普通高等教育“十一五”国家级规划教材

高等院校计算机专业及专业基础课系列教材

编译原理

孙家骕 编著

北京大学出版社

普通高等教育“十一五”国家级规划教材

北京高等教育精品教材
BEIJING GAODENG JIAOYU JINGPIN JIAOCAI

编译原理

孙家骕　编著

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书较全面地介绍了编译程序设计的基本原理和方法，详细地介绍了编译过程中的词法分析、语法分析、语义处理及中间代码生成、中间代码优化、目标代码生成及寄存器分配、运行时刻的存储分配等的原理和实现技术。本书采用属性文法的形式辅助描述程序语言的语义，用语法制导翻译的策略实现对程序语言的翻译，这样做使得语义描述更为直观、严谨，翻译过程表述更为清晰、易懂。本书适于用作高等学校计算机专业编译原理课的教材，也可以用作软件工程师的参考书。

图书在版编目(CIP)数据

编译原理/孙家骕编著. —北京：北京大学出版社，2008.1
ISBN 978-7-301-09803-5

Ⅰ.编… Ⅱ.孙… Ⅲ.编译程序—程序设计—高等学校—教材 Ⅳ.TP314

中国版本图书馆 CIP 数据核字(2005)第 119556 号

书　　名：编译原理
著作责任者：孙家骕　编著
责 任 编 辑：沈承凤
封 面 设 计：张　虹
标 准 书 号：ISBN 978-7-301-09803-5/TP·0822
出 版 发 行：北京大学出版社
地　　址：北京市海淀区成府路 205 号　100871
网　　址：http://www.pup.cn
电 子 信 箱：zpup@pup.pku.edu.cn
电　　话：邮购部 62752015　市场营销中心 62750672　编辑部 62752038　出版部 62754962
印 刷 者：涿州市星河印刷有限公司
经 销 者：新华书店
787 毫米×1092 毫米　16 开本　15.75 印张　390 千字
2008 年 1 月第 1 版　2008 年 1 月第 1 次印刷
定　　价：28.00 元

前　言

“编译原理”课是计算机软件专业重要基础课之一，它涉及的内容不仅有计算机语言编译的基本理论和方法，还有计算机语言定义的方法，特别是计算机语言语法的形式定义方法。这些知识不仅对设计计算机高级程序设计语言及其编译系统是十分必要的，而且对设计、实现计算机操作系统的命令语言、数据库管理系统中的查询语言、编辑系统以及自然语言的处理，都是必要的。为了进一步搞好编译原理的教学，在参考国内外优秀编译原理教材的基础上，我们编写了这本教材。

本书共有 10 章。

第 1 章介绍了高级程序设计语言语法的形式定义及相关概念，包括推导、归约、句型、句子、分析树等，还对形式语言的分类作了简单介绍。

第 2 章介绍了组成编译程序的各个部分的功能及相互关系，还介绍了构造编译程序的几个途径。

第 3 章介绍了词法分析的功能、词法分析程序的构造方法以及相关的概念。这些概念有三型文法、正则表达式和有限状态自动机等。三型文法和正则表达式可以用来描述程序设计语言中单词的结构；有限状态自动机可以用来辅助构造词法分析程序。本章还对如何通过正则表达式、有限状态自动机自动构造词法分析程序作了简单介绍，涉及到正则表达式到有限状态自动机的转换、有限状态自动机的确定化、化简等。

第 4 章介绍了常用的语法分析方法和语法分析程序的构造算法。包括预测分析法、算符优先分析法、LR(k)分析法以及 LL(1)分析表、递归下降子程序、算符优先关系表、SLR 分析表、LALR(1)分析表和 LR(1)分析表的构造算法。本章还介绍了为了适应语法分析要求而对文法进行等价改写的算法。

第 5 章介绍了属性文法，包括属性文法、综合属性、继承属性的定义和作用，并给出了若干描述程序语言构成部分的属性文法。本章还介绍了基于属性文法的翻译模式，给出了基于基础文法是 LL(1)文法的翻译模式的构造预测翻译程序的算法。

第 6 章介绍了语义检查，给出了类型表达式及实现类型检查的相关翻译模式。这一章还介绍了符号表的构成及作用。

第 7 章介绍了运行时刻的内存分配方法，包括静态分配、动态栈式分配和动态堆式分配以及在允许子程序嵌套的程序语言里如何实现对非局部变量的访问。本章还介绍了高级程序设计语言采用的几种参数传递机制。

第 8 章介绍了中间代码生成，给出了几种常用的中间代码形式，还给出了算数表达式、

布尔表达式、各种语句的中间代码以及产生这些中间代码的翻译模式。

第 9 章介绍了中间代码优化，主要包括几种数据流方程的求解算法、循环的查找、循环的优化以及删除公共子表达式等优化。

第 10 章介绍了目标代码生成，给出了一个简单的目标代码生成算法。还给出了图着色的寄存器分配算法、生成 DAG 及从 DAG 生成目标代码的算法。

由于作者水平有限，书中难免有不当之处，欢迎读者朋友批评指正。

北京大学信息科学技术学院

孙家骕

2007 年 10 月

目　录

第1章 预备知识

高级程序设计语言(以下简称为高级语言)是人与计算机进行交流的重要工具。我们用计算机解决问题时,一个典型的过程是:首先认真分析这个问题,在此基础上设计出解决该问题的算法;再根据算法,用高级语言编写出程序;最后把程序输入到计算机中,让计算机执行这个程序,以得到预期的结果。对绝大多数计算机而言,它们都不能直接执行高级语言编写的程序,需要有一个语言处理程序把高级语言程序转换成与其等价的机器语言程序(即由计算机指令构成的程序)。这个转换工作,通常由编译程序完成。要想由编译程序正确地完成这个转换工作,即把高级语言程序转换成与其等价的机器语言程序,必须对高级语言进行严格、准确的定义,包括语法和语义的定义。至今,我们已经有了定义高级语言语法的形式化方法,它是严格而准确的;而对于高级语言的语义定义,目前还没有一个被广泛接受的形式化方法,一般都是采用自然语言去定义高级语言的语义。本章将给出高级语言的语法定义。

1.1 相关定义

从形式上看,任何语言都是符号串的集合,集合中的符号串必须满足某些条件。下面给出符号、符号串的有关定义。

1.1.1 字母表

字母表是由符号构成的非空的、有穷的集合,这里的符号是对客观存在的一种抽象描述,它可以是字母、数字、汉字、音符,等等,我们对它不去形式地定义,正像我们不去形式定义几何当中的点、线、角一样。由此可知,{a},{a,b,1,2},{1,2,+,-},{人,口,手}等四个集合,都各自是一个字母表。

1.1.2 符号串

符号串是由符号组成的有穷序列,而符号属于某个字母表,所以,符号串总是和某个字母表相联系的。我们称由字母表中符号组成的有穷序列为该字母表上的符号串。例如,设字母表$\Sigma=\{a,b,c\}$,则符号序列 ε(空串,即一个符号也没有的序列),a,ab,bc,bac 等都是Σ上的符号串。在关于语言的理论里,也称符号串为字或句子。

1. *常用的符号串术语*

(1) 符号串长度

指的是符号串所含符号个数,若 α 为符号串,则其长度记为$|\alpha|$。例如:ε 的长度$|\varepsilon|=0$;设Σ上的符号串 $\alpha=abc$,$\beta=bc$,显然 α,β 分别含 3 个和 2 个符号,于是$|\alpha|=3$ 而$|\beta|=2$。

(2) 符号串前缀

由非空符号串 α 的第 1 个符号开始的连续的 $n(n\geqslant 0$ 且 $n\leqslant|\alpha|)$个符号构成的序列称为 α 的前缀，空符号串 ε 的唯一前缀为 ε 本身。

例如，符号串 abc 的前缀有 ε,a,ab 及 abc，共 4 个。显然若符号串 α 的长度为 n，则 α 的前缀共有 n+1 个。

(3) 符号串后缀

从一个符号串删去它的任意一个前缀后的剩余部分是该符号串的后缀。例如，ε 的唯一前缀是 ε。从 ε 删去 ε 后仍为 ε，所以 ε 的后缀为 ε，这是它唯一的后缀。符号串 abc 的前缀有 ε,a,ab,abc，依次从 abc 删除它的这些前缀后，可得到 abc,bc,c,ε。这四个符号串就是符号串 abc 的四个后缀。显然，长度为 n 的符号串 α 共有 n+1 个后缀。

(4) 符号串子串

由非空符号串 α 的第 $i(1\leqslant i\leqslant|\alpha|)$个符号开始的连续 $j(0\leqslant j\leqslant|\alpha|+1-i)$个符号构成的序列称为 α 的子串，空符号串 ε 的子串是 ε。例如，设 α=abcd，取 i=1,j=2 时，可得到 α 的子串 ab；取 i=3,j=1 时，可得到 α 的子串 c；若取 i=1,j=4，则可得到 α 的子串 abcd；当 i 值取 2，j 值取 0 时，α 的子串为 ε。

(5) 符号串的真前缀、真后缀、真子串

若 x 是非空符号串 S 的前缀(后缀、子串)并且 $x\neq S$，则 x 为 S 的真前缀(真后缀、真子串)。例如，非空符号串 abc 的真前缀共有 3 个，它们是 ε,a,ab；abc 的真后缀共有 3 个，它们是 ε,c,bc；abc 的真子串有 ε,a,b,c,ab,bc。显然，空符号串 ε 没有真前缀、真后缀及真子串。

2. 符号串之间的运算

(1) 连接运算

符号串 α 和 β 的连接结果是把 β 紧接在 α 之后得到的符号串，记为 $\alpha\beta$。例如，α=ab，β=123，则 $\alpha\beta$=ab123 而 $\beta\alpha$=123ab。一般，$\alpha\beta\neq\beta\alpha$，即连接运算没有交换律。不过，对于任何符号串 α，都有 $\alpha\varepsilon=\varepsilon\alpha=\alpha$。显然，连接运算具有结合律，即$(\alpha\beta)\gamma=\alpha(\beta\gamma)$，其中，$\alpha,\beta,\gamma$ 均为任意的符号串。

(2) 方幂运算

符号串方幂运算是同一个符号串的连接结果。符号串 α 的方幂记为 α^n，其中 $n\geqslant 0$，当 n=0 时，$\alpha^n=\alpha^0=\varepsilon$。

$n>0$ 时，$\alpha^n=\alpha\alpha\cdots\alpha$，即 n 个 α 连接运算的结果。例如，

$$\alpha=abc,\ \alpha^0=\varepsilon,\ \alpha^1=abc,\ \alpha^2=abcabc,\cdots,\ \alpha^n=\underbrace{abcabc\cdots abc}_{n个abc}$$

有了字母表、符号串的定义后，那么语言可定义为字母表上符号串构成的集合。例如，设字母表$\Sigma=\{a,b,0,1\}$，则 $L=\{a,b,a0,a1,b0,b1\}$是Σ上的一个语言。L 由 6 个符号串(也称为字或句子)构成。

下面给出语言之间的运算。设 L 和 M 是字母表Σ上的两个语言，它们之间可进行如下

的运算：

(1) 并运算

L 与 M 的并记为 L∪M，其定义为

$$L \cup M = \{x \mid x \in L \text{ 或者 } x \in M\}$$

(2) 连接运算

L 与 M 连接运算记为 LM，其定义为

$$LM = \{xy \mid x \in L \text{ 并且 } y \in M\}$$

LM 中的符号串，是 L 中某个符号串和 M 中某个符号连接的结果，由于 $\varepsilon\alpha=\alpha\varepsilon=\alpha$，所以有 $\{\varepsilon\}L=L\{\varepsilon\}=L$。

显然，语言的连接运算也不具备交换律，但是有结合律。一般 $LM \neq ML$，而 $(LM)N=L(MN)$，这里 L，M，N 均为任意的语言。

连接运算可以扩展成方幂运算。同一个语言 L 的 n 次连接就是 L 的方幂运算，记为 L^n，其定义为

$$n=0 \text{ 时，} L^n = L^0 = \{\varepsilon\}$$

$$n>0 \text{ 时，} L^n = \underbrace{LL\cdots L}_{n\text{个}}$$

(3) Kleen 闭包运算

语言 L 的 Kleen 闭包运算记为 L^*，其定义为

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \cdots$$

(4) 正闭包运算

语言 L 的正闭包运算记为 L^+，其定义是

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup \cdots$$

语言 L 的 Kleen 闭包及其正闭包有如下关系：

$$L^* = L^+ \cup L^0$$

以上对语言的运算定义，适合于任何字母表上的任何语言。

例 1.1 设字母表 $\Sigma_1=\{A,B,C,\cdots,X,Y,Z,a,b,c,\cdots,x,y,z\}$，$\Sigma_2=\{0,1,2,\cdots,9\}$，即，$\Sigma_1$ 由 26 个大写英文字母和 26 个小写英文字母构成，而 Σ_2 由 10 个十进制数字构成。$L=\{A,B,C,\cdots,X,Y,Z,a,b,c,\cdots,x,y,z\}$ 是字母表 Σ_1 上的语言；$D=\{0,1,2,\cdots,9\}$ 是 Σ_2 上的语言。L 与 D 中的符号串长度均为 1。对语言 L 和 D 进行各种运算，可以得到新的语言。

(1) L∪D 是由所有长度为 1 的英文大小写字母串和十进制数字串构成的集合。

(2) LD 是由所有长度为 2、且第 1 个符号是英文字母(大写或小写)、第 2 个符号是十进制数字的符号串构成的集合。

(3) L^3 是由所有长度为 3 的英文字母(大、小写)组成的符号串构成的集合。

(4) L^* 是由所有的英文字母(大、小写)组成的符号串(包括空符号串)构成的集合。

(5) D^+是由所有长度大于零的、十进制数字组成的符号串构成的集合。若设 $D_1=\{1,2,\cdots,9\}$,$D_2=\{0\}$,则 $D_1D^*\cup D_2$ 是由我们通常写的无符号十进整数构成的集合(每个大于零的整数第 1 位数字一定大于零)。

(6) $L(L\cup D)^*$是由所有字母(英文大、小写字母)打头的字母数字符号串构成的集合,也就是 PASCAL 语言的标识符集合。

1.2 高级语言的形式定义

高级语言的定义主要包括语法和语义两部分。目前,对高级语言的语义,尚无理想的形式化定义方法,这里仅对高级语言的语法给出形式化定义。按照形式语言文法及语言的分类,一般的高级语言都是上下文无关语言,即可以用上下文无关文法来描述高级语言的语法。本节介绍上下文无关文法及相应的语言——上下文无关语言,在 1.4 节中,还将进一步解释上下文无关文法。在谈到形式语言时,我们习惯用"文法"这个术语,而谈到高级语言或其他具体的语言(如自然语言)时,习惯用"语法"这个术语。这里"文法"和"语法"指的是同一事物。一个上下文无关文法由四部分组成:一组终结符号,一组非终结符号,一个开始符号以及一组产生式(也称生成式)。其中,终结符号是组成语言的符号(即语言是由这些终结符号串构成的集合),例如高级语言中的英文字母、十进制数字、运算符号、分隔符号等;非终结符号(也称语法变量)用来代表语言中的语法单位(或称语法中的结构),例如,程序语言中的函数、过程、分程序、语句、表达式等。

开始符号是一个特殊的非终结符号,它代表语言中的最大语法单位。例如,程序语言中的程序。

产生式(也称产生规则或简称规则)是一个有序对$\langle A,\alpha\rangle$,通常记为

$$A\to\alpha \quad (\text{或 } A::=\alpha)$$

其中,A 是一个非终结符号,α 是一个由终结符号和非终结符号组成的符号串(可以是空串 ε)。称 A 为产生式的左部,α 为产生式的右部。这里非终结符 A 代表了语法单位 α。

下面给出上下文无关文法的形式定义。

定义 1.1 一个上下文无关文法 G 是一个四元组(V_T,V_N,S,P)即 $G=(V_T,V_N,S,P)$,其中,V_T 是一个非空、有穷集合,其中的每个元素是一个终结符号。

V_N 是一个非空、有穷集合,其中的每个元素是一个非终符号。

S 为开始符号,它是一个非终结符号,即 $S\in V_N$。

P 是一个产生式构成的非空、有穷集合。其中产生式形式如 $A\to\alpha$(或写成 $A::=\alpha$),这里 $A\in V_N$,$\alpha\in(V_N\cup V_T)^*$。

为了简化书写,左部相同的产生式:

$$A\to\alpha_1$$
$$A\to\alpha_2$$
$$\vdots$$

$$A \to \alpha_n$$

可以合写成 $$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$

这里的“|”和“→”一样，是描述文法的元符号。有时也用 T（或别的符号）表示 V_T，用 V（或别的符号）表示 V_N。

定义 1.2 若文法 $G=(V_T,V_N,S,P)$，$A\to\beta$ 是 G 的一条产生式，则称 $\alpha A\gamma$ 直接推导出 $\alpha\beta\gamma$，记为

$$\alpha A\gamma \Rightarrow \alpha\beta\gamma, \text{其中 } \alpha,\gamma\in(V_T\cup V_N)^*$$

不难看出，符号串 $\alpha A\gamma$ 直接推导出 $\alpha\beta\gamma$ 也就是符号串 $\alpha A\gamma$ 中的 A 可以用 β 替换，其前提是 $A\to\beta$ 为 G 的一条产生式。若 $\alpha A\gamma$ 直接推导出 $\alpha\beta\gamma$，则称 $\alpha\beta\gamma$ 直接归约成 $\alpha A\gamma$，也称 $\alpha\beta\gamma$ 是 $\alpha A\gamma$ 的直接推导，$\alpha A\gamma$ 是 $\alpha\beta\gamma$ 的直接归约。直接推导可以看成是对一非终结符的替换。

如果存在一个直接推导序列：

$$\alpha_0 \Rightarrow \alpha_1$$
$$\alpha_1 \Rightarrow \alpha_2$$
$$\vdots$$
$$\alpha_{n-1} \Rightarrow \alpha_n \quad (n>0)$$

则称 α_0，n 步推导出 α_n。记为 $\alpha_0 \overset{+}{\Rightarrow} \alpha_n$ 或准确地记为 $\alpha_0 \overset{n}{\Rightarrow} \alpha_n$。

若 $\alpha_0=\alpha_n$ 或 α_0 可 $n(n>0)$步推导出 α_n，则称 α_0 可推导出 α_n，记为 $\alpha_0 \overset{*}{\Rightarrow} \alpha_n$，这就是说，$\alpha_0 \overset{*}{\Rightarrow} \alpha_n$意味着或者 $\alpha_0=\alpha_n$ 或者$\alpha_0 \overset{+}{\Rightarrow} \alpha_n$。若 α 可推导出 β，则称 β 可归约成 α。可见，推导和归约是互逆的。

定义 1.3 设文法 $G=(V_T,V_N,S,P)$。如果有 $S\overset{*}{\Rightarrow}\alpha$，则称 α 是文法 G 的一个句型。文法 G 的不含非终结符号的句型称为文法 G 的一个句子。文法 G 所有句子构成的集合称为文法 G 产生的语言，记为 L(G)。显然有

$$L(G)=\{w \mid S\overset{+}{\Rightarrow}w \quad \text{且} \quad w\in V_T^*\}$$

例 1.2 设 L 为表达式构成的语言，表达式中只有＋和＊运算，运算对象为 i（代表表示变量的标识符）。表达式中允许圆括号出现，以改变运算次序。产生该语言的文法如下：

$$G=(\{+,*,(,),i\},\{\langle\text{表达式}\rangle,\langle\text{项}\rangle,\langle\text{因子}\rangle\},\langle\text{表达式}\rangle,P)$$

P 中产生式如下：

$$\langle\text{表达式}\rangle \to \langle\text{表达式}\rangle + \langle\text{项}\rangle$$
$$\langle\text{表达式}\rangle \to \langle\text{项}\rangle$$
$$\langle\text{项}\rangle \to \langle\text{项}\rangle * \langle\text{因子}\rangle$$
$$\langle\text{项}\rangle \to \langle\text{因子}\rangle$$
$$\langle\text{因子}\rangle \to (\langle\text{表达式}\rangle)$$
$$\langle\text{因子}\rangle \to i$$

若把具有相同左部的产生式合并写在一起，上述产生式可写成如下形式：

$$\langle\text{表达式}\rangle \to \langle\text{表达式}\rangle + \langle\text{项}\rangle \mid \langle\text{项}\rangle$$

〈项〉→〈项〉*〈因子〉|〈因子〉

〈因子〉→(〈表达式〉)|i

上下文无关文法也可以用BNF范式(Backus normal form 也称 Backus_Naur form——巴克斯诺尔形式)表示。例如,上述文法用BNF范式表示,其产生式如下:

〈表达式〉::=〈表达式〉+〈项〉|〈项〉

〈项〉::=〈项〉*〈因子〉|〈因子〉

〈因子〉::=(〈表达式〉)|i

为了书写上的方便,我们常常用大写英文字母表示非终结符。如果依次用E,T,F表示〈表达式〉,〈项〉,〈因子〉三个非终结符,则上面的文法G可写成:

$$G=(\{+,*,(,),i\},\{E,T,F\},E,P)$$

P由下述产生式构成:

$$E\to E+T\mid T$$

$$T\to T*F\mid F$$

$$F\to (E)\mid i$$

$E\to E+T$ 是文法G的一条产生式,由定义1.2知,符号串E直接推导出E+T。即 $E\Rightarrow E+T$,由于 $E\to T$,$T\to F$,$F\to i$ 等都是G的产生式,我们还有

$$E+T\Rightarrow T+T$$

$$T+T\Rightarrow F+T$$

$$F+T\Rightarrow i+T$$

$$i+T\Rightarrow i+F$$

$$i+F\Rightarrow i+i$$

把上面推导连写在一起,得到下面推导:

$$E\Rightarrow E+T\Rightarrow T+T\Rightarrow F+T\Rightarrow i+T\Rightarrow i+F\Rightarrow i+i$$

可记为 $E\overset{+}{\Rightarrow}i+i$ 或准确地记为 $E\overset{6}{\Rightarrow}i+i$,即E六步推导出i+i。

按定义1.3,由于E是开始符号,并且有 $E\overset{*}{\Rightarrow}E$,$E\overset{*}{\Rightarrow}E+T$,$E\overset{*}{\Rightarrow}T+T$,…,$E\overset{*}{\Rightarrow}i+F$,$E\overset{*}{\Rightarrow}i+i$,所以E,E+T,T+T,…,i+F,i+i等都是文法G的句型。而i+i是不含非终结符的句型,所以它是文法G的一个句子。

E能推导出的所有终结符号串都是表达式(可包含+,*运算,运算对象为i,并且可以有圆括号);而所有这样的表达式,都能从E推导出来。所以L(G)是一个表达式构成的语言。例如

$$E\Rightarrow T\Rightarrow F\Rightarrow (E)\Rightarrow (E+T)\Rightarrow (T+T)\Rightarrow (F+T)\Rightarrow (i+T)$$

$$\Rightarrow (i+T*F)\Rightarrow (i+F*F)\Rightarrow (i+i*F)\Rightarrow (i+i*i)$$

定义1.4 若 $\alpha A\gamma\Rightarrow\alpha\beta\gamma$ 且 $\alpha\in V_T^*$,即A为符号串 $\alpha A\gamma$ 中的最左边的非终结符号,则称 $\alpha A\gamma$ 最左直接推导出 $\alpha\beta\gamma$,也称 $\alpha\beta\gamma$ 是 $\alpha A\gamma$ 的最左直接推导。记为 $\alpha A\gamma\underset{lm}{\Rightarrow}\alpha\beta\gamma$。

若 $\alpha A\gamma \Rightarrow \alpha\beta\gamma$ 且 $\gamma \in V_T^*$，即 A 为符号串 $\alpha A\gamma$ 中最右边的非终结符号，则称 $\alpha A\gamma$ 最右直接推导出 $\alpha\beta\gamma$，也称 $\alpha\beta\gamma$ 是 $\alpha A\gamma$ 的最右直接推导。记为 $\alpha A\gamma \underset{rm}{\Rightarrow} \alpha\beta\gamma$。

若推导序列

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$$

中的每一个直接推导都是最左直接推导，则称 α_0 最左推导出 α_n，记为 $\alpha_0 \overset{*}{\underset{lm}{\Rightarrow}} \alpha_n$，也称 α_n 是 α_0 的最左推导。

若推导序列

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$$

中的每个直接推导都是最右直接推导，则称 α_0 最右推导出 α_n，记为 $\alpha_0 \overset{*}{\underset{rm}{\Rightarrow}} \alpha_n$，也称 α_n 是 α_0 的最右推导。

最右推导也称作规范推导。

由于推导和归约是互逆的，若 α 最左推导出 β，则称 β 最右归约成 α；若 α 最右推导出 β，则称 β 最左归约成 α，或称 β 规范归约成 α。

对于例 1.2 中的文法，我们有如下推导序列：$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow i+T \Rightarrow i+F \Rightarrow i+i$，这是一个最左推导，E 最左推导出 i+i。

而 $E \Rightarrow E+T \Rightarrow E+F \Rightarrow E+i \Rightarrow T+i \Rightarrow F+i \Rightarrow i+i$ 是一个最右推导序列，E 最右推导出 i+i；而 $E \Rightarrow E+T \Rightarrow E+F \Rightarrow T+T \Rightarrow F+T \Rightarrow F+F \Rightarrow i+F \Rightarrow i+i$ 中既有最左推导，又有最右推导（对于某些文法的推导，可以既不是最左推导，也不是最右推导），在这个推导序列中，E 不是最左推导出 i+i，也不是最右推导出 i+i。

定义 1.5 令 G 是一个文法，S 是 G 的开始符号，$\alpha\beta\gamma$ 是 G 的一个句型，如果有 $S \overset{*}{\Rightarrow} \alpha A\gamma$ 及 $A \overset{+}{\Rightarrow} \beta$，则称 β 是句型 $\alpha\beta\gamma$ 的关于非终结符号 A 的短语。特别地，若有 $A \Rightarrow \beta$（即 A 直接推导出 β），则称 β 是句型 $\alpha\beta\gamma$ 关于非终结符号 A 的直接短语。一个句型的最左边的直接短语称为该句型的句柄。

由定义知，短语是属于某个句型的，而且是关于某个非终结符号的。谈论短语时，句型和非终结符号缺一不可。

仍以例 1.2 中的文法 G 为例，由于有

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow i+T \Rightarrow i+F$$

所以，i+F 是文法 G 的一个句型，并且有

$$E \overset{*}{\Rightarrow} T+F \text{ 以及 } T \overset{+}{\Rightarrow} i$$

所以，i 是句型 i+F 的关于非终结符号 T 的短语。由于 T 不是直接推导出 i（实际上是 $T \overset{2}{\Rightarrow} i$），i 不是 i+F 的关于 T 的直接短语。

在上面的推导序列中，我们还可以得出

$$E \overset{*}{\Rightarrow} F+T \text{ 及 } F \Rightarrow i$$