Sintaxe de Linguagens de Programação

Sérgio Queiroz de Medeiros
sergio@ufs.br

22 de março de 2012

Sintaxe e Semântica

- Uma linguagem de programação deve ser precisa
- Linguagem precisa = Sintaxe bem definida + Semântica bem definida
- Como definir sintaxe?
  - Análise léxica $\rightarrow$ Expressões Regulares
  - Análise sintática $\rightarrow$ Gramáticas Livres de Contexto (CFGs)
- Como definir semântica?
  - Manuais
  - Semânticas formais

Análise léxica: Tokens

- Tokens são elementos básicos de um programa:
  - Palavras-chave: *if*, *while*
  - Símbolos: *+*, *{*, *==*
  - Identificadores: *bola*, *f_99*
  - Literais: *99*, *"plp"*, *3.14*

Tokens e suas variações

- Existem linguagens *case sensitive* e *case insensitive*
  - Case sensitive: C, Java, Lua
  - Case insensitive: Pascal, Ada, Commom Lisp
- Algumas linguagens (e.g., C#, Java, Modula-3) possuem convenções para o uso de letras maiúsculas e minúsculas em identificadores
- Identificadores válidos:
  - bola
  - _bola
  - bol@
  - bolão

Questões de formatação

- Na maioria das linguagens modernas a formatação (espaços, tabulações, quebras de linha) geralmente não é importante
- Até Fortran 90, uma linha do programa não podia ter mais de 72 caracteres (o comprimento de um cartão perfurado)
- Em algumas linguagens quebras de linha são separadores de comando (e.g., Haskell, Tcl, Python)
- Às vezes a identação do programa delimita os comandos de um bloco (e.g., Haskell, Python)i
  - Análise léxica/sintática mais complexa
  - Problemas com a reformatação do texto por ferramentas

Expressões Regulares

- Formalismo usado para descrever tokens
- Analisador léxico é gerado a partir das expressões regulares (lex)
  - Autômato Finito Não-Determinístico (NFA)
  - Autômato Finito Determinístico (DFA)
  - Minimização de DFA
- Alguns elementos léxicos não podem ser descritos através de expressões regulares (e.g., comentários aninhados)

Analisadores léxicos na vida real

- Em compiladores reais, o analisador léxico geralmente é escrito manualmente, sem o auxílio de uma ferramenta
  - Eficiência
  - Independência de ferramenta
- Protótipos de compiladores reais costumam usar uma ferramenta (lex, JFlex) para gerar o analisador léxico
  - Rapidez no desenvolvimento

Pragmas

- Dicas para o compilador
- Informações específicas sobre a plataforma
- Melhorar a eficiência/uso da memória

```
register int i;
for(i=0; i<1000000000; i++) {
    ...
}

struct PackedStructure __attribute__((__packed__))
{
  char a;
  int b;
  short c;
};
// sizeof(PackedStructure == 7)
```

Análise Sintática

- Chama o analisador léxico para obter os tokens do programa
- Gramática Livre de Contexto (CFG)
  - Usada pelo programador para gerar um programa
  - Usada pelo parser para reconhecer um programa

Métodos de Análise Sintática

- Descendente (Top-Down)
  - Descendente recursivo
  - LL(1)
- Ascendente (Bottom-Up)
  - SLR(1)
  - LALR(1)
  - LR(1)

Referências

- Programming Language Pragmatics (Michael Scott)
  - Capítulo 2