编译技术

组编／上海市高等教育自学考试办公室
主编／张守志

上海市高等教育自学考试指定教材

计算机软件专业（专升本）

高等教育出版社

上海市高等教育自学考试指定教材
计算机软件专业(独立本科段)

编 译 技 术

(附:编译技术自学考试大纲)

上海市高等教育自学考试办公室 **组编**

张守志 **主编**

高等教育出版社

内 容 简 介

本书系统地介绍了编译程序构造的一般原理和技术，包括词法分析、语法分析、中间代码生成、代码优化和目标代码生成等内容。本书的重点是介绍编译各阶段中所涉及到的技术与方法以及不同阶段的相互联系。全书内容丰富、概念清晰、逻辑性强、通俗易懂，既便于教学，也适宜于自学。

本书可作为计算机相关专业独立本科段自学考试教材，也可作为高等院校计算机类或信息类相关专业的本科教材或教学参考书，亦可供从事计算机软件开发的科技人员参考。

图书在版编目（CIP）数据

编译技术/张守志主编. —北京：高等教育出版社，2003.12
（上海市高等教育自学考试计算机软件专业（独立本科段）指定系列丛书）
ISBN 7-04-014239-2

Ⅰ.编… Ⅱ.张… Ⅲ.编译程序-程序设计-高等教育-自学考试-教材 Ⅳ.TP314

中国版本图书馆 CIP 数据核字（2003）第 126014 号

责任编辑 司马镭　　**特约编辑** 张　力
封面设计 吴　昊　　**责任印制** 蔡敏燕

书　　名 编译技术
主　　编 张守志

出版发行 高等教育出版社
社　　址 北京市西城区德外大街 4 号
邮政编码 100011
总　　机 010-82028899
传　　真 021-56965341

购书热线 010-64054588
021-56964871
免费咨询 800-810-0598
网　　址 http://www.hep.edu.cn
http://www.hep.com.cn
http://www.hepsh.com

排版校对 展望文化发展有限公司
印　　刷 江苏如皋市印刷有限公司

开　　本 787×1092　1/16
印　　张 16
字　　数 380 000

版　　次 2004 年 1 月第 1 版
印　　次 2004 年 1 月第 1 次
定　　价 22.30 元

凡购买高等教育出版社图书，如有缺页、倒页、脱页等质量问题，请在所购图书销售部门联系调换。

编者的话

随着计算机科学技术的迅猛发展，以及计算机应用的日益广泛，对计算机软件开发人员的需求也与日俱增。为了尽快和多渠道培养软件开发人员，开设自学考试软件专业（独立本科段）是势在必行之举。

编译技术是软件核心技术之一，是计算机软件开发人员必须掌握的知识和技能。

本书是根据上海市高等教育自学考试软件专业（独立本科段）的“编译技术”课程自学考试大纲编写的教材，所选内容符合大纲的要求。全书共9章：第1章介绍编译程序的概念、程序编译的过程及编译程序的结构；第2章介绍文法和语言的形式定义和文法的类型；第3章介绍词法分析程序的构造、正规式和有限自动机理论；第4章介绍各种语法分析方法，包括自上而下递归下降分析法、自下而上算符优先分析法和LR分析法；第5章介绍语法制导翻译和各种语法成分的中间代码的生成；第6章介绍符号表的组织和使用；第7章介绍运行时存储空间的组织与分配；第8章介绍中间代码的局部优化、循环优化和全局优化；第9章介绍目标代码产生过程。

本书在内容表述上，参考了目前国内外几本流行的教材，并结合“编译技术”课程的要求，更加强调编译程序构造的一般原理和技术，力求讲清编译各阶段中所涉及到的技术与方法，以及不同阶段的相互联系。书中每章后均附有一定数量的习题供读者练习和进一步思考。本教材配有《编译技术学习辅导》一书。

本书力求叙述通俗易懂、内容循序渐进、示例丰富，以便于自学。本书除可作为自学考试教材外，也可作为高等院校计算机类或信息类相关专业的本科教材或教学参考书，亦可供从事计算机软件开发的科技人员参考。

本书由张守志主编，参加本书编写的还有苏明柿、王欢和许彦。在本书编写过程中，得到上海市高等教育自学考试委员会、复旦大学继续教育学院有关领导的支持和帮助；对于全书内容的选取，复旦大学信息学院招兆铿教授提出了宝贵建议，在此一并致以诚挚的谢意。

由于编者水平有限、时间仓促，书中难免会有缺点和错误，恳请读者及同行们批评指正。

编　者

2003年10月

目　录

编 译 技 术

编译技术(6370)自学考试大纲

第1章 概　论

从理论上说，构造专用计算机来直接执行用某种高级语言编写的程序是可能的。但是，目前的机器能执行的是非常低级的语言，即机器语言。那么，高级语言最终是怎样在计算机上执行的呢？

术语编译代表从面向人的源语言表示的算法到面向硬件的目标语言表示的算法的一个等价变换。本章将简要介绍程序编译的过程及编译程序的结构。

1.1 编译程序的概念

当没有现成软件来解决所需要计算的问题时，用户可以利用高级语言编制程序来解决。现代计算机系统一般都含有不止一个高级语言的编译程序，对有些高级语言甚至配置了几个不同性能的编译程序，供用户按不同需要进行选择。高级语言编译程序是计算机系统软件的最重要组成部分之一，也是用户最直接关心的工具之一。

在计算机上执行一个高级语言程序一般要分为两步：第一步，用一个编译程序把高级语言程序翻译成机器语言程序；第二步，运行所得的机器语言程序求得计算结果。

通常所说的翻译程序是指这样的一个程序，它能够把某一种语言程序（称为源语言程序）改造成另一种语言程序（称为目标语言程序），而后者与前者在逻辑上是等价的。如果源语言是诸如 FORTRAN、Pascal、ALGOL 或 COBOL 这样的“高级语言”，而目标语言是诸如汇编语言或机器语言之类的“低级语言”，这样的一个翻译程序就称为**编译程序**。如果把编译程序看成一个“黑盒子”，它所执行的转换工作可以用图 1.1 来说明。

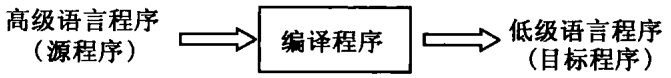

图 1.1　编译程序的功能

高级语言程序除了像上面所说的先编译后执行外，有时也可“解释”执行。一个源语言的解释程序是这样的程序，它以该语言写的源程序作为输入，但不产生目标程序，而是边解释边执行源程序本身。本书将不对解释程序作专门的讨论。实际上，许多编译程序的构造与实现技术同样适用于解释程序。

1.2 编译过程概述

编译程序的工作，是从输入源程序开始到输出目标程序为止的整个过程，整个过程非常

复杂。一个编译程序的整个工作过程是划分成阶段进行的，每个阶段将源程序的一种表示形式转换成另一种表示形式，各个阶段进行的操作在逻辑上是紧密连接在一起的。一般来说，整个过程可以划分成5个阶段：词法分析、语法分析、中间代码生成、代码优化和目标代码生成。另外还有两项重要的工作：表格管理和出错处理，这两项工作与上述5个阶段都有联系。编译过程中源程序的各种信息被保留在种种不同的表格里，编译过程中各阶段的工作都涉及到构造、查找或更新有关的表格，因此需要有表格管理的工作；如果编译过程中发现源程序有错误，编译程序应报告错误的性质和错误发生的地点，并且将错误所造成的影响限制在尽可能小的范围内，使得源程序的剩余部分能继续被编译下去，这些工作被称之为出错处理。

现在，从源程序在不同阶段所被转换成的表示形式的不同来介绍各个阶段的任务。

第一阶段：**词法分析**。词法分析的任务是从左到右一个字符一个字符地读入源程序，对构成源程序的字符流进行扫描和分解，从而识别出一个个单词(也称单词符号或符号)。这里所谓的单词是指逻辑上紧密相连的一组字符，这些字符具有具体含义。比如标识符是由字母字符开头，后跟字母、数字字符的字符序列组成的一种单词，此外还有算符、界符等等。例如某源程序片断如下：

```
begin var sum, first , count :real;  sum := first + count * 10 end.
```

词法分析阶段将构成这段程序的字符组成了如下单词序列：

(1) 保留字	begin	(2) 保留字	var
(3) 标识符	sum	(4) 逗号	,
(5) 标识符	first	(6) 逗号	,
(7) 标识符	count	(8) 冒号	:
(9) 保留字	real	(10) 分号	;
(11) 标识符	sum	(12) 赋值号	:=
(13) 标识符	first	(14) 加号	+
(15) 标识符	count	(16) 乘号	*
(17) 整数	10	(18) 保留字	end
(19) 界符	.		

可以看出，5个字符即“b”、“e”、“g”、“i”和“n”构成了一个分类为保留字的单词begin，两个字符即“：”和“=”构成了表示赋值运算的符号“：=”。这些单词间的空格在词法分析阶段都被滤调了。单词符号是语言的基本组成成分，是人们理解和编写程序的基本要素，识别和理解这些要素无疑是也是翻译的基础。如同将英文翻译成中文的情形一样，如果你对英语单词不理解或对构词法不熟悉，那就谈不上进行正确的翻译。在词法分析阶段工作中所依循的是语言的构词规则。

第二阶段：**语法分析**。语法分析的任务是在词法分析的基础上，根据语言的语法规则(文法规则)，把单词符号串分解成各类语法单位(语法范畴)，如“短语”、“句子”、“程序”等等。通过语法分解，确定整个输入串是否构成一个语法上正确的“程序”。语法分析遵循的是语言的语法规则。例如，符号串 X+0.618 * Y 代表一个“算术表达式”。因而，语法分析的任务就是识别这个符号串属于“算术表达式”这个范畴。

第三阶段：**中间代码生成**。在进行了上述的语法分析工作之后，有的编译程序按语言的语义将源程序变成一种内部表示形式，这种内部表示形式叫做中间语言或中间代码。所谓“中间代码”是一种结构简单、含义明确的记号系统。这种记号系统可以设计为多种多样的形式，重要的设计原则有两条：一是容易生成；二是容易将它翻译成目标代码。很多编译程序采用了一种近似“三地址指令”的“四元式”中间代码，这种四元式的形式为：

（运算符，运算对象 1，运算对象 2，结果）

例如，源程序 sum := first + count * 10 生成的四元式序列如图 1.2 所示，其中 t_i(i = 1,2,3) 是编译程序生成的临时名字，用于存放运算结果的。

(1) (inttoreal, 10, —, t_1)
(2) (* , count, t_1 , t_2)
(3) (+, first, t_2 , t_3)
(4) (:=, t_3 , —, sum)

图 1.2 中间代码

第四阶段：**代码优化**。在此阶段的任务是对前阶段产生的中间代码进行变换或进行改造，目的是使生成的目标代码更为高效，节省时间和空间。例如，图 1.2 的代码可变换为图 1.3 的代码，仅剩了两个四元式而执行同样的计算。也就是编译程序的这个阶段已经把将 10 转换成实型数的代码化简掉了，同时因为 t_3 仅仅用来将其值传递给 sum，也可以被化简掉，这只是优化工作的两个方面，此外诸如删除公共子表达式、强度削弱、循环优化等优化工作将在后面详细介绍。

(1) (* , count, 10.0, t_1)
(2) (+, first, t_1 , sum)

图 1.3 优化后的中间代码

第五阶段：**目标代码生成**。这一阶段的任务是把中间代码变换成特定机器上的绝对指令代码、可重定位的指令代码和汇编指令代码中的一种。这是编译的最后阶段，它的工作与硬件系统结构和指令含义有关，这个阶段的工作很复杂，涉及到硬件系统功能部件的运用、机器指令的选择、各种数据类型变量的存储空间分配以及寄存器和后缓寄存器的调度等。

例如，使用两个寄存器（R_1 和 R_2），可能生成如图 1.4 的某汇编代码。

(1) MOVF count , R_2
(2) MULF #10.0 , R_2
(3) MOVF first , R_1
(4) ADDF R_1 , R_2
(5) MOV R_1 , sum

图 1.4 目标代码

第一条指令将 count 的内容送至寄存器 R_2；第二条指令将其与实常数 10.0 相乘，这里用 # 表明 10.0 处理为常数；第三条指令将 first 的内容移至寄存器 R_1；第四条指令将寄存器 R_1 和 R_2 中的内容相加，结果存储在 R_1 中；第五条指令将寄存器 R_1 的值移到 sum 的地址

中。这些代码实现了对程序片断：begin var sum，first，count：real；sum：=first+count * 10 end. 的赋值。

前面提到过上述编译过程的阶段划分是一种典型的处理模式，事实上并非所有的编译程序都分成这样几个阶段。有些编译程序对优化没有什么要求，优化阶段就可省去；在某些情况下，为了加快编译速度，中间代码产生阶段也可以省去；有些最简单的编译程序在语法分析的同时产生目标指令代码。但是，多数实用的编译程序都采用上述几个阶段的工作过程。

1.3 编译程序的结构

上述编译过程的5个阶段是编译程序工作时的动态特征。编译程序的结构可以按照这5个阶段的任务分模块进行设计。编译程序的结构设计可从图1.5所示的总框开始，这个总框是一个数据转换图，由它能够比较容易地设计出程序结构的框图。

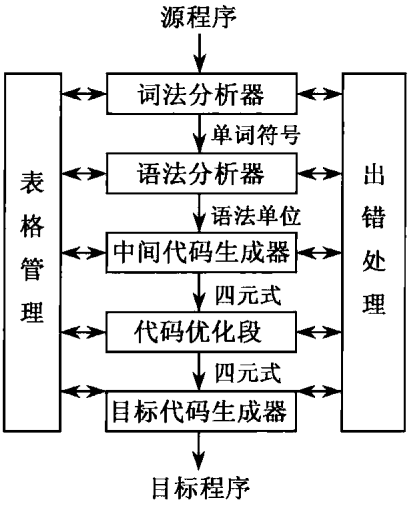

图1.5 编译程序总框

图1.5中的词法分析器、语法分析器、中间代码生成器、代码优化段和目标代码生成器将分别完成上述5个阶段的编译任务。每阶段的输出为下一阶段的输入，第一阶段的输入是源程序，最后阶段的输出是目标程序。每阶段的工作都与表格管理和出错处理这两部分功能模块相关。

词法分析器，又称**扫描器**，输入源程序，进行词法分析，输出单词符号。

语法分析器，简称为**分析器**，对单词符号串进行语法分析（根据语法规则进行推导或归约），输出由语法单位构成的语法树，判断输入串是否构成语法上正确的“程序”。

中间代码生成器，按照语义规则把语法分析器归约出（或推导出）的语法单位翻译成一定形式的中间代码，譬如说四元式。

有的编译程序在识别出各类语法单位后，构造并输出一棵表示语法结构的语法树，然后，根据语法树进行语义分析和中间代码产生；还有许多编译程序在识别出语法单位后并不真正构造语法树，而是调用相应的语义子程序。在后一种编译程序中，扫描器、分析器和中间代码生成器三者并非是截然分开的，而是相互穿插的，这样可以大大提高编译程序自身的工作效率。

代码优化段，对中间代码进行优化处理。

目标代码生成器，把中间代码翻译成目标程序。

编译过程中源程序的各种信息被保留在种种不同的表格里，编译各阶段的工作都涉及到构造、查找或更新有关的表格。因此，在编译程序中需要有一组管理各种表格的程序。

如果源程序有错误，编译程序应设法发现错误，把有关出错信息报告给用户。这部分工

作是由专门的一组程序(叫做**出错处理程序**)完成的。一个好的编译程序应能最大限度地发现源程序中的各种错误,准确地指出错误的性质和发生错误的地点,并且能将错误所造成的影响限制在尽可能小的范围内,使得源程序的其余部分能继续被编译下去,以便进一步发现其他可能的错误。出错处理程序与编译各阶段都有联系,与前三阶段的联系尤为密切。

1.4 学习构造编译程序

要在某一台机器上为某种语言构造一个编译程序,必须掌握下述三方面的内容:

(1) 源语言,对被编译的源语言(如 FORTRAN 或 Pascal),要深刻理解其结构(语法)和含义(语义)。

(2) 目标语言,假定目标语言是机器指令,那么,就必须搞清楚硬件的系统结构和操作系统的功能。

(3) 编译方法,把一种语言程序翻译成另一种语言程序的方法很多,但必须准确地掌握其中的一种到两种。

本课程是讲编译方法的,并且主要是讨论 FORTRAN 和 Pascal 这类强制式语言的翻译技术,需要读者对这些语言有一定的基本知识。

在本门课中,并不假定以某一特定机器作为目标机器。当需要涉及目标机器指令时,将采用一些通用的假想指令。因此,在学习这门课之前,读者必须具有计算机基础程序设计的知识。

由于编译程序是一个极其复杂的系统,在本书中将将它分解开来,一部分一部分地进行研究。因此,在学习过程中应注意前后联系,切忌用静止的、孤立的观点看待问题。作为一门技术课程,学习时务必注意理论联系实际,多做练习、多多实践。

本书中所引用的具体算法有些是用文字描述的,有些是用类 Pascal 语言表示的。所有这些算法都是原理性和解释性的,而且大多是不完备的(忽略某些次要因素或尚未学到的成分)。因此,并不意味着这些算法可以直接照抄使用。

在着手构造一个编译程序时,需要预先考虑种种具体因素,诸如系统功能要求(这种要求常常是多方面的)、硬件设备、软件工具等等,特别是必须估量所有这些因素对编译程序构造的影响。虽然这些都是工程实现时应予考虑的细节,但因篇幅所限,不可能涉及太多。

后文中将按照 1.2 节中所说的编译过程的各基本阶段,逐步介绍编译程序的构造方法和技术。

习 题

1.1 计算机执行用高级语言编写的程序有哪些途径,它们之间的主要区别是什么?

1.2 有人认为编译程序的 5 个组成部分缺一不可,这种看法正确吗?

第2章 文法和语言

一个程序设计语言是一个记号系统，如同自然语言一样，它的完整定义应包括语法和语义两方面。一个语言的语法是指一组规则，用它可以形成和产生一个合适的程序。文法是描述语言的语法结构的形式规则（即语法规则），这些规则必须是准确的、易于理解的，而且有相当强的描述能力，足以描述各种不同的语法结构。由这种规则所形成的程序语言应有利于句子的分析和翻译，而且，最好能通过这些规则自动产生有效的语法分析程序。目前广泛使用的是上下文无关文法，即用上下文无关文法作为程序设计语言语法的描述工具。

本章将对文法和语言给出形式定义，其基础是形式语言理论中有关的概念，重点讨论上下文无关文法及其句型分析中的有关问题。

2.1 符号和符号串

语言是在某个特定字母表上的符号串所组成的集合，为了考察如何按照语法规则来构造这个字母表上的符号串集合，首先讨论符号和符号串的有关概念。

定义 2.1 字母表是元素的有穷非空集合，字母表中的元素称为符号，因此字母表也称为符号集。

字母表包含了语言中所允许出现的一切符号，当然其中至少要包含一个符号。显然，对于不同的语言可以有不同的字母表，例如，二进制数语言的字母表是{0，1}，而对于 Pascal 语言，字母表可以说是由一切可打印字符组成的集合。通常用大写英文字母 A，B，…，以及希腊字母 Σ 等表示字母表，例如，字母表 A = {0，1} 与字母表 Σ = {a，b，c，d} 等等。

定义 2.2 符号串（字）是由字母表中的符号所组成的有穷序列。

例如，a，b，c，ab，ac，aaa 与 bbabc 等都是字母表{a，b，c}上的符号串。又如字母表{0，1}上的符号串可以是 0，1，01，10，100，…，即一切二进制数。

（1）符号串总是建立在某个特定字母表上的，且只由字母表上的有穷多个符号组成。需要注意的是，符号串中符号的出现顺序是很重要的，例如，ab 与 ba 及 010 与 100 都是不同的符号串。通常用小写英文字母 a，b，…，来表示符号串。

（2）允许有不含任何符号的符号串，这种符号串称为空符号串，简称空串（空字），用 ε 表示。

（3）符号串 x 中所包含符号的个数称为符号串 x 的长度，用|x|表示。例如，符号串 abc 的长度是|abc|=3，而符号串 010110 的长度是|010110|=6。显然|ε|=0。

（4）符号串的头、尾、固有头和固有尾。如果 z 是一符号串且 z=xy，那么 x 是 z 的头，y

是 z 的尾，如果 x 是非空的，那么 y 是固有尾；同样如果 y 是非空的，那么 x 是固有头。设字符串 z=abc，那么 z 的头是 ε，a，ab，abc，除 abc 外，其他都是固有头；z 的尾是 ε，c，bc，abc，除 abc 外，其他都是固有尾。

当只对符号 z=xy 的头感兴趣而对其余部分不感兴趣时，可以采用省略写法：z=x…；如果只是为了强调 x 在符号串 z 中的某处出现，则可表示为 z=…x…；符号 t 是符号串 z 的第一个符号，则表示为 z=t…。

(5) 符号串的连接。设 x 和 y 是符号串，它们的连接 xy 是把 y 的符号写在 x 的符号之后得到的符号串，例如设 x = ST，y = abu，则它们的连接 xy = STabu，看出 | x | = 2，| y | = 3，| xy | = 5。由于 ε 的含义，显然有 εx = xε = x。

(6) 符号串的方幂。设 x 是符号串，把 x 自身连接 n(n∈ℕ)次得到符号串 z，即 z=xx…xx，称为符号串 x 的方幂，写作 $z = x^n$，也即符号串 x 相继地重复写 n 次。例如，设 x = AB，那么 $x^0 = \varepsilon$，$x^1 = AB$，$x^2 = ABAB$，$x^3 = ABABAB$。对于 n > 0，有 $x^n = xx^{n-1} = x^{n-1}x$。

定义 2.3 若集合 A 中的一切元素都是某字母表上的符号串，则称 A 为该字母表上的符号串集合。

(1) 符号串集合的乘积。设两个符号串集合 A 和 B，它们的乘积定义为：

$$AB = \{xy \mid x \in A \text{ 且 } y \in B\}$$

即 AB 是满足 x 属于 A，y 属于 B 的所有符号串 xy 所组成的集合。例如，A = {a, b}，B = {c, d}，则集合 AB = {ac, ad, bc, bd}。因为对任意符号串 x 有 εx = xε = x，所以有 {ε}A = A{ε} = A。

(2) 闭包。指定字母表 Σ 之后，可用 Σ^* 表示 Σ 上的所有有穷长的串的集合。例如，令 Σ = {0, 1}，则 $\Sigma^* = \{\varepsilon, 0, 1, 01, 10, \cdots\}$，也可表示为字母表的方幂形式：

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cdots \cup \Sigma^n \cdots$$

Σ^* 称为集合 Σ 的闭包。而 $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cdots \cup \Sigma^n \cdots$ 称为 Σ 的正闭包。显然有：

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

$$\Sigma^+ = \Sigma\Sigma^* = \Sigma^*\Sigma$$

Σ^* 具有可数的无穷数量的元素。使用一般集合论中的表示符号：若 x 是 Σ^* 中的元素，则表示为 $x \in \Sigma^*$，否则 $x \notin \Sigma^*$。对于所有的 Σ，有 $\varepsilon \in \Sigma^*$。

2.2 文法和语言的形式定义

2.2.1 文法的直观概念

在给出文法和语言的形式定义之前，首先直观地认识一下文法的概念。

在表述一种语言时，无非是说明这种语言的句子，如果语言只含有有穷多个句子，则只需列出句子的有穷集就行了，但对于含有无穷句子的语言来讲，存在着如何给出它的有穷表

示的问题。

以自然语言为例，人们无法列出全部句子，但是人们可以给出一些规则，用这些规则来说明（或者定义）句子的组成结构，比如："我是大学生"是汉语的一个句子。汉语句子可以是由主语后随谓语而成，构成谓语的是动词和直接宾语，我们采用巴科斯范式（EBNF）来表示这种句子的构成规则：

<句子>::=<主语> <谓语>

<主语>::=<代词>|<名词>

<代词>::=我|你|他

<名词>::=王明|大学生|工人|英语

<谓语>::=<动词> <直接宾语>

<动词>::=是|学习

<直接宾语>::=<代词>|<名词>

其中，符号"::="表示该符号的左部由右部定义，可读作"定义为"；符号"|"表示"或"，意义为左部可由多个右部定义。

"我是大学生"的构成符合上述规则，而"我大学生是"不符合上述规则，因此它不是句子。这些规则成为判别句子结构合法与否的依据。换句话说，将这些规则看成是一种元语言，用它描述汉语，这里仅仅涉及汉语句子的结构描述，这样的语言描述称为**文法**。

一旦有了一组规则以后，我们可以按照如下方式用它们去推导或产生句子：

首先去找"::="左端带有<句子>的规则并把它表示成"::="右端的符号串，将该动作表示成：

<句子>⇒<主语> <谓语>

然后在得到的串<主语> <谓语>中，选取<主语>或<谓语>，再用相应规则中"::="的右端代替之。比如，选取了<主语>，并采用规则<主语>::=<代词>，那么得到：

<主语> <谓语>⇒<代词> <谓语>

如此重复做下去，可以得到句子"我是大学生"的全部动作过程是：

<句子>⇒<主语> <谓语>

⇒<代词> <谓语>

⇒我<谓语>

⇒我<动词> <直接宾语>

⇒我是<直接宾语>

⇒我是<名词>

⇒我是大学生