非线性报表模型原理

蒋步星　著

中国科学技术出版社
·北　京·

图书在版编目(CIP)数据

非线性报表模型原理/蒋步星著. —北京:中国科学技术出版社,2007.5

ISBN 978-7-5046-4605-7

Ⅰ.非… Ⅱ.蒋… Ⅲ.计算机算法理论 Ⅳ.F231.5-39

中国版本图书馆 CIP 数据核字(2007)第 000442 号

中国科学技术出版社出版

北京市海淀区中关村南大街 16 号 邮政编码:100081

电话:010-62103210 传真:010-62183872

科学普及出版社发行部发行

北京长宁印刷有限公司印刷

*

开本:850 毫米×1168 毫米 1/32 印张:4.25 字数:100 千字

2007 年 5 月第 1 版 2007 年 5 月第 1 次印刷

印数:1-600 册 定价:18.00 元

内 容 简 介

本书全面系统地阐述了非线性报表模型的数学原理。从基本概念出发,逐步引出报表及相关概念的严格数学定义,然后提出能充分体现当前复杂报表内在规律的全新数学模型和相关的实用算法,并证明其合理性及分析运算复杂度。

非线性报表模型的提出为报表软件学术界引入一个新的研究课题,填补了报表软件业的一大空白,为报表软件技术的进一步发展研究奠定了理论基础。

本书主要面向计算机科学领域的研究人员和软件开发人员,具有很高的学术价值和社会意义。对于应用相关产品的用户也有重要的指导意义。

作者简介

蒋步星:非线性报表模型的主要设计者。1989 年获得国际数学奥林匹克竞赛(IMO)金牌;1996 年毕业于清华大学计算机系并获硕士学位。长期从事报表软件的研究和开发工作。现任北京润乾软件技术有限公司董事长。

前　言

以报表为载体进行数据统计与填写是信息系统中不可缺少的重要环节。

随着应用软件技术的不断发展，报表技术经历了从直接编程、工具与编程相结合、主要依靠报表工具三个不同的阶段。然而，目前业界流行的各类报表工具在设计报表时有着巨大的困难。究其根本原因，我们认为主要由于这些工具采用的数学模型过于简单，没有体现复杂报表的内在规律，难以方便地描述现实应用中的报表，致使报表的开发仍需大量编码。使用这些工具并不能起到提高效率的作用。

在经历了五年多的摸索和实践、研究分析了数以千计的报表实例之后，我们提出了拥有完全自主知识产权的全新理论体系，即非线性报表模型。

由于非线性报表模型充分体现了复杂报表的规律，因而基于该理论完成报表工具能够方便地描述绝大多数复杂报表，从而极大地改进报表的设计方式，使大量复杂的报表都不再依靠繁琐的代码实现，做到真正意义地零编码，大幅度提高了报表的设计效率，实现了报表技术的跨越性发展。

本书将从理论上系统地阐述非线性报表模型的数学原理。全书分为 6 章。第 1 章回顾和介绍需要用到的基本数学概念；第 2 章给出报表的严格数学定义和基本变换算法；第 3 章研究报表扩展变换的算法及相关性质；第 4 章介绍在扩展后报表中进行运算引用的方案；第 5 章讨论报表计算和填写的算法并分析其复杂度；第 6 章作一个总结和展望。

本书中仅进行纯粹的理论探讨，而不涉及软件的实现和应用（尽管高效地实现也是非常重要的）。这些理论可以很好地指导

报表工具的设计与实现。事实上，润乾报表正是一个非线性报表模型的实例产品。

最后，我要感谢润乾报表的用户们——是他们在各种复杂、关键信息系统中的实践，充分发挥了非线性报表模型的作用；而用户的需求又进一步发展丰富了模型的内容。从而使得这一全新的理论正迅速被业界认可并逐步成为报表工具的标准。

我还要感谢我在北京润乾软件技术有限公司的同事们——是他们精湛的程序实现能力，使非线性报表模型不仅存在于理论中，而是实实在在地变成了可实用的软件产品，以实例的形式充分证明了模型的可行性。

我还要感谢在学校给予我指导和交流的老师与同学们——与他们共渡的数学训练培养了我严格抽象的思维能力，而这对于非线性报表模型理论的提出和完善是至关重要的。

2007 年 5 月 8 日

目　录

1 准 备

1.1 集合

1.1.1 集合

作为基本概念，我们不对集合作严格定义，集合概念的研究远远超出了本文范围，这里仅对文中需要用到的术语和符号作一些回顾和约定。

一些确定的事物汇集成集合，构成集合的事物称为集合的元素或成员。我们在{}中写上元素表示集合，如{1,2,3,…}是所有自然数的集合。

设 x 为一事物，P 为一性质，$P(x)$表示 x 具有性质 P；用$\{x|P(x)\}$表示一切具有性质 P 的事物构成的集合。

x 是集合 X 的元素时称 x 属于 X，记作 $x\in A$，x 不是集合 X 的元素时称 x 不属于 X，记作 $x\notin X$。

用$|X|$表示集合 X 的元素个数（$|x|$同时仍是绝对值运算符），元素个数为 0 或自然数的集合称为有限集。本文中涉及的集合绝大多数是有限集。

没有元素的集合称为空集，即{}，记作ϕ，显然$|\phi|=0$。

我们有时会在不产生混淆时忽略仅有唯一元素 x 的集合 $X=\{x\}$与其元素 x 的差别，即用 X 表示 x 或用 x 表示 X；在需要明确区分两者时，则用符号$_{\}}X_{\{}$来表示 x。

设有两个集合 A 与 B：

1. 若对任何 $a\in A$，均有 $a\in B$，则称 A 包含于 B 或 B 包含 A，记作 $A\subseteq B$ 或 $B\supseteq A$，同时称 A 是 B 的子集而 B 是 A 的超集；

2. 若 $A\subseteq B$ 且 $B\subseteq A$，则称 A 与 B 相等或 A 等于 B（或 B 等于 A），记作 $A=B$；否则称 A 不等于 B，记作 $A\neq B$；

显然当 $A=B$ 时，A 与 B 拥有完全相同的元素；

3. 若 $A\subseteq B$ 且 $A\neq B$，则称 A 真包含于 B 或 B 真包含 A，记作 $A\subset B$ 或 $B\supset A$，同时称 A 是 B 的真子集；

4. 集合$\{x|x\in A$ 或 $x\in B\}$称为 A 与 B 的并集，记作 $A\cup B$；

5. 集合$\{x|x\in A$ 且 $x\in B\}$称为 A 与 B 的交集，记作 $A\cap B$；

6. 集合$\{x|x\in A$ 且 $x\notin B\}$称为 A 与 B 的差集，记作 $A\backslash B$；

除以上集合论中通行概念外，我们再定义几个本文中将会用到的术语：

7. 若 $A\cap B\neq\phi$，则称 A 与 B 相交，否则称 A 与 B 相离；

8. 若 $A\subseteq B$ 或 $B\subseteq A$，则称 A 与 B 相含；

9. 若 A 与 B 相交但不相含，则称 A 与 B 交叉。

对任何两个实数 a,b，集合$\{x|a\leqslant x\leqslant b\}\cup\{x|a\geqslant x\geqslant b\}$称为从 a 到 b 的区间，记作$[a,b]$；实数 $x\in[a,b]$时称 x 落在区间$[a,b]$内。

自然数集用大写字母 **N** 表示。

1.1.2 数组

将有限集的元素按某种规则排序，得到一个有先后次序的序列，称为数组；数组的成员个数称为数组的长度。

由于有序性，对于数组可使用形如第 i 个成员的术语，i 是第 i 个成员的序号。与无序的集合不同，我们在()中按序写上成员来表示数组，如$(a_1,a_2,\ldots,a_n)$。

数组的相等概念需要重新定义，称数组$(a_1,a_2,\ldots,a_n)$和$(b_1,b_2,\ldots,b_m)$相等当且仅当$n=m$且$a_1=b_1,a_2=b_2,\ldots,a_n=b_m$。

长度为n的数组称为n元组。特别地，2元组又称为序偶，其成员按次序分别称为左元和右元。

n元组一般用于定义以其为元素的集合。称某集合由n元组构成，即表明作为该集合的元素的数组的长度均相同且为n，而仅称由数组构成时则不蕴涵其中元素的长度相同。

设有n个集合$X_1,X_2,\ldots,X_n$，称n元组构成的集合

$$\{(x_1,x_2,\ldots,x_n)|x_1\in X_1,x_2\in X_2,\ldots,x_n\in X_n\}$$

为集合$X_1,X_2,\ldots,X_n$的叉积或笛卡儿积，记作$X_1\times X_2\times\ldots\times X_n$。

由多个n元组构成的数组称为矩阵。矩阵本身及其成员都是数组，矩阵的长度称为行数，其成员的长度称为列数。矩阵的成员又称为行，行的序号称为行号，由矩阵各行中相同序号的成员以其所在行次序为次序构成的长度为矩阵行数的数组称为矩阵的列，其成员在行中的序号（都相同）称为列号。

1.1.3 变换及符号

作为算法过程的基础动作，我们定义一些集合及其元素的基本变++换及符号：

1. S+=a

若$a\notin S$，则在S中添入a，如果S是数组，未明确指明和约定时缺省地认为a将被加到S最后。

2. S−=a

若 $a \in S$，则从 S 中删除 a。

3. $a \to b$

对上下文中所有集合 S（包括作为其他集合元素的集合，以下同），若 $a \in S$，则将 S 中的 a 替换成 b；注意这个动作不是简单地顺序执行 $S-=a$ 和 $S+=b$，S 可能是数组，替换时要保持原有次序（专门说明的除外）。

4. $a \to \times$

对上下文中所有集合 S 执行 $S-=a$，若 S 是 n 元组，则再递归地执行 $S \to \times$；注意这里变化的是以 a 为元素的集合而非 a 本身。显然，若上下文中的集合都是有限集且出现的集合数量有限，该动作经过有限步后一定会停止。

注意不要将这里的变换符号 $\to$ 与通行的符号 $\Rightarrow$ 混淆，后者将用于表示证明过程中的推出（蕴涵）。

1.2 函数

1.2.1 函数

设有两个集合 X 和 Y，如果按照某种规律 f，对任何 $x \in X$，均存在某个 $y \in Y$ 与之对应，则称有一个定义在 X 上在 Y 中取值的函数或映射，仍用 f 表示；集合 X 称为 f 的定义域，X 中的元素称为 f 的自变量或参数（本文更常用后者），与 x 对应的 y 称为函数 f 在 x 上的返回值，记作 $f(x)$；集合 $\{f(x)|x \in X\}$ 称为 f 的值域，记作 $f(X)$；如果对任何 $x_1 \neq x_2$ 均有 $f(x_1) \neq f(x_2)$，则称 f 是可逆函数。

当集合 X 与 Y 分别是某可逆函数 f 的定义域和值域时，则称 X 与 Y 基于 f 一一对应，不关心 f 时也仅称 X 与 Y 一一对应。显然两个一一对应的集合元素数量相同，而且若集合 X 与 Y 一一对应，Y 与 Z 一一对应，则 X 与 Z 也一一对应。

原则上，定义一个函数需要指出其定义域、取值范围（值域的某个超集）以及对应关系，但定义域和取值范围通常可从上下文中看出，这样我们只要指明函数中的对应关系即可。

定义域是 n 个集合叉积的某个子集的函数 f 被称为 n 元函数，这时我们习惯于用 $f(x_1,\ldots,x_n)$ 代替 $f((x_1,\ldots,x_n))$ 表示其返回值，并称 f 有 n 个参数 $x_1,\ldots,x_n$。特别地，当 $n=1$ 时称 f 为单元函数。

某个参数总是集合（或数组）的函数（n 元）称为聚合函数。

设某有限集 $\{x_1,\ldots,x_n\}$ 为单元聚合函数 f 的参数，在不与 n 元函数相混淆的情况下，我们有时会用 $f(x_1,\ldots,x_n)$ 代替 $f(\{x_1,\ldots,x_n\})$ 表示其返回值。注意，作为同一个聚合函数参数的有限集的元素个数不一定相同，这与 n 元函数是不同的。

设 X 是由实数构成的有限集，在其上定义常用的聚合函数：

1. $\mathrm{sum}(X):=X$ 中所有数的和；
2. $\min(X):=X$ 中最小的数；
3. $\max(X):=X$ 中最大的数；
4. $\mathrm{avg}(X):=X$ 中所有数的平均值。

这里符号:=用于表示函数的定义，下文中也会采用该符号。

所有返回值都是数组的函数称为集合函数，定义两个常用的集合函数：

1. $\mathrm{list}(v_1,v_2,\ldots,v_n):=(v_1,v_2,\ldots,v_n)$；
2. $\mathrm{to}[q](n,m)$，其中 n 和 m 是同一个数组 q 中的两个成员；返回值为 q 中 n 和 m 之间的成员依原序排列而成的数组，包括 n,m，q 在上下文中意义明确时，$[q]$ 部分可不写，如

to(2001,2005)=(2001,2002,2003,2004,2005)。

需要指出的是，聚合函数和集合函数是两个无关概念，聚合函数未必一定是集合函数，反之亦然。这两个概念对于报表模型的建立至关重要，集合函数是报表自动化的基础，而聚合函数则用于处理表中不可或缺的统计运算。

1.2.2 表达式

与集合类似，数据也是一个不给定义的基本概念，它能够通俗地解释为信息系统中处理的信息基本单元，原则上可以是任何事物。

每个数据均有数据类型，不同类型的数据在处理过程中允许进行的运算和遵循的规则不同，不同类型可以转换。常见的数据类型有：

1. 数：即实数，可进行各种数学运算；
2. 字符串：以字符为成员的数组，有连接、比较等运算；
3. 逻辑：真假值，可进行与、或、非等运算；
4. 集合：数据本身是一个集合，可进行交、并等运算；

事实上，由于数据的多样性，数据类型也可能有无穷多种。

字符串是很重要且常用的数据类型，为书写方便，我们简单地将其成员顺序地写在一起，即用 $a_1a_2…a_n$ 表示字符串 $(a_1,a_2,…,a_n)$，有时为与其他类型数据区分还可能加上引号，即写成"$a_1a_2…a_n$"。

设字符串 $A=a_1a_2…a_n$ 和 $B=b_1b_2…b_m$，称字符串 $a_1…a_nb_1…b_m$ 为 A 与 B 的连接，记作 $A+B$，显然$|A+B|=|A|+|B|$。

将代表数据、函数等基本项的符号用一些运算操作符号按一定规则连接起来即可获得一个表达式。在一定条件下（符号代表的数据都已确定），表达式可被计算得到一个数据，称为表达式的计算结果。

我们这里不对表达式的概念及书写规则作详细讨论（可计算理论中有专门论述），而是简单延用计算语言化的数学表达式。如 3+5*a、2*ln(x)等，与纯粹的数学表达式不同，这里乘号用*表示且不能省略，而函数参数一定要用括号括起来，运算符（如逻辑运算与,或,非等）则采用 C 语言的符号体系（&&,||,!等）。

表达式的计算结果不一定是个简单的数值，原则上可以是任何数据，如字符串、数组、图片等。计算结果是数组的表达式称为集合运算，集合函数即可看作是一种集合运算。计算结果是逻辑值的表达式称为逻辑表达式。

由于表达式书写时是一串连续的符号，因而可以用字符串的形式描述。内容为一个表达式的字符串称为计算串。

定义函数 eval(x)，参数 x 是个计算串，返回值是将 x 的内容作为表达式计算后的计算结果。作为一个数据，eval 函数的返回值仍可能是计算串（字符串），可以作为 eval 的参数再次计算。

1.2.3 属性

设 p 是两集合 X,Y 的叉积 $X\times Y$ 的子集，若 p 中没有两个元素有相同的左元，则称 p 是定义在集合 X 上的一个属性。集合 $\{x|(x,y)\in p\}$称为 p 的定义集，$\{y|(x,y)\in p\}$为 p 的取值集；设$(x,y)\in p$，称 y 是 x 的 p 属性值，简称 x 的 p，记作 $x.p$ 或 $p(x)$；此时也称 x 有 p 属性或 $x.p$ 有定义；若没有 y 使得$(x,y)\in p$，则称 x 无 p 属性

或 *x.p* 无定义。

仔细考查属性和函数的定义，可以发现两者并无本质不同。事实上，属性和函数确实是等价概念，其差别只在于应用场合不同。函数更强调参数与返回值间的某种对应关系，大多数情况下这种对应关系有规律可循（如可用表达式描述）；而属性则更多地理解为事物本身的某种性质，其对应关系通常没有规律，需要逐一设置其属性值。

与函数类似，我们在定义属性时经常并不明确指出其定义集，而仅说为某些确定的元素定义某属性，并给出这些元素的相应属性值或取值集，属性的定义集可从上下文中看出。

我们尽量采用与函数不同的写法 *x.p* 表示属性，但有时文字中符号优先级可能混淆时，也会采用写法 *p*(*x*)以用括号明确符号的作用次序，两种写法是等价的。

我们在实际应用中经常会将属性值写成表达式的形式，称为属性表达式，其计算结果即是属性值。由于任何数据值都可理解为常数表达式，因而可认为任何属性都有表达式形式。一般情况下，我们在模型讨论中无需刻意区分属性表达式和属性值，这时两者都用符号 *x.p* 表示；而在需要明确区分时，属性表达式更为常用，此时我们约定符号 *x.p* 表示的是元素 *x* 的 *p* 属性表达式（可能是常数），而用 *x.p* 表示属性值（计算结果）。

取值集为逻辑值{真,假}的属性称为逻辑属性，定义集上的所有元素的属性表达式总是常数的属性称为常数属性。

约定如下与属性相关的变换动作及符号：

1. $x.p=v$

1)若 $x.p$ 有定义则做 $p^{-}=(x,x.p)$；

2)做 $p+=(x,v)$。

2. $a\leftarrow b$

1)对上下文中所有使 $a.p$ 有定义的属性 p 做 $p^{-}=(a,a.p)$；

2)对上下文中所有使 $b.p$ 有定义的属性 p 做 $p+=(a,b.p)$。