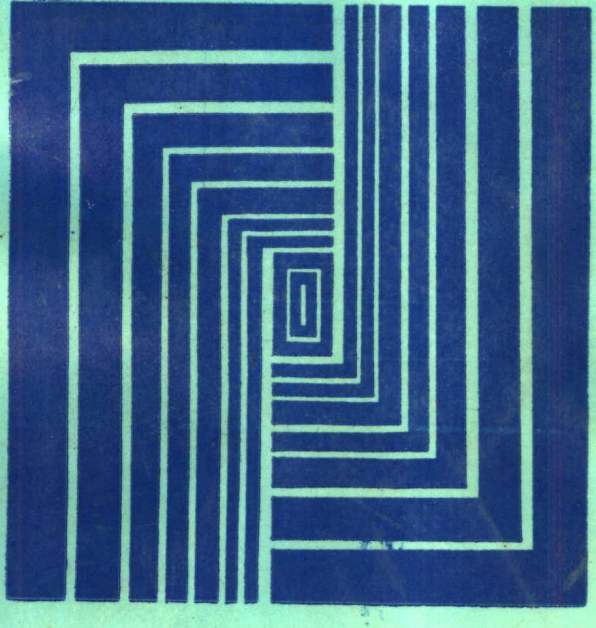
计算机科学中的范畴论

陈意云　编著



中国科学技术大学出版社

国家自然科学基金资助项目

计算机科学中的范畴论

陈意云　编著

中国科学技术大学出版社
1993·合肥

内 容 简 介

范畴论是近十年来兴起的计算机科学前沿研究方向之一，前景广阔．本书作者对这一领域做了很多研究工作，80年代末在美国期间收集了最新的有关资料．在此基础上写成的本书反映了范畴论作为工具应用于计算机科学的最新情况．

书中首先介绍代数规范的基本知识和一些泛代数的知识，然后系统地介绍了范畴论的主要内容：范畴、函子、自然变换、积与和、极限和余极限、伴随、笛卡儿封闭的范畴和素描等，并通过很多例子，介绍了范畴论在程序设计语言的语义、λ演算、论域理论、演绎系统和形式规范等方面的应用．各章节后面附有习题．

本书可作为计算机专业的高年级本科生、研究生的教材，亦可供从事计算机科学研究和开发的科技人员参考．

[皖]新登字08号

计算机科学中的范畴论

陈意云 编著

*

中国科学技术大学出版社出版

（安徽省合肥市金寨路96号，230026）

中国科学技术大学印刷车间印刷

安徽省新华书店发行

开本：850×1168/32 印张：8.75 字数：226千

1993年2月第1版 1993年2月第1次印刷

印数：1—5000册

ISBN7-312-00410-5/TP·48 定价：6.00元

前　言

本书包括两部分内容，代数规范和范畴论，重点是范畴论.

代数规范起源于七十年代中期. 它基于纯数学中的经典代数和泛代数的概念，基于计算机科学中抽象数据类型和软件系统的规范的概念. 在计算机科学和软件开发中，这是一个对理论和实际工作者都有意义的研究领域. 在代数规范方面迅速增长的兴趣促进了它的理论的形成. 本书介绍等式代数规范的基本概念和初始语义.

十九世纪末和二十世纪前半期，高等代数发展到近世代数(即抽象代数)阶段，代数工具深入地渗透到数学的其它领域. 数学在这一时间的一个突出特点是观察各种数学对象的普遍特征和相似性，强调各种数学对象之间的联系而不是孤立地分开研究. 于是在四十年代，反映数学各分支这种共性的一种理论，即研究各种数学结构之间联系的一般性理论产生了. 这就是范畴论.

范畴论为各数学学科提供了一种公共的语言、工具、思维方法和研究手段. 近年来范畴论也用于一些应用科学. 在计算机科学中，范畴论对于澄清数学结构、理论、语言和形式系统的联系，尤其是对于理解各种规范、设计、正确性证明和程序设计语言之间的联系是很有益处的.

本书是专为从事计算机科学的研究者和学生编写的基本范畴论的教材. 本书强调概念的理解而不是形式的证明，并经常提供不同的方式来思考和理解一个概念. 另外，本书还致力于一种特别的概念和方法，即用"素描"作为一种系统的方法来描述数学对象.

和其它有关范畴论的书相比，本书的另一特点是加入了代数规范的基础知识. 代数规范的语义是以范畴论为语言和工具来描述

的，这一部分的加入有助于了解范畴论的应用. 代数规范部分源于泛代数，泛代数是研究代数系统，如群、环、格等的共同特征的. 因此学习这一部分也有助于学习和理解范畴论本身. 此外，代数规范和上面提到的素描有密切的联系.

由于编者水平有限，书中难免还存在一些缺点和错误，恳请广大读者批评指正.

陈意云

1992 年 5 月

目　次

1 预备知识

本章是与代数规范和范畴论有关的预备知识，主要内容是：

1) 引入关于集合和函数的符号及术语，讨论有关集合的一些细致的问题，以免以后引起麻烦.

2) 回顾经典代数中的一些例子，非形式地引入代数规范.

1.1 集合与函数

集合概念在数学上通常取做是已知的. 我们不打算在这儿给集合下一个定义，而是给出一个足以胜任本书目的的集合的说明.

一个集合是一个数学实体，它区别于它的元素(如果有的话)，且完全由它的元素所确定. 对实体 x 和集合 S，$x\in S$ 是一个命题，其结果是真或假.

有限集合可以由枚举它的元素于花括号中来定义. 例如，$\{1,3,5\}$ 是有限集，它仅有的三个元素是数 1，3 和 5. 特别地，$\{\}$ 表示没有任何元素的集合，称为空集，用 ϕ 表示.

集合可由概括(*comprehension*)记号定义，表示集合由满足某一谓词的实体组成. 如果 x 是一变量，其取值范围是某一数据类型，$P(x)$ 是关于该数据类型的谓词，那么，记号 $\{x|P(x)\}$ 表示该数据类型中所有能使 P 为真的那些实体组成的集合. 例如，$\{x\in R|x>7\}$ 是所有大于 7 的实数的集合. 集合 $\{x|P(x)\}$ 称为谓词 P 的外延(*extension*).

由概括记号定义集合基于这样一个假设，即每个谓词决定一个集合. Russell 悖论揭示了其中的毛病. 该悖论用概括记号定义了一个东西，它不可能是集合：

$$\{S|S \text{ 是一个集合，并且 } S\notin S\}.$$

这个定义试图表示这样一个集合，它由不为自己的元素的集合所组成. 若它确实定义了集合 T，那么由定义，$T\in T$ 蕴涵 $T\notin T$，而 $T\notin T$ 又蕴涵 $T\in T$. 这个矛盾说明这样的集合 T 不存在.

避免这个悖论的一种简单办法是限制 x 到某个特定的数据类型，它已经形成一个集合，如实数和整数等不同的数系. 这样可以保证用概括记号定义的是集合.

定义 如果 S 和 T 是集合，S 和 T 的**笛卡儿积**(*cartesion product*)是有序二元组(s,t)的集合 $S\times T$，其中 $s\in S, t\in T$，即

$$S\times T=\{(s,t)\mid s\in S \text{ 且 } t\in T\}.$$

有序对(s,t)的第一坐标是 s，第二坐标是 t. 有序对由它的两个坐标唯一地确定. 这本质上是有序对的一个说明. 积的形式化的范畴论定义是基于这一点的.

更一般地，一个有序 n 元组是一个序列$(a_1,\cdots,a_n)$，它唯一地由它的 n 个坐标 $a_i(i=1,\cdots,n)$确定. 笛卡儿积 $S_1\times S_2\times\cdots\times S_n$ 是所有 n 元组$(a_1,\cdots,a_n)$的集合，其中 $a_i\in S_i(i=1,\cdots,n)$.

我们用 $\boldsymbol{N}$ 表示自然数集合，$\boldsymbol{Z}$ 表示整数集合，$\boldsymbol{Q}$ 表示有理数集合，$\boldsymbol{R}$ 表示实数集合.

下面我们转入函数.

函数 f 是具有下列性质的数学实体.

1) f 有一**论域**(*domain*)和一**余论域**(*codomain*)，它们都必须是集合.

2) 对论域的每个元素 x，f 有一**值**，该值是余论域的一个元素，表示为 $f(x)$.

3) 论域、余论域和论域的每个 x 的值 $f(x)$都完全由这个函数确定.

4) 反之，构成论域和余论域的数据，以及论域中每个 x 的值 $f(x)$完全确定了这个函数.

论域和余论域通常分别称为 f 的**源**(*source*)和**目标**(*target*). 记号 $f:S\to T$ 表示具有论域 S 和余论域 T 的函数 f. 该记号强调了类

型，也就是 f 是从 S 到 T 的一个函数.

我们将用带竖线的箭头给函数提供一个匿名记号. 例如，从 R 到 R 的平方函数可以表示为 $x\mapsto x^2 : R\to R$. 箭头是从论域到余论域，而带竖线的箭头是从数据到数据.

上面关于函数的性质 3)强调了函数是一种规则连同它的论域和余论域，而又不单是规则. 这是范畴学家关于函数的观点，它强调函数的类型特征，并非所有的数学家都持这种观点. 例如，函数

1) $x\mapsto x^2 : \boldsymbol{R}\to\boldsymbol{R}^+$,

2) $x\mapsto x^2 : \boldsymbol{R}\to\boldsymbol{R}$,

3) $x\mapsto x^2 : \boldsymbol{R}^+\to\boldsymbol{R}^+$,

4) $x\mapsto x^2 : \boldsymbol{R}^+\to\boldsymbol{R}$

是四个不同的函数(其中 $\boldsymbol{R}^+$ 是非负实数集合). 通常，大学的数学教科书不区分它们，也的确没有理由要区分它们. 但是在范畴论和抽象数学的其它分支中，这种区分成为必要.

注意，这种区别甚至在更基本的情况下也是有用的. 例如，每个集合 S 有一个**恒等函数**(*identity function*)$\mathrm{id}_S : S\to S$，对所有的 $x\in S$，$\mathrm{id}_S(x)=x$. 如果 S 是 T 的子集，那么有一个**包含函数**(*inclusion function*)$\mathrm{i} : S\to T$，对所有的 $x\in S$，$\mathrm{i}(x)=x$. 即使对公共论域上的每个元素，id_S 和 i 有相同的值，它们仍是不同的函数，因为它们有不同的余论域.

定义 **函数 $f : S\to T$ 的图**是有序对集合 $\{(x,f(x))\mid x\in S\}$.

函数的图具有函数性，即对所有的 $s\in S$，只有一个 $t\in T$，使得 (s,t) 在该图中. 许多教科书(不包括本书)用具有这种函数性的关系来定义函数，某些书称函数为映射.

定义 函数的**象**(也称为值域)是它的值集，也就是，$f : S\to T$ 的象是 $\{t\in T\mid \exists\ s\in S$ 使得 $f(s)=t\}$. 前面提到的四个函数的象当然都是非负实数集.

定义 函数 $f : S\to T$ 是**内射的**(*injective*)，如果每当 S 中 $s\neq s'$ 就有 T 中 $f(s)\neq f(s')$.

内射的另一个名字是**一到一**(*one to one*). 先前描述的恒等函数和包含函数是内射函数. 函数 $x\mapsto x^2 : R\to R$ 不是内射函数,因为 2 和 -2 的值都是 4. 另一方面,$x\mapsto x^3 : R\to R$ 是内射函数. 对任何集合 T,有一个唯一的函数 $e : \phi\to T$,它没有值,也是内射的.

所有的函数都有性质:如果 $s=s'$,那么 $f(s)=f(s')$. 不要把内射的定义和这个性质混淆. 内射函数的另一说明方式是用内射定义的换质位:如果 $f(s)=f(s')$,那么 $s=s'$.

定义 函数 $f:S\to T$ **是满射的**(*surjective*),如果它的象是 T.

恒等函数是满射的,但任何其它的包含函数都不是满射的. 函数是否满射取决于余论域. 例如,前面提到的四个平方函数,只有第 1 和第 3 两个是满射的. 满射函数通常叫做**映成函数**.

一个函数是**双射的**(*bijective*),如果它内射且满射. 双射函数又叫做——**对应**(*one to one correspondence*)**函数**.

如果 S 和 T 是集合,**笛卡儿积** $S\times T$ 装备有两个**坐标或射影函数** $\mathrm{proj}_1 : S\times T\to S$ 和 $\mathrm{proj}_2 : S\times T\to T$. 如果 S 和 T 都非空,那么这两个射影函数都是满射的. 多于两个集合的积的射影函数可以类似地定义.

还有两个记号和笛卡儿积相联系.

定义 如果 X,S 和 T **是集合**,$f : X\to S$ 和 $g : X\to T$ 是函数,那么函数 $\langle f,g\rangle : X\to S\times T$ **定义为**:对所有的 $x\in X$,$\langle f,g\rangle(x)=(f(x),g(x))$.

定义 如果 X,Y,S 和 T **是集合**,$f : X\to S$ 和 $g : Y\to T$ **是函数**,那么函数 $f\times g : X\times Y\to S\times T$ **由** $(f\times g)(x,y)=(f(x),g(y))$ **定义**. 它叫做**函数** f 和 g **的笛卡儿积**.

这两个函数在第 7 章再进一步讨论.

定义 如果 $f : S\to T$ 且 $g : T\to U$,那么**合成函数**(*composite function*) $g\circ f : S\to U$ 唯一地定义为:对所有的 $x\in S$,$(g\circ f)(x)=g(f(x))$. 在计算机科学的文献中,常用 $f;g$ 代替 $g\circ f$.

范畴论基于合成,把合成作为一个基本的运算,就象经典集合论

基于从属关系那样. 在范畴论中,对于合成 $g\circ f$,必须坚持 f 的余论域是 g 的论域,就象上面所定义的那样. 在数学的其它分支的许多教科书中,仅要求 f 的象包含在 g 的论域中.

定义　如果 $f:S\to T$ 且 $A\subseteq S$,那么合成函数 $f\circ i$ 叫做 f(到 A)的限制(*restriction*),其中 $i:A\to S$ 是包含函数. 前面提到的第 3 个平方函数是第 1 个平方函数限制到 R^+.

类似地,如果 $T\subseteq B$,那么 f 叫做函数 $j\circ f:S\to B$(到 A)的余限制(*corestriction*),其中 $j:T\to B$ 是包含函数. 前面提到的第 1 个平方函数是第 2 个平方函数余限制到 R^+.

习　题

1. 令 S 和 T 都是集合,用 $Hom(S,T)$ 表示所有论域为 S 和余论域为 T 的函数的集合. 令 $f:T\to V$ 是一个函数,定义函数

$$Hom(S,f):Hom(S,T)\to Hom(S,V)$$

为　　$$Hom(S,f)(g)=f\circ g.$$

注意,$Hom(S,x)$ 是超载的记号:当 x 是集合,$Hom(S,x)$ 是函数的集合;当 x 是函数,$Hom(S,x)$ 也是函数. 证明,若 S 不是空集,f 是内射的当且仅当 $Hom(S,f)$ 是内射的.

2. 仍用习题 1 的记号,令 $h:W\to S$ 是一个函数,定义 $Hom(h,T):Hom(S,T)\to Hom(W,T)$ 为 $Hom(h,T)(g)=g\circ h$. 证明:如果 T 至少有两个元素,那么 h 是满射的当且仅当 $Hom(h,T)$ 是内射的.

3. 1)用习题 1 的记号,证明:把函数对 $(f:X\to S,g:X\to T)$ 映射到函数 $\langle f,g\rangle:X\to S\times T$ 的函数是从 $Hom(X,S)\times Hom(X,T)$ 到 $Hom(X,S\times T)$ 的双射.

2) 在 1)中,令 $X=S\times T$,在此双射下,$\mathrm{id}_{S\times T}$ 对应什么?

1.2　经典代数

本节回顾经典代数中的一些例子和计算机科学中使用的一些数

系，非形式地引入代数规范.

例 1.1 我们从半群开始. 一个**半群**$(A, *_A)$由基集 A 和二元运算 $*_A : A\times A\to A$ 组成，该运算是可结合的，也就是对任意 $a_1, a_2, a_3 \in A$，有

$$(a_1 *_A a_2) *_A a_3 = a_1 *_A (a_2 *_A a_3).$$

自然数及其加法$(N, +_N)$和自然数及其乘法$(N, \times_N)$都是半群的例子. 另一个熟知的例子是，给定字母表 A 的所有非空串集 A^+和并置运算 $\cdot_{A^+}$构成自由半群$(A^+, \cdot_{A^+})$.

上面给出的是半群的语义表示，下面给出半群的语法表示，叫做半群的规范. 有穷的语法表示在计算机科学中十分重要. 在语法表示中，用一个符号 s 代替基集 A，称为 A 的类；用运算符号 $* : s\ s\to s$ 代替二元运算 $*_A : A\times A\to A$，其中 $*$ 是名字，$s\ s\to s$ 是它的声明，箭头左边的两个 s 是源类，右边的 s 是目标类. 如果只有一个类，这种表示有点冗余，但在后面的例中有不同的类出现，在那儿运算符号的声明必须显式地给出. 因此我们一开始便用这种一般的表示法.

剩下的是用语法项表示结合性. 出于这个目的，我们把 $m1, m2$ 和 $m3$ 看成类 s 的变量，使得结合性可以用"等式"

$$(m1 * m2) * m3 = m1 * (m2 * m3)$$

表示. 把这称为等式有点误用，因为$(m1 * m2) * m3$ 和 $m1 * (m2 * m3)$是完全不相同的语法项. 但是，如果分别指派变量 $m1, m2$ 和 $m3$ 到 $a_1, a_2, a_3 \in A$，用运算 $*_A : A\times A\to A$ 代替算符 $* : s\ s\to s$，那么我们可以得到语义级的等式

$$(a_1 *_A a_2) *_A a_3 = a_1 *_A (a_2 *_A a_3).$$

这样就导致了下面的半群的规范

$$\begin{aligned}
&semigroup=\\
&\quad sorts : s\\
&\quad funs : * : s\ s\to s\\
&\quad eqns : m1, m2, m3 \in s\\
&\qquad (m1 * m2) * m3 = m1 * (m2 * m3)
\end{aligned}$$

例 1.2　在上例中，如果我们加上一个常量符号 e 和两个等式 $e * m = m$ 和 $m * e = m$，那么可以从半群的规范得到**含幺半群**(*monoid*)的规范. 为了避免重复半群的规范，我们写成

$$
\begin{aligned}
&monoid = semigroup +\\
&\quad \textbf{\textit{funs}}:\ e : \to s\\
&\quad \textbf{\textit{eqns}}:\ m \in s\\
&\qquad\qquad e * m = m\\
&\qquad\qquad m * e = m
\end{aligned}
$$

在这儿(以及在以后的例中)，+表示类、运算符号和等式的不相交的并集.

含幺半群$(A, *_A, e_A)$是一个半群$(A, *_A)$，元素 $e_A \in A$ 满足上面的幺元等式. $(N, +_N, 0)$和$(N, \times_N, 1)$都是含幺半群的例子. 如果我们用 ε 表示空串，令 $A^* = A^+ \cup \{\varepsilon\}$，并且把并置运算扩充到空串，那么$(A^*, \cdot, \varepsilon)$是含幺半群，称为 A 上的自由含幺半群.

例 1.3　继续例 1.2. 逆元的存在与唯一性可以由算符 $()^{-1} : s \to s$ 及其相应的公理表示. 这样可以得到群的规范

$$
\begin{aligned}
&group = monoid +\\
&\quad \textbf{\textit{funs}}:\ ()^{-1} : s \to s\\
&\quad \textbf{\textit{eqns}}:\ m \in s\\
&\qquad\qquad m * m^{-1} = e\\
&\qquad\qquad m^{-1} * m = e
\end{aligned}
$$

群$(A, *_A, e_A, ()_A^{-1})$是带有一元运算 $()_A^{-1} : A \to A$ 的含幺半群，该一元运算满足上面的逆元等式. $(\mathbf{Z}, +, 0)$是含幺半群，若运算 $- : \mathbf{Z} \to \mathbf{Z}$ 定义为 $-(z) = -z (z \in \mathbf{Z})$，那么$(\mathbf{Z}, +, 0, -)$是群. 不包括 0 的实数 $\boldsymbol{R}$ 定义一个群$(\boldsymbol{R} - \{0\}, *, 1, ()^{-1})$，其中 $r^{-1} = \frac{1}{r} (r \in \boldsymbol{R} - \{0\})$.

例 1.4　一个代数结构$(A, *_A, z_A, +_A, -_A)$，如果二元运算 $+_A$ 是可结合的和可交换的，z_A 是零元，一元运算 $-_A$ 是求逆元，两元运算 $*_A$ 是可结合的，且对 $+_A$ 是可分配的，那么称这种代数结构为环.

环的规范如下：

$$
\begin{aligned}
&ring = semigroup + \\
&\quad funs: \ z: \to s \\
&\qquad\quad + : s\ s \to s \\
&\qquad\quad - : s \to s \\
&\quad eqns: \ m, m1, m2, m3 \in s \\
&\qquad\quad (m1+m2)+m3 = m1+(m2+m3) \\
&\qquad\quad m1+m2 = m2+m1 \\
&\qquad\quad m+z = m \\
&\qquad\quad m+(-m) = z \\
&\qquad\quad (m1+m2)*m3 = (m1*m3)+(m2*m3) \\
&\qquad\quad m1*(m2+m3) = (m1*m2)+(m1*m3)
\end{aligned}
$$

$(Z, *, 0, +, -)$和$(R, *, 0, +, -)$是熟知的环的例子. 实数集还是域的典型例子. 域是一个交换环，其乘法有幺元，对所有的非零元素有乘法逆元. 但是域的规范不可能仅由等式给出，因为乘法的求逆运算是一个部分函数.

下面我们转向数据类型的例子. 自然数、整数、有理数、实数和复数，加上它们的算术运算，是数学上众所周知的数系. 在计算机科学中，通常我们不把实数和复数这两个数系与数据类型联系起来，因为它们不是可枚举的. 另一方面，我们考虑布尔值 B 及布尔运算和一些其它的数据类型，如自然数模 m. 下面给出几个数据类型的规范.

例 1.5 首先考虑布尔值和运算的基本规范

$$
\begin{aligned}
&bool = \\
&\quad sorts: \ bool \\
&\quad funs: \ TRUE, FALSE: \to bool \\
&\qquad\quad NOT: bool \to bool \\
&\qquad\quad AND: bool\ bool \to bool \\
&\quad eqns: \ b \in bool
\end{aligned}
$$

$$NOT(TRUE)=FALSE$$

$$NOT(NOT(b))=b$$

$$b\ AND\ TRUE=b$$

$$b\ AND\ FALSE=FALSE$$

代数 $BOOL=(B,true,false,\neg,\wedge)$是满足这个规范的一个代数，其中 $B=\{true,false\}$，$\neg$和$\wedge$分别是逻辑否定和逻辑与运算. 可以增加一些逻辑运算符号到这个规范，如 OR(代表$\vee$)，$IMPLIES$(代表$\Rightarrow$)和 $EQUIV$(代表$\Leftrightarrow$)，而使之丰富起来.

例 1.6 有零、后继和加法的自然数的规范如下.

$nat=$

sorts : nat

funs : $0:\rightarrow nat$

$SUCC:\ nat\rightarrow nat$

$ADD:\ nat\ nat\rightarrow nat$

eqns : $n,m\in nat$

$ADD(n,0)=N$

$ADD(n,SUCC(m))=SUCC(ADD(n,m))$

$NAT=(N,0,+1,+)$是满足这个规范的一个代数.

自然数模 m 的规范如下.

$natmod(m)=nat+$

eqns : $SUCC^m(0)=0$

其中 $SUCC^m(0)$是有 m 重 $SUCC$ 的 $SUCC(\cdots SUCC(0)\cdots)$的缩写.

对于数据类型的规范和经典代数的规范，在语义级上我们的考虑是有区别的. 在下一章的最后一节，我们讨论这个问题.