形式语义学基础与形式说明

屈延文 编著

科学出版社

形式语义学基础与形式说明

屈延文 编著

科学出版社

1989

内 容 简 介

本书是国家教委计算机软件专业教材编委会推荐教材之一. 本书较详细地给出了形式语义学的基础理论框架，但它并不是一本纯理论的教材，而是一本理论与软件实践相结合的教材.

全书共分十章. 介绍了指称语义学、代数语义学、操作语义学与公理语义学的基本内容及其应用，并介绍了并发程序设计语言各流派的语义模型和新一代计算机计算模型的理论问题. 例如 Curry 的组合逻辑，Martin-Löf 的直觉主义数学的讨论都是现代计算机理论较重要的基础内容.

本书内容丰富，重点突出，并配有大量习题. 可作为大学计算机系本科高年级学生、研究生软件专业的教材，也可供计算机软件设计、工程人员参考.

形式语义学基础与形式说明

屈延文 编著

责任编辑 刘晓融

科学出版社出版

北京东黄城根北街 16 号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1989 年 12 月第 一 版 开本：850×1168 1/32

1989 年 12 月第一次印刷 印张：21 5/8

印数：0001—690 字数：572,000

ISBN 7-03-000606-2/TP·40

定价：28.70 元

前　言

本书是计算机软件专业人员，大学软件专业本科高年级学生及研究生了解计算机科学理论在软件工程中应用的专门书籍．作者是站在软件的立场上来讨论计算机科学(尤其是形式语义学)的理论及其应用的．虽然，我们比较详细地介绍了形式语义学的基本理论框架，但它不是一本纯理论的书籍，而是一本理论联系实际的书籍．因此，在叙述中不苛求理论上的严密性．

早在 1980 年，作者在做 Ada 语言的准备工作时，就已开始对形式语义学的理论与实践进行研究．在研制 Ada 语言的编译程序时，我想到的第一个问题是，一本很厚的 Ada 语言文本，放在每一个程序设计人员面前，你怎么知道他们/她们看懂了 Ada 文本呢？怎么能相信每一个人能正确地归纳 Ada 文本的语义呢？如何知道编译程序的设计基础是可靠的呢？从这一点来说，没有 Ada 语言的形式定义是不行的，这一点可以从已有的文献中得到支持．

在研制 Ada 编译程序时，我们制定了计算机辅助管理、计算机辅助设计、严格质量管理及严格验收的设计原则．在当时，我们还认为编译程序的语义分析采用属性文法说明及属性文法的语义计算器是比较理想的方案．为了提高程序质量，提出了比较高的程序抽象要求，要求基于抽象数据类型的 class (package) 程序设计方法，要求程序结构有最好的层次分解性与模块分解性．所有这些，都促使我去写一本形式语义学的讲义，以提高 Ada 编译程序设计人员的素质．在当时，我们提出，指称语义学、属性文法及抽象数据类型应作为研制 Ada 编译程序的最基本理论准备．

后来，由于开展了对新一代计算机的研究，我们将形式语义学的研究与新一代计算机计算模型的研究结合起来了．在这方面最

为重要的理论研究是对 Curry 的组合逻辑与 Martin-Löf 的直觉主义数学的讨论.

作者在不断完善本书(原是一本油印讲义)的过程中，曾受到过多方面的支持与帮助. 原北京大学二分校（现在改名为信息工程学院)校长杨天行同志，最早支持我在该校讲授此课. 后来,在北京大学计算机科学系主任杨芙清教授的支持下，在北京大学两次为研究生班与学位研究生讲授此课. 同时,还在北京航空学院、华北计算技术研究所讲授此课.为使本书达到比较完善的程度,在编写过程中，曾得到中国科学院软件研究所唐稚松教授和周巢尘教授多方面的帮助，他们曾向我提供了许多珍贵的材料. 还应当提到的是已故的北京大学吴允曾教授，他也曾给过我多方面的帮助. 在本书出版之际,特向他们表示衷心的感谢.

屈 延 文

1987 年 7 月

目　录

• ▼ •

第一章 引 论

在这一章，我们将介绍形式语义学的基本概念，形式语义学的流派及分类.

1.1. 形式语义学

一位中国人学习英语，他对英语语义的理解是通过英语的相应汉语的解释来进行的.就是对汉语，他对未知语言成分意义的理解也是通过已知语义的语言成分的注释而达到的. 显然，在对自然语言理解的过程中，在说明对象语言的语义时，需要一个已知语义的语言，并建立这两种语言的对应关系，从而得到对于对象语言语义的理解. 换句话说，已知语义的语言称之为说明语言，当然说明语言有自己的语法及语义. 令说明语言的语句是 s，其语义记为$[\![s]\!]$，而对象语言的一个语句 r，语句 r 的语义记为 $[\![r]\!]$. $[\![r]\!]$ 与 $[\![s]\!]$ 的关系就是对象语言的注释. 自然语言之间的映射关系是通过语法书及对照字典来实现的，虽然其中存在着很大的不确定性.

什么叫做形式语义学呢?

形式语义学是对形式语言及其程序采用形式系统方法进行语义定义的学问. 在定义形式语言时，流行采用 BNF 定义语言的文法. 如何解释用 BNF 定义的语言的语义？这个问题就是形式语言研究的对象之一. 一个逻辑系统，例如一阶谓词演算，就可以看成为一个形式语言系统，那么这个逻辑语言的语义研究，也就是形式语义研究的对象. 形式语言不仅是串文法，还有图文法（graph grammar）及树文法等等，这些语言的语义研究，也是形式语义研究的对象. 一个程序也有语法与语义两个方面，语法在程序中主要表现为程序结构，而语义在于对语法特性适用的域的解释.

由于研究形式语言的语义方法不同，从而形式语义学大致可以分成如下的几个分支：

●指称语义学 (Denotational Semantics).

●代数语义学 (Algebraic Semantics).

●操作语义学 (Operational Semantics).

●公理语义方法 (Axiomatic Semantic Approach).

我们将在下面分别介绍这些语义学的基本点.

一个程序设计语言有两个最重要的概念：论域及辖域. 在本书的讨论中将特别强调这两点.

当代的程序设计语言，除纯表达式语言之外，一般都包括三个语法范畴：

●声明范畴.

●命令范畴.

●表达式范畴.

声明范畴的成分在于建立及改变环境；命令范畴的成分(即语句世界的成分)在于执行并改变状态；而表达式范畴的成分在于计算产生值. 在语言的声明范畴中，又主要包括作用域及可见性两个概念. 在命令范畴中的原子成分是赋值语句，GOTO 语句的存在主要用于改变程序的执行顺序，也就是说，具有 GOTO 语句的语言程序，其书写顺序与执行顺序是不同的. 在表达式范畴中，函数定义(或称函数抽象)与函数施用是两个基本概念. 如果一个表达式不仅计算产生值，而且还改变状态，我们称这个表达式有副作用 (side-effect).

在 Von neumann 计算机体系中，程序设计语言还引入了地址概念，致使表达式计算产生的值与地址联系起来. 如果使用这个值，则必须引用置有该值的内存的地址，我们称这样的程序是依赖于环境的，因为计算的结果与所在存贮器的位置有关.

如果一个语言没有命令范畴，它只有表达式范畴及表示类型与对象的声明范畴；而且这种语言中的每一个函数都不依赖于环境而存在；它所计算的结果与所在存贮器的位置无关，只要满足

了函数定义域则可以处处施用，绝无副作用，那么称这样的语言为施用型语言（Applicative Lauguage）.

为什么要研究形式语义学呢?

理论研究的最强大的动力是发展生产，促进社会的进步．软件生产长期停留在手工劳动的状态，生产力很低．像所有其它工业发展那样，软件产业也要追求自动化生产，大规模生成，高效率及高质量生产的目标．

为了发展软件生产力及保证软件生产的质量，软件工程方法被广泛地研究并得到普遍的应用．软件工程方法一般化分为如下几个阶段：

•软件要求．

•软件说明．

•程序设计．

•测试．

•维护．

保证软件正确性，能正确地写出软件说明（包括功能说明、系统说明、结构说明、测试说明等等）是十分关键的．正确的软件说明是程序设计的正确性的重要保证．怎样才能正确地写出说明呢？形式语义学方法是其中的最重要的方法之一．

软件生产的发展并不会满足于写出正确的程序说明．为了提高软件生产力，最有效率的措施是

•提高软件重复使用能力．

•软件自动生成技术．

发展软件自动生成技术，必须向计算机说明“做什么”（What to do）的语义及“如何做”（How to do）的语义．为了使计算机能够理解程序员说明的内容，形式语义说明方法是一个必由之路．提高软件重复使用能力，采用建立在类别代数理论基础之上的抽象数据类型形式方法（代数方法），是一个基本途径．形式语义学的重要性是不言而喻的．

正是由于对软件工程的研究，才使人们认识到程序设计必须

从 Von neumann 计算机中解放出来，从而提出研制新一代机的计划．也正是由于研制软件说明语言及软件自动生成，发现一阶谓词演算完全构造程序的困难性，使人们又重新认识数学并促进了构造数学的发展．而非 Von neumann 计算机的研制及构造数学的发展，既大大丰富了计算机科学的研究内容，又促进了计算机与软件的发展．

研究形式语义学，将会大大提高我们对程序设计语言本质的理解．从某种意义上来说，它会使我们对软件的认识有一个本质性的飞跃．

为什么会出现这样四个语义学的研究领域？这主要是有如下的分成四派的程序计算模型，它们是

●图灵机模型．

●谓词演算模型．

●递归函数论模型．

●代数模型．

这些模型与计算机模型、程序设计语言有图 1.1.1 所示的关系．

	1	2	3	4
计算模型	图灵机模型	谓词演算模型	递归函数论模型	代数模型
硬件系统	Von neumann 机器		组合逻辑计算机	
程序设计语言	状态语言 PASCAL, Ada 等	逻辑语言 (如 PROLOG)	函数施用型语言 (如 LISP)	类型程序设计语言

图 1.1.1

1.2. 指称语义学

要想了解指称语义学，必须了解什么是指称．例如下面的一

个语法符号串

$$fxy$$

其中 f，x，y 都是语法符号，是我们需要了解语义的语言的句子单位．显然仅从这个句子中，我们是无法知道它的语义的．如果我们有如下的域(数学对象的域)，一个域是函数域，我们记为 [Nat. Nat → Nat]；在这个函数域中，也有它的值或对象，例如 plus, mult, minus 等等．另一个域是一个自然数的值域，即 Nat = {0, 1, 2, ···,}．如果符号 f 的语义被记为 $[\![f]\!]$，如令 $[\![f]\!]$ = plus: Nat, Nat → Nat，我们就说 plus 是 f 的指称．同样的道理，我们也可以注释语法符号 x, y 的自然数指称．

因此，指称语义是采用形式系统方法，用相应的数学对象(例如 set, function 等等) 对一个即定形式语言的语义进行注释的学问．指称语义还可以被解释为：存在着两个域，一个是语法域，在语法域中定义了一个形式语言系统；另外一个是数学的域(或称之为已知语义的形式系统)．用一个语义解释函数，以语义域中的对象(值)来注释语法域中定义的语言对象的语义，即为指称语义．由于指称语义的理论支持是论域方程，在函数空间解这些论域方程需要不动点理论，于是也有人说："指称语义就是不动点语义"．

指称语义的研究首先是由 D. Scott 及 C. Strachey 等人开始的．

指称语义学的基础包括论域理论和 λ-演算，这些我们将在第二章中介绍．论域理论是在偏序集合上展开的，我们还将讨论完全偏序，函数单调性，函数连续性，泛函不动点等理论．将偏序关系限制在 Egli-milner 序及 Smyth 序上进行讨论，从而引出了研究不确定性的幂域 (Powerdomains) 理论．在本书的第三章，我们讨论程序设计语言的指称语义．在第四章，我们给出几个较大的程序设计语言的指称语义的例子．我们希望读者学完这些内容之后，能够学会读写指称语义及指称语义的证明技术．

1.3. 代数语义学

代数语义学是另一种数学语义的注释方法. 用代数方法对形式语言系统进行语义注释的语义学,便称之为代数语义学. 例如,我们可以定义一个逻辑系统,当然这个逻辑系统也是一个语言系统,我们就可以用范畴论的抽象代数概念对这个语言进行语义注释. 又例如,递归程序设计语言,我们可以用所谓的树文法描述递归程序设计语言程序的语法结构. 研究这种程序结构在语义域 I 的注释下,其代数特性(如分配律、交换律、结合律,等价性、条件等价性等等)的变化情况,也是一种代数语义学方法. 再例如,程序结构框图,就可以用所谓的图的重写 (rewriting) 系统(图范畴)进行注释其结构方面的语义.

根据代数语义学的发展状况,我们将在本书的第五章介绍代数语义学的基础,它们包括范畴论、图范畴、类别代数理论、抽象数据类型的说明与实现等内容.

对于函数型程序设计语言,也许读者已经知道 McCarthy 的 LISP 语言,这是一个纯表达式语言. 而表达式则可以用一个执行树来描述,可以借助于递归函数论及代数语义学的知识来理解 LISP 语言. Buckus 在 1978 年的 Turing 奖演说中提出了 FP (Functional Programming) 程序设计思想,即将一般的函数施用形式 $f(x)$ (LISP 语言就是采用这种施用形式的),定义成一种新的施用形式,即 $F:\langle x\rangle$. 例如一个表达式,采用 LISP 风格,则为

$$f(x, g(h(y), z))$$

其树结构如图 1.3.1 所示.

如果把图 1.3.1 树结构的叶子结点去掉,剩下的树不再有变量,如果用 F 来表示,将它施用三元组 $\langle x, y, z\rangle$ 之后与 $f(x, g(h(y), z)$有相同的计算结果. 可见$F:\langle x, y, z\rangle$本质意义在于将函数进行无变量(无形式参数)的抽象,也巧, Curry 的组合逻辑

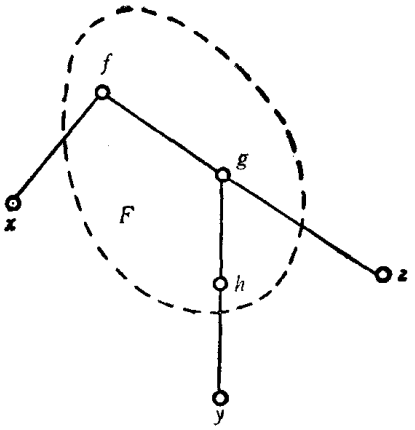

图 1.3.1

(Combinatory Logic)可以解决这个问题,这就是组合逻辑被关注的原因. 组合逻辑有可能成为新一代计算机的模型，我们将在第七章比较详细地介绍组合逻辑.

1.4. 操作语义学

操作语义是用机器模型语言来解释语言语义的，一般说来它与编译程序有直接关系. 在定义操作语义时，被采用的机器模型是 SECD 机器. Landin (1964 年)第一个在 SECD 机器上定义了 λ-表达式的操作语义，虽然现在看来这种定义方法并不那么美,也不如后来由 G. Plotkin 采用的归约系统方法清楚,但 Landin 的工作是有重大意义的.

操作语义学是所有语义学派别中最早出现的，在近几年又得到了新的发展.

操作语义学除定义"做什么"之外,主要是定义"怎么做",从这个意义上讲，属性文法属于操作语义学范畴. 我们在本书中将介绍属性文法的基本概念,属性文法的分类,如何用属性文法进行编译程序设计及定义语言.

1.5. 公理语义方法

公理语义方法是把程序设计语言视为一个数学对象，建立它的公理系统，从而使程序设计语言有坚实的逻辑基础．在这方面从事有意义工作的人不少，例如 Floyd, Z. Manna, Hoare, Dijkstra 等人．

由于 Martin-Löf 的研究成果，使我们比较清楚地认识到，一阶谓词演算因其不完全显式性及不完全构造性，并不十分适合于计算机与程序设计语言．为根本性地解决软件问题，除 Buckus 所说从 Von neumann 计算机中解放出来之外，Martin-Löf 又向我们说，应当创立更适合于计算机与程序设计语言的新数学．而 Martin-Löf 的类型论是构造数学的最新流派．不管别人怎么看待构造数学，而我们却认为从计算机科学的角度来说，构造数学派是应当受到欢迎的．计算机科学家对构造数学感兴趣的道理是完全可以理解的．

1.6. 形式说明语言

最早用于程序设计的形式说明语言是 VDL (Vienna Definition Lauguage)，这个语言曾被用于定义 PL/1 的操作语义．随着软件工程的发展，形式说明语言也多了起来．例如，属性文法定义语言 ALADIN，为 Ada 语言的编译程序定义过语义的 VDM 的元语言，还有 ALPHARD 抽象数据类型的定义语言，以及将逻辑与函数组合在一起的 LCF 语言、AFFIRM 语言等等．

由于篇幅及应用方面的原因，在本书中，只向读者较详细地介绍 VDM 的元语言．

第二章　指称语义学基础

在这一章，我们将向读者介绍指称语义学所用到的数学基础，主要内容包括两部分：

- 指称语义的论域(不动点理论).
- λ-注释及 λ-演算.

2.1. 论域问题引子

一个程序可以看成一个部分函数，例如一个函数 f，它的对应程序为 P_f. 如果 P_f 对于每一个输入都可以计算终止，那么称 f **是全函数**. 如果不是对于每一个输入都计算终止则称之为**部分函数**，即函数 f 在一些输入上无定义. 一个函数的论域有它的定义域与值域.

下面，我们首先给出两个小例子.

例 2.1.1　有如图 2.1.1 所示的有限时序机：

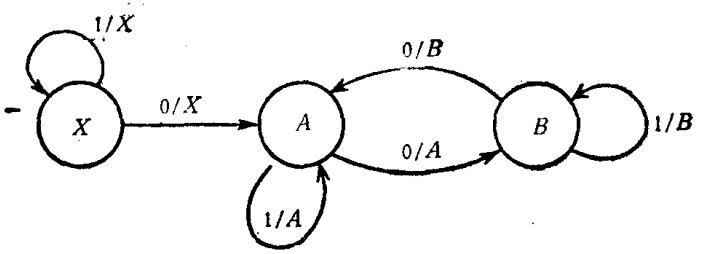

图　2.1.1

试编制一个程序实现该有限时序机的功能.

有人做出的程序是下面的样子：