Formal languages and Automata

形式语言与自动机

Chapter5 CONTEXT-FREE LANGUAGES

Beijing University of Posts and Telecommunications

北京邮电大学

刘咏彬 liuyb@bupt.edu.cn

2023.11

5.3 CONTEXT-FREE GRAMMARS AND PROGRAMMING LANGUAGES

- Importance of Formal Languages:
  - Defining programming languages precisely.
  - Foundation for efficient and reliable interpreters and compilers.
- Role of Regular and Context-Free Languages:
  - Regular languages: Recognizing simple patterns in programming languages.
  - Context-Free languages: Modeling complex aspects of programming languages.

5.3 CONTEXT-FREE GRAMMARS AND PROGRAMMING LANGUAGES

Defining Programming Languages with Backus-Naur Form (巴克斯范式):

- Backus–Naur form or Backus normal form (BNF) is a metasyntax notation for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets and communication protocols.
- Use explicit variable identifiers（有明确含义的变量标识符） in BNF for clarity
- Example 5.12 converted into BNF:

$$\langle \text{expression} \rangle ::= \langle \text{term} \rangle \mid \langle \text{expression} \rangle + \langle \text{term} \rangle ,$$

$$\langle \text{term} \rangle ::= \langle \text{factor} \rangle \mid \langle \text{term} \rangle * \langle \text{factor} \rangle$$

- The while statement in C can be defined as below, which looks like an s-grammar（简单文法） production.

$$\langle \text{while statement} \rangle ::= \text{while } \langle \text{expression} \rangle \langle \text{statement} \rangle$$

- Keywords (关键词) not only provide some visual structure that can guide the reader of a program, but also make the work of a compiler much easier.

Definition of Python

https://docs.python.org/3/reference/introduction.html#notation
https://docs.python.org/3/reference/compound_stmts.html#the-if-statement

1.2. Notation

The descriptions of lexical analysis and syntax use a modified Backus–Naur form (BNF) grammar notation. This uses the following style of definition:

```
name      ::=  lc_letter (lc_letter | "_")*
lc_letter ::=  "a"..."z"
```

The first line says that a `name` is an `lc_letter` followed by a sequence of zero or more `lc_letter`s and underscores. An `lc_letter` in turn is any of the single characters `'a'` through `'z'`. (This rule is actually adhered to for the names defined in lexical and grammar rules in this document.)

The `if` statement is used for conditional execution:

```
if_stmt ::=  "if" assignment_expression ":" suite
             ("elif" assignment_expression ":" suite)*
             ["else" ":" suite]
```

It selects exactly one of the suites by evaluating the expressions one by one until one is found to be true (see section Boolean operations for the definition of true and false); then that suite is executed (and no other part of the `if` statement is executed or evaluated). If all expressions are false, the suite of the `else` clause, if present, is executed.

```
assignment_expression ::=  [identifier ":="] expression
```

An assignment expression (sometimes also called a “named expression” or “walrus”) assigns an `expression` to an `identifier`, while also returning the value of the `expression`.

5.3 CONTEXT-FREE GRAMMARS AND PROGRAMMING LANGUAGES

- Limitations of S-Grammars in Programming Languages
  - Not all programming language features can be expressed by s-grammars.
- Use of LL and LR Grammars in Compilers （编译器）
  - LL and LR grammars are extensively used for their ability to express complex features and allow linear time parsing.
  - These grammars handle less obvious features of programming languages effectively.
- Importance of Unambiguous Language Specification
  - Algorithms for detecting and removing ambiguities in context-free grammars are complex and often difficult.
  - Deciding inherent ambiguity in a context-free language is a challenging task.

Semantics definition of Programming Languages

- Context-free grammar typically models a language's syntax.
- Challenges in Defining Programming Language Semantics

  Context-free grammars cannot capture semantic rules like type constraints.

  For C, the usual BNF definition allows constructs such as

  $$\mathrm{char\ a,\ b,\ c;}$$

  followed by

  $$\mathrm{c = 3.2;}$$

END