编译原理

北方工业大学信息学院
North China University of Technology Computer Science
束劼
shujie@ncut.edu.cn
瀚学楼1122，88801615

第二章 高级语言及其语法描述

第二章 高级语言及其语法描述

- 本章目录

第二章 高级语言及其语法描述

- 大纲要求

1. 了解：文法的分类；闭包、正闭包、文法、语言、推导、语法树的概念。

2. 熟悉：语法、语义，文法与语言的关系。

3. 掌握：上下文无关文法，语法分析树与二义性。

2.1 程序语言的定义

2.1 程序语言的定义

- 程序语言的定义
  - 语言是由句子组成的集合，是由一组符号所构成的集合。
    汉语--所有符合汉语语法的句子的全体
    英语--所有符合英语语法的句子的全体
    程序设计语言--所有该语言的程序的全体
  - 研究语言
    每个句子构成的规律
    每个句子的含义
    每个句子和使用者的关系

2.1 程序语言的定义

- 程序语言的定义
  - 语言定义是语言实现的基础。
  - 程序语言由两方面定义
    - **语法：**一组规则，用它可以形成和产生一个合式(well-formed)的程序。（词法规则、语法规则）
    - **语义：**语言成分的含义，由程序执行的效果来说明。

2.1 程序语言的定义

- 程序语言的定义
  - **语法：**一组规则，用它可以形成和产生一个合式(well-formed)的程序。（词法规则、语法规则）

    例如：赋值语句由一个变量，后随一个符号“=”，再在后面跟一个表达式所构成。A=B+C*60。
  - **语义：**语言成分的含义，由程序执行的效果来说明。

    例如：先对该语句的右部表达式求值，然后把所得结果与语句左部的变量相结合，并取代该变量原有的值。A=B+C*60，假如B=1, C=1, 则A=61。

2.1 程序语言的定义

- 程序语言的定义
  - **语法** = 词法规则 + 语法规则
  - **词法规则**：单词符号的形成规则。
    单词符号是语言中具有独立意义的最基本结构。一般包括：常数、标识符、基本字、算符、界符等。
    描述工具：有限自动机。
  - **语法规则**：语法单位的形成规则。
    语法单位通常包括：表达式、语句、分程序、过程、函数、程序等;
    描述工具：上下文无关文法(Context-free)。

9

2.1 程序语言的定义

- 程序语言的定义
  - **语义**：对于一个语言，不仅要定义它的词法和语法规则，还要定义它的单词符号和语法单位的意义。
    例如：X+F(X)+Y 它的语义是什么？

    这是一个算术表达式，但不同程序中，其含义相差很大。ALOGL中规定按左结合的规则计算，FORTRAN则容许使用交换律和结合律来计算。

    程序语言基本功能是描述数据和对数据的运算

10

2.2 高级语言的一般特性

2.2 高级语言的一般特性

- 高级语言的分类
  - **时代(Generation)分类**

  第一代 机器语言

  第二代 汇编语言

  第三代 高级语言， Fortran, C, C++, C#, Java等等

  第四代 基于应用的语言，SQL数据库语言等等

2.2 高级语言的一般特性

- 高级语言的分类
  - **命令(imperative)式语言**

    C, C++, C# and Java等等　面向对像的语言
  - **功能式(functional)语言**

    ML and Haskell
  - **说明式(declarative)语言**

    Prolog　基于规则的语言

13

2.2 高级语言的一般特性

- 数据类型与操作
  - **初等数据类型**

    数值数据，逻辑数据，字符数据，指针类型

    名字（名字，属性）
  - **数据结构**

    数组，记录，字符串、表格、栈和队列
  - **抽象数据类型**

    表达式和语句（赋值句、控制句、说明句、简单句和复合句）

14

2.3 程序语言的语法描述

2.3 程序语言的语法描述

- 2.3.1 上下文无关文法
- 2.3.2 语法分析树与二义性
- 2.3.3 形式语言鸟瞰

前言：语言的历史与发展

2.3 程序语言的语法描述

- 语言的描述方法——现状
  - 自然语言：自然、方便-非形式化
  - 数学语言（符号）：严格、准确-形式化
  - 形式化描述
    - 高度的抽象，严格的理论基础和方便的计算机表示。

2.3 程序语言的语法描述

- 语言学家Chomsky最初从产生语言的角度研究语言。
  - 1956年，通过抽象，他将语言形式地定义为是由一个字母表中的字母组成的一些串的集合。可以在字母表上按照一定的规则定义一个文法（Grammar），该文法所能产生的所有句子组成的集合就是该文法产生的语言。
- 克林（Kleene）在1951年到1956年间，从识别语言的角度研究语言，给出了语言的另一种描述。
  - 克林是在研究神经细胞中，建立了自动机，他用这种自动机来识别语言：对于按照一定的规则构造的任一个自动机，该自动机就定义了一个语言，这个语言由该自动机所能识别的所有句子组成。

19

2.3 程序语言的语法描述

- 1959年，Chomsky通过深入研究，将他本人的研究成果与克林的研究成果结合了起来，不仅确定了文法和自动机分别从生成和识别的角度去表达语言，而且证明了文法与自动机的等价性。
- 20世纪50年代，人们用巴科斯范式（Backus Nour Form 或 Backus Normal Form，简记为BNF）成功地对高级语言ALGOL-60进行了描述。
- 实际上，巴科斯范式就是上下文无关文法（Context Free Grammar）的一种表示形式。这一成功，使得形式语言在20世纪60年代得到了大力的发展。

20

2.3 程序语言的语法描述

- 语言——形式化的内容提取
  - 语言(Language)：满足一定条件的句子集合
  - 句子(Sentence)：满足一定规则的单词序列
  - 单词(Token)：满足一定规则的字符(Character)串
- 程序设计语言——形式化的内容提取
  - 程序设计语言(Programming Language)：组成程序的所有语句的集合。
  - 程序(Program)：满足语法规则的语句序列。
  - 语句(Sentence)：满足语法规则的单词序列。
  - 单词(Token)：满足词法规则的字符串。

21

2.3 程序语言的语法描述

- 描述形式——文法
  - 语法——语句
    - 语句的组成规则
    - 描述方法：BNF范式、语法(描述)图
  - 词法——单词
    - 单词的组成规则
    - 描述方法：有限自动机、正规式

22

前言：几个基本定义

2.3 程序语言的语法描述

- 符号Σ
  - ① **Σ:**一个有穷字母表，它的每个元素称为一个符号
  - ② **Σ上的一个符号串：**部分字母组成的符号串

- 符号ε(epsilon ['epsɪlɒn])
  空符号串,不包含任何符号的序列

2.3 程序语言的语法描述

- 符号Σ^*

  Σ^* : $\Sigma+\varepsilon$

  例如：$\Sigma=\{a,b\}$

  $\Sigma^*=\{\varepsilon,a,b,aa,bb,ab,ba,aaa,\ldots\}$

- 符号$\emptyset$

  空集，不是ε，也不是$\{\varepsilon\}$符号

2.3 程序语言的语法描述

- 符号 Σ^*

  Σ^*的子集U和V的**连接（积）**定义为

  $UV=\{\ \alpha\beta \mid \alpha\in U\ \&\ \beta\in V\ \}$

  设：

  $U=\{\ a,\ aa\ \}$

  $V=\{\ b,\ bb\ \}$

  则：

  $UV=\{ab,\ abb,\ aab,\ aabb\}$

2.3 程序语言的语法描述

- 闭包(Kleene Closure)和正则闭包(Positive Closure)

  V自身的n次连接积为

  $V^* = V^0 \cup V^1 \cup V^2 \cup V^3 \cup \ldots V^n$ 规定$V^0 = \{\varepsilon\}$

  $V^* = \{ \varepsilon , a, aa, aaa, aaaa, \ldots\}$

  $V^+ = \{ a, aa, aaa, aaaa, \ldots\}$

  V^* 是V的闭包, n=0 ~ ∞

  $V^+ = V\ V^*$是V的正则闭包, n=1 ~ ∞

27

2.3.1 上下文无关文法

2.3.1 上下文无关文法

- 文法(Grammar)
  描述语言结构的形式规则。

  文法贯穿整个课程，用于组织编译程序的前端。

  特点：这些规则必须是准确的，易于理解的，而且，应当有相当强的描述能力，足以描述各种不同的结构。

29

2.3.1 上下文无关文法

- 文法(Grammar)
  描述语言结构的形式规则。

  比如：if-then

  if (表达式 [expression, expr]) **then** 陈述 [statement, stmt]

  stmt → **if** (expr) **then** stmt

  if, then 终结符terminals

  expr, stmt 非终结符 non-terminals

30

2.3.1 上下文无关文法

- **终结符号**(Terminals)：
  - 是组成语言的基本符号，在程序语言中就是以前屡次提到的单词符号，如基本字、标识符、常数、算符和界符等。

  例如： if, then.

- **非终结符号**(Non-terminals)：
  - 用来代表语法范畴。如：语句、表达式等。

  例如： expr or stmt.

31

2.3.1 上下文无关文法

- 上下文无关文法(Context-Free Grammar)

  自然语言是上下文有关的。

  它所定义的语法范畴是完全独立于这种范畴可能出现的环境的。
- 一个上下文无关文法G包括四个组成部分：
  - 一组终结符号 terminals
  - 一组非终结符号 non-terminals
  - 一个开始符号 start symbol
  - 一组产生式 production

32

2.3.1 上下文无关文法

- 例如：一个简单的算术表达式文法：
  a) E→id
  b) E→E+E
  c) E→E*E
  d) E→(E)

  产生式固定模式A→α
  - **终结符号**：id(数字或特定符号)，+，*，(，)
  - **非终结符号**：E
  - **开始符号**：算术表达式E
  - **产生式**：a), b), c), d)都是产生式

33

2.3.1 上下文无关文法

- **开始符号**：
  - 是一个特殊的非终结符号，它代表所定义的语言中我们最终感兴趣的语法范畴，这个语法范畴通常称为“句子”或是“程序”。
- **产生式**：
  - 是定义语法范畴的一种书写规则。
  - 一个产生式的形式是A→α或A::=α
    - A：是非终结符号
    - α：是由终结符号或与非终结符号组成的一个符号串。

34

2.3.1 上下文无关文法

- 形式化定义：
  一个上下文无关文法是一个**四元式**（V_T,V_N,S,P）
  - V_T是一个非空有限集，它的每个元素称为终结符号；
  - V_N是一个非空有限集，它的每个元素称为非终结符号，$V_T \cap V_N = \phi$；
  - S是一个非终结符号，称为开始符号；$S \in V_N$。

35

2.3.1 上下文无关文法

- 形式化定义：
  一个上下文无关文法是一个**四元式**（V_T,V_N,S,P）
  - P是一个产生式集合（有限），每个产生式的形式是$P \rightarrow a$。

    其中，$P \in V_N$，$a \in (V_T \cup V_N)^*$。开始符号S至少必须在某个产生式的左部出现一次。

    $P \rightarrow a_1|a_2|\ldots|a_n$。其中，$a_i$称为是P的一个候选式。→读作“定义或产生”，直竖读为“或”,它是元语言符号。

36

2.3.1 上下文无关文法

- 形式化定义（**简化**）：
  一个上下文无关文法是一个四元式（V_T ,V_N ,S , P ）
  ① V_T 是 id(数字或特定符号 if, then, else…)，+，*，(，)…；

  ② V_N 是所有的非终结符的集合；

  ③ S 是一个开始符号。

  ④ P 是一个产生式集合，比如之前的a), b), c), d)。

2.3.1 上下文无关文法

- 上下文无关文法的**简化式**
  一组产生式，如果其开始符号都是同样的非终结符，则可以把这一组产生式组合在一起，每个产生式用符号“|”隔开，读作“或(or)”。

  a) E→id
  b) E→E+E
  c) E→E*E
  d) E→(E)

E → E | id | E+E | E*E | (E)

2.3.1 上下文无关文法

- 上下文无关文法
  如何使用上下文无关文法定义一个语言？
  a) E→id
  b) E→E+E
  c) E→E*E
  d) E→(E)

⇒ 每次仅使用一条规则，仅推导一步

随机：$E \Rightarrow (E)$ $\Rightarrow (E+E) \Rightarrow (id+E) \Rightarrow (id+id)$

2.3.1 上下文无关文法

- 上下文无关文法的**推导**

$$E \Rightarrow (E) \Rightarrow (E+E) \Rightarrow (\underbrace{id+}_{\alpha} \underbrace{E}_{A} \underbrace{)}_{\beta} \Rightarrow (\underbrace{id+}_{\alpha} \underbrace{id}_{\gamma} \underbrace{)}_{\beta}$$

$\alpha A \beta \Rightarrow \alpha \gamma \beta$

$A \rightarrow \gamma$ 是一个产生式，且α、$\beta \in (V_T \cup V_N)^*$

$a_1 \Rightarrow a_2 \Rightarrow a_3 \Rightarrow a_4 \ldots \Rightarrow a_n$ 称为从a_1到a_n的一个推导

2.3.1 上下文无关文法

- 上下文无关文法的**推导**

$a_1 \Rightarrow a_2 \Rightarrow a_3 \Rightarrow a_4 \ldots \Rightarrow a_n$ 称为从a_1到a_n的一个推导

如果从a_1到a_n的一个推导如果存在，则称a_1可推导出a_n

$a_1 \overset{+}{\Rightarrow} a_n$ 从a_1出发，经过一步或诺干步，可推导出a_n

$a_1 \overset{*}{\Rightarrow} a_n$ 从a_1出发，经过0步或诺干步，可推导出a_n

41

2.3.1 上下文无关文法

- **例题**

证明 (i*i+i)是文法

G(E)： $E \to i \mid E+E \mid E*E \mid (E)$

的一个句子。

证明：

$E \Rightarrow (E)$
$\Rightarrow (E+E)$
$\Rightarrow (E*E+E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$

$E \overset{*}{\Rightarrow} \alpha$, 则α是一个句型

$E \Rightarrow (i*i+i)$, **仅含终结符**，则$(i*i+i)$是**文法G**的一个句子。

文法G所产生的句子的**全体**是一个语言，记为$L(G)$。

42

2.3.1 上下文无关文法

- **例题**

文法$G_1(A)$：

$A \rightarrow c \mid Ab$

求$G_1(A)$的语言。

$L(G_1)$={cb，cbb，cbbb，…}
以若干c开头，后继若干个b
$L(G_1)=\{cb^n|n>=1\}$

$A \Rightarrow Ab \Rightarrow cb$

$A \Rightarrow Ab \Rightarrow Abb \Rightarrow cbb$

$A \Rightarrow Ab \Rightarrow Abb \Rightarrow Abbb \Rightarrow cbbb$

$A \Rightarrow Ab \Rightarrow Abb \Rightarrow Abbb \Rightarrow Ab\ldots b \Rightarrow cb\ldots b$

2.3.1 上下文无关文法

- **例题**

文法$G_2(S)$：

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$G_2(S)$的语言?

$L(G_2)$={ab，aabb，aaabbb，…}
以若干a开头，后继若干个b
$L(G_2)=\{a^mb^n|m, n>=1\}$

$S \Rightarrow AB \Rightarrow ab$

$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aabb$

$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aaAbbB \Rightarrow aaabbb$

$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aaAbbB \Rightarrow a\ldots Ab\ldots B \Rightarrow a\ldots ab\ldots b$

2.3.1 上下文无关文法

- **例题**

给出产生语言为$\{a^nb^n|n\geq1\}$的文法。

$G_3(S)$：

$S \rightarrow aSb$

$S \rightarrow ab$

2.3.1 上下文无关文法

- **例题**

$\{ ab^na \mid n\geq1\}$，构造其文法。

G1[Z]：

Z→aBa

B→b | bB

G2[Z]：

Z→aBa

B→b | Bb

等价文法： G和G′是两个不同的文法，若 L(G) = L(G′)

2.3.1 上下文无关文法

- 句型之间的推导并不是唯一

$E+E \overset{*}{\Rightarrow} i+i$

$E+E \Rightarrow \underline{E+i} \Rightarrow i+i$ **最右推导**：任何一步$\alpha \Rightarrow \beta$都是对α的最右非终结符进行替换

or

$E+E \Rightarrow \underline{i+E} \Rightarrow i+i$ **最左推导**：任何一步$\alpha \Rightarrow \beta$都是对α的最左非终结符进行替换

2.3.1 上下文无关文法

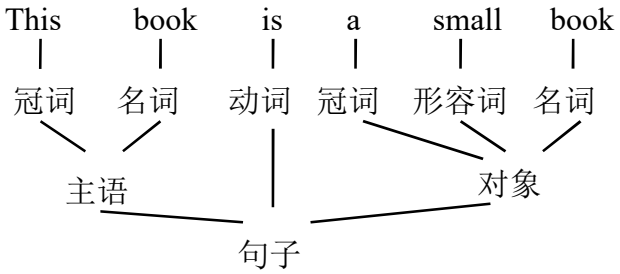
- 上下文无关文法的限制
  - 文法中不含任何下面形式的产生式 $P \rightarrow P$
  - 每个非终结符P必须都有用处。即：必须存在含P的句型，并且对于P不存在永不终结的回路。

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

2.3.2 语法分析树与二义性

2.3.2 语法分析树与二义性

- 自然语言的语法分析树



2.3.2 语法分析树与二义性

- 程序语言的语法分析树

证明 (i*i+i)是文法

G(E)： $E \rightarrow i \mid E+E \mid E*E \mid (E)$

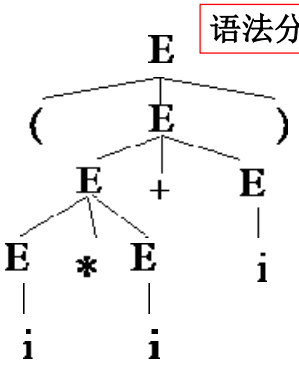
的一个句子。

如何最右推导？

语法分析树还一样吗？

证明：最左推导

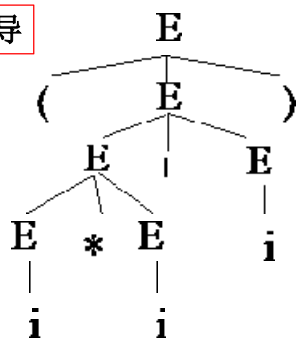
$E \Rightarrow (E)$
$\Rightarrow (E+E)$
$\Rightarrow (E*E+E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$



2.3.2 语法分析树与二义性

- 程序语言的语法分析树

$E \Rightarrow (E)$ 最左推导
$\Rightarrow (E+E)$
$\Rightarrow (E*E+E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$



最右推导？

$E \Rightarrow (E)$
$\Rightarrow (E+E)$
$\Rightarrow (E+i)$
$\Rightarrow (E*E+i)$
$\Rightarrow (E*i+i)$
$\Rightarrow (i*i+i)$

一棵语法树是不同推导过程的共性抽象。

2.3.2 语法分析树与二义性

- 程序语言的语法分析树

  建树方法

  从根节点开始，自上而下建立语法树

  从开始符号开始，自左向右建立推导序列。

一个句型是否只对应唯一一棵语法树？

2.3.2 语法分析树与二义性

- 程序语言的语法分析树

  证明 (i*i+i)是文法

  G(E)： $E \rightarrow i \mid E+E \mid E*E \mid (E)$

  的一个句子。

$E \Rightarrow (E)$
$\Rightarrow (E+E)$
$\Rightarrow (E*E+E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$

最左推导过程唯一？

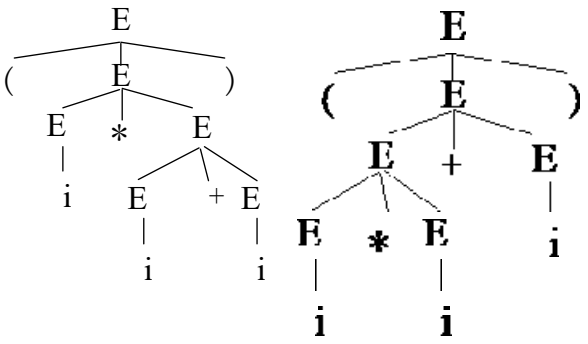
不是唯一推导

$E \Rightarrow (E)$
$\Rightarrow (E*E)$
$\Rightarrow (i*E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$

2.3.2 语法分析树与二义性

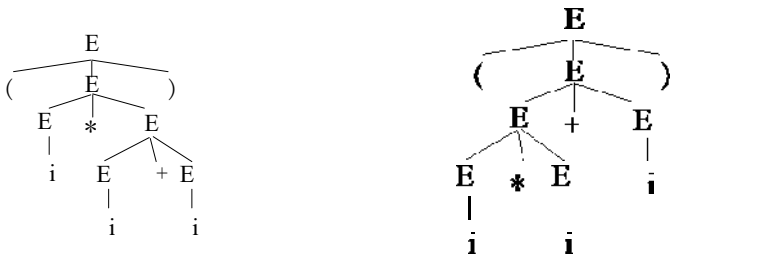
- 程序语言的语法分析树

$E \Rightarrow (E)$
$\Rightarrow (E*E)$
$\Rightarrow (i*E)$
$\Rightarrow (i*E+E)$
$\Rightarrow (i*i+E)$
$\Rightarrow (i*i+i)$



2.3.2 语法分析树与二义性

- 语法分析树的**二义性**
  如果一个文法存在某个句子对应两棵不同的语法树，则称这个文法是二义的。
  句子：(i*i+i)
  文法：G(E)： $E \rightarrow i|E+E|E*E|(E)$ 是二义文法。



2.3.2 语法分析树与二义性

- 语法分析树的**二义性**
  如果一个文法存在某个句子对应两棵不同的语法树，则称这个文法是二义的。

文法：G(E)： $E \rightarrow i|E+E|E*E|(E)$ 是二义文法。

句子：(i*i+i)

2.3.2 语法分析树与二义性

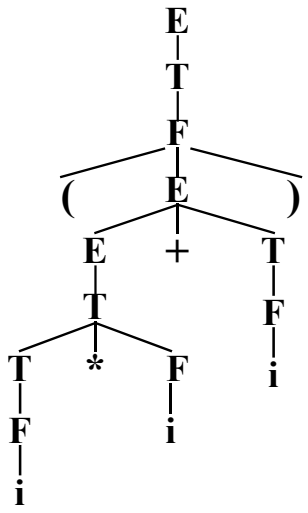
- 语法分析树的**二义性**
  句子：(i*i+i)

无二义文法G(E)：

$E \rightarrow T \mid E+T$
$T \rightarrow F \mid T*F$
$F \rightarrow (E) \mid i$

$E \Rightarrow T$
$\Rightarrow F$
$\Rightarrow (E)$
$\Rightarrow ?$

每个非终结符必须都有用处



2.3.3 形式语言鸟瞰

第二章 高级语言及其语法描述 2.3 程序语言的语法描述

2.3.3 形式语言鸟瞰

- 乔姆斯基（Chomsky）把文法分四类：
  - 0型、1型、2型和3型。
  - 其描述能力的强度有下列关系：0型强于1型、1型强于2型、2型强于3型。

2.3.3 形式语言鸟瞰

- 0型文法：
  - 设G=（V_T,V_N,S, P ）
    对每个产生式α→β有
    α∈（$V_N \cup V_T$）* 且至少含有一个非终结符
    β∈（$V_N \cup V_T$）*

    例如：可以是$A_0 \rightarrow A_0$, 也可以是$A_1 \rightarrow B$

61

2.3.3 形式语言鸟瞰

- 1型文法
  1. 每个产生式为α→β均满足|α|≤|β|; 仅S→ε例外，但S不得出现在任何产生式的右部。

     |α|是α的长度，|β|是β的长度

     例如，1型文法：$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$，如果要用β替换A，则只能在上下文为$\alpha_1$和$\alpha_2$时才可以。也称为上下文有关文法。

62

2.3.3 形式语言鸟瞰

- 2型文法

  G为任何产生式为A→β，$A \in V_N$，$\beta \in (V_N \cup V_T)^*$

  其左边必须有且仅有1个非终结符，当用β替换A时，不用考虑上下文。

  也叫做上下文无关文法。

2.3.3 形式语言鸟瞰

- 3型文法

  G的任何产生式为

  (1) A→α**B** 或A→α

  (2) A→**B**α 或A→α

  其中$\alpha \in V_T^*$，A、$B \in V_N$。

  (1)式为右线性文法；(2)式为左线性文法。

  也称线性文法，或称为正规文法

2.3.3 形式语言鸟瞰

- 文法说明：
  - 1型文法也称上下文有关文法。这种文法意味着，对非终结符进行替换时务必考虑上下文，并且，一般不允许替换成空串ε。
  - 2型文法也称上下文无关文法。
  - 3型文法也称线性文法，或称为正规文法。

65

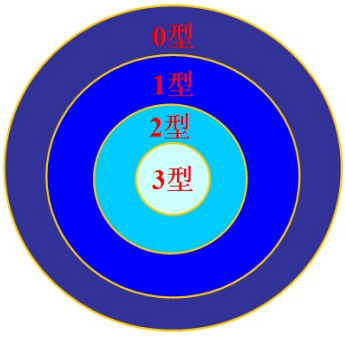
2.3.3 形式语言鸟瞰

- * G是0型文法，L(G)是0型语言；
  其能力相当于图灵机(TM)
- * G是1型文法，L(G)是1型语言;
  其识别系统是线性界限自动机(LBA)
- * G是2型文法，L(G)是2型语言;
  其识别系统是不确定的下推自动机(PDA)
- * G是右线性文法，L(G)是3型语言
  G是左线性文法，L(G)是3型语言
  其识别系统是有限自动机(FA)

66

2.3.3 形式语言鸟瞰

- 四种文法之间的关系是将产生式作进一步限制而定义的
  四种文法之间的逐级“包含”关系如下：



67

第二章 小结

第二章 小结

2.1 程序语言的定义
2.2 高级语言的一般特性
2.3 程序语言的语法描述

Coursework

2.1 设有文法G[A]：

A→bA | aA | bc

判断符号串abc，bbac，bbaA，bbAc是否能由G[A]生成。

2.2 令文法G_6为

N→D｜ND

D→0｜1｜2｜3｜4｜5｜6｜7｜8｜9

（1）G_6的语言$L(G_6)$是什么？

（2）给出句子0137、334和568的最左推导和最右推导。

Coursework

2.3 写一个文法，使其语言是偶正整数的集合。 要求：

（1）允许0 打头；

（2）不允许0 打头。

2.4 令文法为

E→T｜E+ T｜E-T

T→F｜T*F｜T/F

F→(E)｜i

（1）给出i+i*i、i*(i+i)的最左推导和最右推导；

（2）给出i+i+i、i+i*i和i-i-i的语法树。

Coursework

2.5 证明下面的文法G[S]是二义的：

S→ S (S) S | ε