PCS3616

Programação de Sistemas

(Sistemas de Programação)

Semana 11, Aula 19

Linguagens e Compiladores

Programação em linguagem de alto nível

Escola Politécnica da Universidade de São Paulo

Linguagens e Gramáticas

- Terminologia básica e simbologia
- Gramáticas – hierarquia de Chomsky
- Reconhecedores
- Reconhecedores (não-)determinísticos
- Transdutores
- Notações para definição de linguagens
- Técnicas de construção de reconhecedores

Gramáticas

- Gramáticas permitem gerar as sentenças das linguagens, e são caracterizadas como quádruplas ordenadas, da seguinte forma:
  - G = (V, Σ, P, S) – ou G = (N, T, P, S)
    - V representa o vocabulário da gramática G
    - Σ representa alguns dos elementos de V, que são os os símbolos ou átomos da linguagem (símbolos terminais), N = (V - Σ) (símbolos não-terminais)
    - P representa o conjunto de todas as leis de formação da gramática (que têm a forma $\alpha \rightarrow \beta$)
    - S é um elemento de N, cuja propriedade é ser o símbolo não-terminal que dá início ao processo de geração de sentenças (derivação)

Hierarquia de Chomsky

- Gramáticas irrestritas (tipo 0): Nenhuma limitação é imposta, as produções são do tipo: $\alpha \rightarrow \beta$, com $\alpha \in V^* N V^*$ e $\beta \in V^*$
  - Ex: G=({A,B,C,a,b}, {a,b}, {A→BC, BC→CB, B→b, C→a}, A)
- Gramáticas sensíveis ao contexto (tipo 1): impõe-se a restrição de que nenhuma substituição possa reduzir o comprimento da forma sentencial que se está aplicando, assim: $\alpha \rightarrow \beta$, com $|\alpha| \leq |\beta|$, onde $\alpha \in V^* N V^*$ e $\beta \in V^*$
  - Ex: G=({A,B,C,a,b}, {a,b}, {A→AB, AB→AC, C→aba}, A)

Hierarquia de Chomsky (cont.)

- Gramáticas livres de contexto (tipo 2): impõe-se mais uma restrição às produções, tornando-as da forma: $A \rightarrow \alpha$, onde $A \in N$, $\alpha \in V^*$.
  - Ex: $G=(\{A,B,C,a,b\}, \{a,b\}, \{A \rightarrow BC, B \rightarrow CB, B \rightarrow b, C \rightarrow a\}, A)$
- Gramáticas regulares (tipo 3): impõe-se mais uma restrição às produções, tornando-as da forma: $A \rightarrow \alpha B$ ou $A \rightarrow B\alpha$ (XOR); e $A \rightarrow \alpha$, onde $\alpha \in \Sigma^*$; $A, B \in N$.
  - Ex: $G=(\{S,a,b\}, \{a,b\}, \{S \rightarrow aS\}, S \rightarrow b\}, S)$

Reconhecedores

- Alternativa para a definição de uma linguagem, na qual utiliza-se dispositivos aceitadores denominados reconhecedores da linguagem.
- Um reconhecedor é dispositivo conceitual composto de texto de entrada (cadeia de símbolos), cursor de leitura, controle (máquina finita de estados) que pode usar uma memória auxiliar
- Há dois tipos de reconhecedores: determinísticos (um único movimento para qualquer configuração) e não-determinísticos (conjunto de movimentos possíveis para cada configuração)
- Um reconhecedor pode definir uma linguagem (quando só aceita cadeias desta linguagem)

Tipos de Linguagem e Reconhecedores

Linguagem	*Gramática*	*Reconhecedor*
tipo 0: conjuntos recursivamente enumeráveis	tipo 0: gramáticas irrestritas	Máquina de Turing
tipo 1: sensíveis ao contexto	tipo 1: gramáticas sensíveis ao contexto	Máquinas de Turing com fita limitada
tipo 2: livres de contexto	tipo 2: gramáticas livres de contexto	Autômatos de Pilha
tipo 3: conjuntos regulares	tipo 3: gramáticas regulares	Autômatos Finitos

Reconhecedores Usuais

- Autômatos Finitos
- Autômatos de Pilha
- Autômatos Adaptativos
- Máquina de Turing

Transdutores

- São reconhecedores que apresentam um alfabeto de saída e uma função de geração de saídas
- Também podem representar as linguagens
- Há classes de transdutores
  - Finitos: oriundo dos Autômatos Finitos
  - De Pilha: oriundo dos Autômatos de Pilha
  - Máquina de Turing: funciona como um transdutor

Notações para Definição de Linguagens

- Diagramas de sintaxe
- Metalinguagens
  - BNF
  - Notação de Wirth (EBNF)
- Expressões regulares
- Produções gramaticais
- Diagrama de estados

Backus-Naur Form (BNF)

<x> - representa um não terminal de nome x

::= - associa um não terminal a um conjunto de cadeias ("define-se como")

| - símbolo separador ("ou")

X - representa um terminal da linguagem sendo definida

ε - representa a cadeia vazia

yz - representa uma cadeia construída por concatenação de y e z nesta ordem

<A> ::= <B> a | <C> b | <A> c

<B> ::= <C> c | <B> a

<C> ::= <C> b | c

Notação de Wirth (EBNF)

- Segue BNF, simplificando em termos de legibilidade.
- Elimina <X>, deixando X (não-terminais)
- Impõe “x” no lugar de x (terminais)
- [z] construção opcional
- {z} iterações arbitrárias da construção z
- (z) o mesmo que z
- zt concatenação das construções z, e t
- z | t simboliza que z e t são alternativas válidas
- = substitui ::=
- . delimita o final de um regra

Técnicas de Construção de Reconhecedores

- Encontrar um mapeamento adequado
- Manipulação de definições formais
  - Conversão de notação (Ex.: BNF $\rightarrow$ diagrama de estados)
- Mapeamento de gramáticas em reconhecedores
  - Montar árvore de produções
  - Obter autômatos finitos
  - Obter autômatos de pilha

Bibliografia (Compiladores)

- "Introdução à Compilação", J. J. Neto, LTC, 1987 (**Livro-Base**)
- "Compilers: Principles, Techniques and Tools", Alfred Aho, Ravi Sethi and Jeffrey Ullman, Addison-Wesley, 2007
- "Introdução à compilação", I. Ricarte, ed Campus, 2008
- "Projeto Moderno de Compiladores – Implementação e Aplicações", D. Grune, H. Bal, C. Jacobs, K. G. Langendoen, ed. Campus, 2002
- "Compiladores: princípios e práticas", K. C. Louden, ed. Thompson, 2004
- "Implementação de Linguagens de Programação: Compiladores", A. M. A. Price & S. S. Toscani, ed. Sagra-Luzzatto, 2002
- Trembley & Sorenson – The theory and practice of compiler writing, McGraw-Hill, 1985
- Robert W. Sebesta – Programming language concepts, Addison-Wesley, 2007.