expr → (bool) | expr && expr | expr || expr
bool → true | false

Exemplos

Gramática para Expressões

E → E OP E | “(“ E “)” | x
OP → “*” | “/” | “+” | “-”

Exemplos

Gramática para Listas

S → a | “[“ L “]”
L → S “;” L | S

Ressaltando

- Convenções
  - Símbolos que representam terminais em minúsculos:
    - **u, v, x, y**, ...
  - Símbolos que representam não-terminais em maiúsculos:
    - **X, Y, TERM, S**,...
  - Símbolos que representam formas sentenciais (seqüências de terminais e não-terminais): letras gregas ou sublinhadas:
    - **α, β, ω, <u>w</u>, <u>z</u>**
- Outras…

Derivações

- Derivação em um passo

$$A \Rightarrow xy$$

- Derivação em múltiplos passos

  Se $A \Rightarrow w_2 \Rightarrow w_3 \ldots \Rightarrow w_n$ dizemos que

$$A \Rightarrow^* w_n$$

Tipos e Características

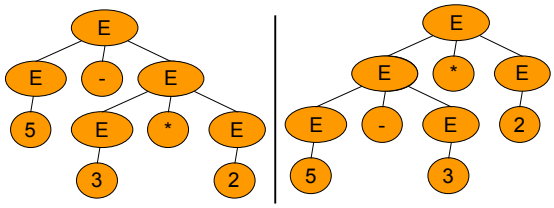
- Gramáticas Ambíguas
- Gramáticas sem ciclos
- Gramáticas ε-livres
- Gramáticas fatoradas à esquerda
- Gramáticas recursivas à esquerda
- Gramáticas simplificadas

Gramática Ambígua

E → E OP E | "(" E ")" | x

OP → "*" | "/" | "+" | "-"

- Considere 5 – 3 * 2



- Gramática sem ciclos:
  - Uma gramática sem ciclos é uma GLC que não possui derivações da forma

    $A \Rightarrow^+ A$ para algum $A \in N$

- Gramática ε-livre :
  - GLC que não possui produções vazias do tipo

    $A \rightarrow \varepsilon$

    exceto a produção $S \rightarrow \varepsilon$ (S é o símbolo inicial).

- Gramática fatorada à esquerda:
  - GLC que **não** possui produções do tipo $A \rightarrow \alpha\beta_1 | \alpha\beta_2$ para alguma forma sentencial α.

- Gramática recursiva à esquerda:
  - GLC que permite a derivação

    $A \Rightarrow^+ A\alpha$ para algum $A \in N$

    O não terminal A deriva ele mesmo, de forma direta ou indireta, como símbolo mais à esquerda de uma subpalavra gerada.

Transformações de GLCs

- (A) Eliminação de produções vazias
- (B) Eliminação de recursividade à esquerda:
  - recursão direta
  - recursão indireta
- (C) Fatoração de uma gramática

Eliminação de Produções Vazias

- Objetivo:
  - eliminar produções da forma $A \rightarrow \varepsilon$.
- Algoritmo: seja G = (N,T,P,S) uma GLC
  - Etapa 1:
    - construir N_ε , o conjunto de não-terminais que geram a palavra vazia:

    $N_\varepsilon = \{A \mid A \rightarrow \varepsilon\}$; // é um conjunto de símbolos

    Repita

    $N_\varepsilon = N_\varepsilon \cup \{X \mid X \rightarrow X_1...X_n \in P \text{ tq } X_1,...,X_n \in N_\varepsilon\}$

    Até que o cardinal de N_ε não aumente.

Eliminação de Produções Vazias

- Etapa 2:
  - construir o conjunto de produções sem produções vazias:

  gera $G_1 = (N,T,P_1,S)$, onde P_1 é construído como segue:

  $P_1 = \{A \rightarrow \alpha \mid \alpha \neq \varepsilon\}$;

  Repita

  Para toda produção $A \rightarrow \alpha \in P_1$ e $X \in N_\varepsilon$ tal que $\alpha = \alpha_1 X \alpha_2$ e $\alpha_1\alpha_2 \neq \varepsilon$

  Faça $P_1 = P_1 \cup \{A \rightarrow \alpha_1\alpha_2\}$

  Até que o cardinal de P_1 não aumente

Eliminação de Produções Vazias

- Etapa 3:
  - incluir a geração da palavra vazia, se necessário:
  - Se a palavra vazia pertence à linguagem, então a gramática resultante é

  $G_2 = (N,T,P_2,S)$, onde $P_2 = P_1 \cup \{S \rightarrow \varepsilon\}$

(B) Eliminação de recursividade à esquerda

- Exemplo de GLC recursiva à esquerda:

  $A \rightarrow Aa \mid b$

- Gramáticas transformadas equivalentes:

Com a palavra vazia	Sem a palavra vazia
$A \rightarrow bX$	$A \rightarrow b \mid bX$
$X \rightarrow aX \mid \varepsilon$	$X \rightarrow a \mid aX$

Obs: pode ainda haver recursão indireta!

(B) Outro exemplo de recursividade

- E -> E+T | T
- T -> T*F | F
- F -> (E)| **Id**

A regra E -> E+T | T se torna:

E -> TE'

E' -> +TE'| ε

A regra T- > T*F | F se torna:

T -> FT'

T'-> *FT' | ε

(C) Fatoração de uma gramática

- Elimina a indecisão de qual produção aplicar quando duas ou mais produções iniciam com a mesma forma sentencial

$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$

Se torna:

$A \rightarrow \alpha X$

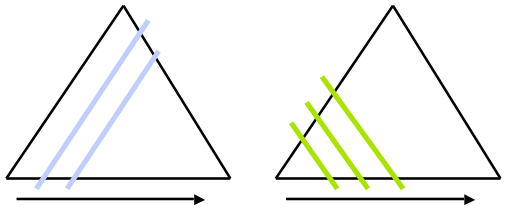
$X \rightarrow \beta_1 \mid \beta_2$

(C) Exemplo de Fatoração a Esquerda

Cmd → if Expr then Cmd else Cmd
Cmd → if Expr then Cmd
Cmd → Outro

- Fatorando a esquerda:
  Cmd → if Expr then Cmd ElseOpc
  Cmd → Outro
  ElseOpc → else Cmd | ε

Análise top-down e bottom-up



Analisadores Descendentes

- Também chamados recursivos ou top-down
- Recursivos com Retrocesso
  - Testam cada possível derivação
  - Exemplo
- Recursivos Preditivos
  - Determinam produção pelo símbolo terminal atual
  - Exemplo
- Conjunto FIRST e método de encontrá-lo
- Implementação

Exemplo

Gramática para Listas

S → a | "[" L "]"
L → S ";" L | S

Entrada: [a] $

Top-Down: Backtracking

S → A B
A → c | ε
B → cbB | ca

- Gera S ⇒* cbca?

S	cbca	tentar S→AB
AB	cbca	tentar A→c
cB	cbca	casar c
B	bca	sem-saída, tentar A→ε
εB	cbca	tentar B→cbB
cbB	cbca	casar c
bB	bca	casar b
B	ca	tentar B→cbB
cbB	ca	casar c
bB	a	sem-saída, tentar B→ca
ca	ca	casar c
a	a	casar a -> Fim!

Observações sobre o método recursivo com retrocesso

- É fácil de implementar.
- É necessário:
  1. Que a gramática não seja recursiva à esquerda
     - A → Aa se tornará
       ReconheceA() { ReconheceA();... }
     - Recursão infinita!

Analisadores Descendentes

- Também chamados recursivos ou top-down
- Recursivos com Retrocesso
  - Testam cada possível derivação
  - Exemplo
- Recursivos Preditivos
  - Determinam produção pelo símbolo terminal atual
  - Exemplo
- Conjunto FIRST e método de encontrá-lo
- Implementação

Exemplo

Gramática onde é fácil distinguir produções

CMD → “if” EXPR “then” CMD
CMD → “while” EXPR “do” CMD
CMD → “repeat” LISTA “until” EXPR
CMD → ID “:=“ EXPR

Exemplo

Mesma Gramática com mais Não-Terminais

CMD → COND | ITER | ATRIB
COND → “if” EXPR “then” CMD
ITER → “while” EXPR “do” CMD |
“repeat” LISTA “until” EXPR
ATRIB → ID “:=“ EXPR

Exemplo

Outra Gramática

CMD: IF | ASS | WHILE | BL;
BL: ´{´ CMDL ´}´;
CMDL: CMD RESTO;
RESTO: ´;´ CMDL | epsilon;
IF: "if" E "then" CMD;
WHILE: "while" E "do" CMD;
ASS: L ´=´ E;
L: ´*´ E | E;
E: NUM | ID;

Definição: Conjuntos “First”

- Seja α qualquer seqüência de símbolos
  - terminais ou não terminais
- First(α):
  - Definição informal:
    - conjunto de todos os terminais que começam qualquer seqüência derivável de α.
  - Definição formal:
    - Se existe um t ∈ **T** e um β ∈ **V*** tal que α ⇒* t β então t ∈ First(α)
    - Se α ⇒* ε então ε ∈ First(α)

A → B | C | D	First(A)={b,c,d}
B → b	First(B)={b}
C → c	First(C)={c}
D → d	First(D)={d}

Condição para que se possa usar um analisador preditivo

- Informalmente: no caso que em os First() dos lados direitos das regras de produção sejam “simpáticos”, não terá retrocesso.
- Formalmente: para qualquer produção
  $A \rightarrow \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_n$,
  quer-se:
  $First(\alpha_1) \cap First(\alpha_2) \cap ... \cap First(\alpha_n) = \emptyset$

proc First(α: string of symbols)

```
Repeat {
  Para todas as produções α → X1 X2 X3 ... Xn do
    if X1 ∈T then  // caso simples onde X1 é um terminal
        First(α) := First(α) ∪ {X1}
    else {         // caso menos simples: X1 é um não-terminal
            First(α) = First(α) ∪ First{X1} \ {ε};
            for (i=1 ; i<=n ; i++) {
              if ε is in First(X1) and in First(X2) and in… First(Xi-1)
                        First(α) := First(α) ∪ First(Xi) \ {ε}
            }
    }
    if (α =>* ε)
       then First(α) := First(α) ∪ {ε}
  end do
} until no change in any First(α)
```


Observações sobre o método recursivo preditivo

- É fácil de implementar.
- É necessário:
  1. Que a gramática não seja recursiva à esquerda
     - A → Aa se tornará
       ReconheceA() { ReconheceA();... }
     - Recursão infinita!
  2. Que a gramática seja fatorada à esquerda
     - Senão, há ambigüidade na escolha da derivação.
  3. Que os primeiros terminais deriváveis possibilitem a decisão de uma produção a aplicar!
     - Não há retrocesso sobre não-terminais...

Implementação

- CMD: IF | ASS | WHILE | BL;
- BL: '{' CMDL '}';
- CMDL: CMD RESTO;
- RESTO: ';' CMDL;
- IF: "if" E "then" CMD;
- WHILE: "while" E "do" CMD;
- ASS: L '=' E;
- L: '*' E | E;
- E: NUM | ID;

Etapas…

…voltando ao yacc

Leituras e Tarefas sugeridas

Ler e Reler o Livro
Implementar outra GLC
Ligar sua implementação com yylex
Ligar com autômato