信息与逻辑丛书

# 可计算性理论

张宏裕 编著

天津科学技术出版社

责任编辑：黄立民

信息与逻辑丛书

可计算性理论

张宏裕　编著

*

天津科学技术出版社出版

天津市赤峰道130号

天津市武清县永兴印刷厂印刷

新华书店天津发行所发行

*

开本850×1168毫米　1/32　印张7.125　字数173 000

1989年5月第1版

1989年5月第1次印刷

印数：1－2 620

ISBN 7-5308-0384-0/TP·13　定价：3.10元

# 目　录

# 前　言

本书的内容，大体上可分成两大部分。第一部分是第一到第五章以及第六章的第一、第二两节，这部分可作为可计算函数理论的基本教材，适合初学者自学之用；第二部分是第六章的第三、第四两节以及第七、第八两章，这部分是可计算函数理论中深入的内容，初次阅读时可以省去。

当前有许多从事计算机工作的同志，过去没有机会系统学习可计算函数理论，而实际工作中又迫切需要这方面的知识，本书就是为他们写的基本教材。具有本丛书的第一本：离散数学引论的知识即可阅读本书。

第一章介绍一种理想计算机M，它有一个存贮器，具有无穷多个的存贮单元$R_1$，$R_2$，…，每一个存贮单元可存一个任意大小的自然数，各个单元中存放的数的序列$r_1$，$r_2$，…，称为内部状态；M有四种算术指令：零指令，后继指令，传送指令和条件转移指令；由有穷多条算术指令组成的序列称为程序。程序$P$执行前的内部状态的初始状态。程序$P$执行完成以后，自然认为它完成了一项计算，其结果放在$R_1$中。允许程序执行后无结果，也允许程序执行过程永不停止。于是，一程序$P$，初始状态为$r_1$，…，$r_n$，计算结果为$r$，则称$P$收敛于$r$，记为$P(r_1, \cdots, r_n)\downarrow r$。如果有一函数$f(x_1, \cdots, x_n)$程序$P$，并且$f(a_1, \cdots, a_n)=b \Longleftrightarrow P(a_1, \cdots, a_n)\downarrow b$，则称$f(x_1, \cdots, x_n)$是可计算函数。这是检查函数可计算的基本方法。为有系统地研究可计算函数，单纯依靠这种方法是不够的，于是我们转入讨论可计算函数类。方法是：证明存在计算一组基本函数的程序，得出基本函数是可计算的，然后证明从基本函数出发，用三种方法（代入、原始递归和最小

数算子）构造出来的新函数还是可计算的。这样，我们就有了一个研究可计算函数性质的系统的方法，而不是杂乱无章了。但是，除了基本函数和从基本函数出发运用代入、原始递归和最小数算子构造出来的函数是可计算的以外，还有没有其他可计算函数了呢？这个问题的完满解决则是通过第二到第四章进行。

第二章讲原始递归函数，即从基本函数出发，只运用代入和原始递归两种运算所形成的函数类。显然，原始递归函数是可计算的。所以，虽然改换了名称，其实质还是在研究可计算函数。因为这部分内容特别重要，所以着重研究它。首先介绍一类常用的原始递归函数，接着讨论原始递归谓词。原始递归函数和原始递归谓词有密切的联系，谓词的特征函数是原始递归的，则它就是原始递归的。本书较详细地介绍了串值递归和无嵌套的二重递归，这两种递归式可以归结为原始递归和代入，它们在以后各章中经常被应用到。哥德尔$\beta$函数和它的另一个重要性质，是第七、八两章所必须用到的。配对函数在以后各章中经常被应用到。所以，这两章应作为重点来学习。

第三章着重说明原始递归函数类还可以进一步扩大。可以构造一般递归函数类和部分递归函数类。一个全函数$f(x_1, \cdots, x_n, y)$，经过最小数算子（$\mu$-算子）而得新函数$g(x_1, \cdots, x_n) = \mu y(f(x_1, \cdots, x_n, y) = 0)$，还是全函数，则称$f(x_1, \cdots, x_n, y)$为正则全函数，这时的$\mu$-算子不妨称为正则$\mu$-算子，从基本函数出发，通过代入。原始递归和正则$\mu$-算子而构造出来的函数类，称为一般递归函数类，而由基本函数出发，经过代入、原始递归和$\mu$-算子所形成的函数类，称为部分递归函数类。显然，一般递归函数类是部分递归函数类的真子类。为证明原始递归函数类是一般递归函数类的真子类，书中讨论了阿克曼函数的性质，证明它不是原始递归而是一般递归的，这样就构造了三个函数类，它们都是可计算函数类的子类，以部分函数类的最大。

第四章研究哥德尔编码技术，通过它证明可计算函数都是部

分递归的。从而证明了可计算函数类与部分递归函数类是相等的。

第三章第五节等式系，是用形式系统中有穷多个等式以及代换、替换运算来定义函数，并证明了部分递归函数可用等式系来定义，第四章第五节则用哥德尔编码方法，将形式系统算术化，证明等式系定义的函数都是部分递归的。从而，我们就给出了可计算函数类的三种不同的定义方法，从不同的角度描述了同一个函数类的性质，这对加深理解可计算函数的性质是非常必要的。

第四章第四节，我们突出了范式定理和 $s$-$m$-$n$定理，它们是可计算函数理论的支柱，在以后许多地方经常应用到，必须十分重视。

第四章第五节车赤论题，必须很好理解，这里讲“直观可计算函数”与用严格数学方法定义的可计算函数的关系问题。所谓函数$f(x)$是直观可计算的，就是存在一个能行的机械的方法，用它可以求出$f(x)$的值来。例如，函数$f(x, y)$定义为：如果$x$，$y$互质，则$f(x, y)=1$；如果$x$，$y$有公因子，则$f(x, y)=0$。显然，用辗转相除法可以求出$f(x, y)$的值。就说$f(x, y)$是直观可计算函数。因为第一章里定义的算术指令以及程序的执行过程是能行的、机械的，所以，可计算函数都是直观可计算的。反之，直观可计算函数都是可计算的吗？因为“直观可计算函数”这一概念不能用数学方法严格定义，它有些含糊其辞。所以“直观可计算函数都是可计算的”（$*$）这一命题是不能证明的。车赤把（$*$）作为一个假设提出来，通常称为车赤论题。车赤论题是讲“直观可计算函数”与“可计算函数”之间的关系问题。不要把它与可计算函数类同部分递归函数类（或同等式系定义的$\varepsilon$-递归函数类）相等的关系混为一谈。

第五章是把可计算函数理论应用到“判定问题”上来。一语句类$K$，如果存在一个能行方法，用它可决定任一给定的语句$P$是不是属于$K$，这个方法称为类$K$的判定法。类$K$的判定问题就

是给出它的判定法（这时类$K$可判定），或者证明类$K$的判定法不存在（这时类$K$不可判定）。因为可数集合与自然数集合可以建立对应关系，所以，可数集合上的判定问题常常转化为自然数集合上的判定问题。因此，可以就自然数集合上的语句类讨论判定问题，又因为一语句类$K$，可用谓词来描述，所以可用谓词讨论判定问题。根据车赤论题，谓词是直观可计算的主要条件，它是一般递归的，于是有定义：如果谓词$p(x)$是一般递归的，则$p(x)$的判定问题是递归可解的，否则，它的判定问题是递归不可解的。在这一章中，只限于讨论几个递归不可解的例子，因为它们与计算机理论以及可计算函数理论本身有密切联系。可解的判定问题有专门论著，限于篇幅，本书不加讨论。

第六章讨论一般递归集合和递归可枚举集合。函数、谓词、集合这三者有密切的联系，在前五章中，集中讨论函数和谓词，而把集合列专章讨论，这是为了避免把函数、谓词和集合平行讨论引起混乱。递归变换群一节，是从函数集合的角度看递归函数的一个性质，把递归性质看成是递归变换群的不变性，可加深对它的认识。更深入系统的研究必须参阅有关文献。

第七章介绍丢番图方程在整数集合中是否有解的判定问题是递归不可解的，其证明方法归结为证明谓词是丢番图的主要条件，它是递归可枚举的。这是第五章介绍的不可解的判定问题理论的具体应用，在前六章的基础上，阅读它不会产生困难。

最后一章讨论相关递归性和克林尼分层。相关递归性是递归概念的推广，从理想机的结构上看，有一集合$A$，称它为外部信息源，它向理想机输送信息而导入相关于$A$的可计算概念。而第一章讨论的理想机无外部信息源，所以，可计算概念是$A$-可计算的特例（外部信息源是空集合），从递归概念来看，函数$f$是$A$-原始递归的，实质就是：如果集合$A$是原始递归的，则$f$是原始递归的；如果集合$A$不是原始递归的，则$f$自然也不是原始递归的，这就是说$f$的原始递归性归结为集合$A$的原始递归性。同

样，对$A$--一般递归、$A$-部分递归也相应地这样来理解。有了相关递归的概念，就可以讨论归约，把$A$的递归性归结为讨论$B$的递归性。从而可以讨论非递归谓词之间的互相递归关系，把这些谓词分出层次来，这就是克林尼分层理论的基本点。算术分层的进一步发展，还可研究解析分层，这些内容都是递归函数理论中较新的课题。它超出了本书的范围。

本书讲解力求详尽，并举了许多例子帮助读者理解重要概念；又考虑到读者查阅资料的困难，书中应用到的初等数论、数学分析等学科中的基本知识，也以引理或附注形式放在正文中。为加深读者对函数类的认识，在第一章中特地介绍了归纳集合和最小归纳集合五种等价定义形式，为施归于函数结构的证法提供理论基础。

本书在编写过程中，得到张锦文同志许多具体帮助，在此表示深切致谢，由于作者水平所限，书中不妥乃至错误的地方难免，期望读者指正。

作　者

1983年9月

# 第一章 可计算函数

在这一章中，我们介绍一种理想计算机M，通过它定义可计算函数的概念，并且研究从已知函数生成新函数的三种方法：代入、原始递归和$\mu$-算子。最后讨论归纳集合，证明从基本函数出发，用生成新函数的方法可形成一个最小归纳集合。这个集合对代入、原始递归和$\mu$-算子是封闭的。

## §1 理想计算机

设有一架理想计算机M，它有一个存贮器，具有无穷多个存贮单元，现在分别地记它们为$R_1$，$R_2$，…，每一个存贮单元$R_j$($j=1$，2，…)可以存贮一个任意大小的自然数$r_j$（如图1·1所示），$r_j$称为$R_j$的内容。全部存贮单元的内容组成M的一个内部状态亦称格局。自然，内部状态随一些存贮单元的内容改变而改变。因为在讨论具体函数的计算问题时，用到的存贮单元总是有限的，其余存贮单元的内容都是0，因此，有时就把存贮器前有穷多个存贮单元的内容作内部状态。

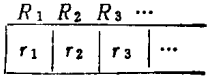


图 1·1

理想机M所能完成的基本运算称为指令，它有四种指令。

（1）零指令$Z(n)$。对每一个$n$（$n=1$，2，…），存在零指令$Z(n)$，它把存贮单元$R_n$的内容$r_n$清除掉，而其他单元的内部在执行$Z(n)$时不变，今以$r_n:=0$表示$Z(n)$的功能。

（2）后继指令$s(n)$。对每一个$n$（$n=1$，2，…），存在

后继指令$s(n)$，它把存贮单元$R_n$的内容$r_n$加1后仍存入$R_n$，而其他单元的内容在执行$s(n)$时不变，今以$r_n:=r_n+1$表示$s(n)$的功能。

（3）传送指令$T(m, n)$。对每一数对$\langle m, n\rangle$（$m, n=1, 2, \cdots$），存在指令$T(m, n)$，它把$R_m$的内容$r_m$传送到$R_n$中去，并把$R_n$中原有的内容冲掉，而$R_M$以及其他单元的内容在执行$T(m, n)$时不变。现在以$r_n:=r_m$表示$T(m, n)$的功能。

为了更明确地说明第四种指令，我们先来定义程序的概念。

**定义1·1** 有穷条指令的序列$P=I_1I_2\cdots I_n$，称为程序。程序中指令条数$n$称为它的长度。指令的编号称为它的标号。

（4）条件转移指令$J(m, n, q)$。假定有一程序$P=I_1I_2\cdots I_k$，其中$I_i$（$1\leqslant I_i\leqslant k$）是下列形式的新指令：对每一个三元数组$\langle m, n, q\rangle$（$m, n, q=1, 2, \cdots$），存在指令$J(m, n, q)$。

如果$r_m=r_n$，则机器M执行$P$中标号为$q$的指令；

如果$r_m\neq r_n$，则机器M执行$P$中指令$I_{i+1}$。

如果$q>k$，则机器M在条件$r_m=r_n$时停止运算。执行$J(m, n, q)$时，内部状态不改变。

指令$Z$、$S$、$T$、$J$总称为算术指令，它们在框图中的图示如图1·2所列。

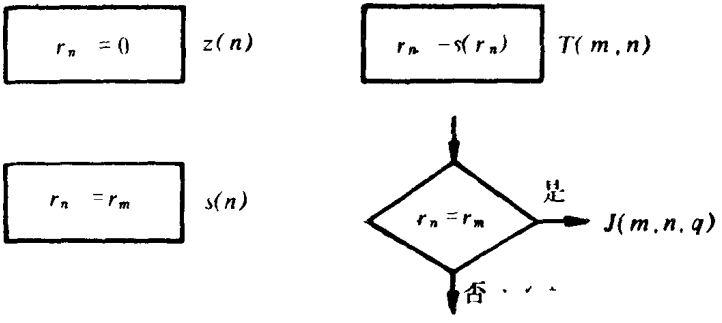


图 1·2

计算机M执行一条指令完成两项工作：

(i)改变存贮器中某单元的内容；

(ii)指出下一步执行那一条指令。

例如，若程序中指令$I_i$是$z(n)$，则$I_i$改变$R_n$的内容$r_n$为0，指出下一步执行$I_{i+1}$，又若$I_i$为$J(m, n, q)$，则它不改变内部状态（是改变的特例），指出下一步执行标号为$q$的指令（在$r_m=r_n$时），或执行指令$I_{i+1}$（在$r_m\neq r_n$时）。

根据指令的规定，执行指令所需条件是由内部状态提供的。给定程序$P=I_1I_2\cdots I_k$，它的长度为$k$，$n$个初始数据$a_1$，$a_2$，…，$a_n$分别存放在$R_1$，$R_2$，…，$R_n$中，这时的内部状态称为初始内部状态，程序$P$从$I_1$开始执行，$I_1$所需条件由初始内部状态提供。一指令完成时的内部状态称为瞬时内部状态。它为下一指令提供执行条件。因此，瞬时内部状态和下一指令的标号构成机器$M$的一个瞬时状态，它表示机器的一步运算。第$n$条指令完成后，如果不转移到执行前面的指令，则自动停机，若有一指令$I_i(1\leqslant i\leqslant k)$为$J(m, n, q)$，其中$q>k$，则当执行到$I_i$且$r_m=r_n$时，机器M也自动停机。计算结束后，结果存放在$R_1$中，其他单元的内容如何就可以不必关心了。

$P(a_1, \cdots, a_n)$表示机器M通过程序$P$执行有初始内部状态为$a_1$，…，$a_n$的计算，$P(a_1, \cdots, a_n)\downarrow$表示计算有结果，$P(a_1, \cdots, a_n)\uparrow$表示计算无结果，有结果时又称计算是收敛的，无结果时又称计算是发散的，计算收敛于$b$又记为$P(a_1, \cdots, a_n)\downarrow b$。

【例1·1】程序$P=Z(1)S(1)J(1,1,1)$，试证明对任何$a$，$P(a)\uparrow$。

证明　初始内部状态为$a$，$Z(1)$实现$r_1:=0$，$S(1)$实现$r_1:=0+1$，通过$J(1, 1, 1)$，计算又转到$Z(1)$，于是开始循环往复，所以$P(a)$是发散的。其框图如图1·3所示。

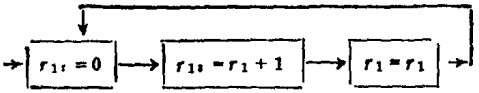


图 1·3

## §2 可计算函数的概念

如果有一程序$P$，初始内部状态为$a_1$，…，$a_n$，$P$（$a_1$，…，$a_n$）$\downarrow b$，则$a_1$，…，$a_n$与$b$之间的对应是完全确定的，可见$P$确定一函数，记它为$f_P$，其关系为：

$$f_P(x_1, \cdots, x_n) = \begin{cases} y & P(x_1, \cdots, x_n) \downarrow y \\ \text{不定义} & \text{否则} \end{cases}$$

**定义1·2** $f$是$N^*$到$N$的函数，$P$是$M$的程序，如果对每一个$n$元数组$\langle a_1, \cdots, a_n\rangle \in \mathrm{dom} f$，存在数$b$，使得$P$（$a_1$，…，$a_n$）$\downarrow b \Longleftrightarrow f(a_1, \cdots, a_n) = b$，则称$M$通过$P$能计算函数$f$。

**定义1·3** 如果有一程序$P$，$M$通过$P$能计算函数$f$，则称$f$是可计算函数。

定义1·3的意义就是：存在$P$，使得$f = f_P$。

【例1·2】已知初始数据为10、5，程序$P$为

$I_1$ $J$（3，2，5）

$I_2$ $S$（1）

$I_3$ $S$（3）

$I_4$ $J$（1，1，1）

试问$f_P(10, 5) = ?$ $f_P(x, y)$是什么函数？

解 程序$P$的框图如图1·4所示。初始内部状态为10、5，$I_1$为条件转移指令，判别条件为$r_2$是否等于$r_3$。如果$r_2 \neq r_3$，则转入$I_2$，将$r_1$加1仍放入$R_1$，然后执行$I_3$将$r_3$加1仍放入$R_3$，$I_4$是利用条件转移指令实现无条件转移，因为$r_1 = r_1$是始终成立的，所以，$I_4$总是把计算转移到$I_1$，以后就实现循环运算，直到$r_3 =$

5，这时$r_2=r_3$，于是$I_1$转移到执行标号为5的指令，而程序长度为4，所以，根据$J$指令功能的规定，机器$M$自动停机。由于$I_2$、$I_3$实现对$r_1$及$r_3$加1，并且把结果仍分别放入$R_1$、$R_3$，可见$R_3$起计数器作用，它计算$r_1:=r_1+1$运算的次数，当$r_2=r_3$时，表明已作了5次运算。因此，机器停止以后，$r_1=10+5=15$。所以$f_\mu(10,\ 5)=15$，如果初始数据为$x$、$y$，则$f_\mu(x,\ y)=x+y$。

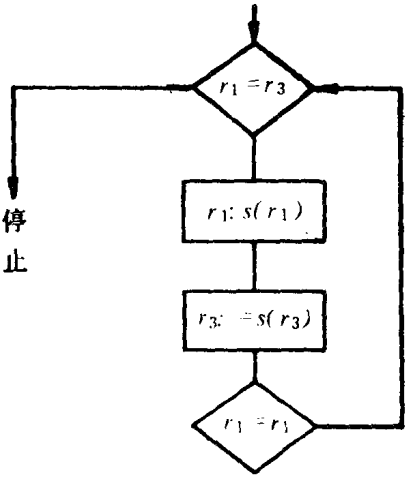


图 1·4

【例1·3】试编写计算$f(x,\ y)=x\cdot y$的程序。

解　初始内部状态为$x$、$y$，以$R_3$做工作单元，存放和数(计算过程中存放部分和)，以$R_4$、$R_5$做计数器，分别计$x$、$y$，程序一开始应该判别$x$是不是0，因为$x$存放在$R_1$中，$R_4$为$x$计数器，未计数时$r_4=0$，所以，判别$x$是否为0，可由$r_1$是否等于$r_4$来实现，如果$r_1=r_4$，则只要把$R_3$中的0传送到$R_1$而停止运算，如果$r_1\neq r_4$（即$x\neq 0$），则再判别$y$是否为0，因为$y$存放在$R_2$中，$R_5$为$y$计数器，未计数时为0，所以，用$r_2$是否等于$r_5$作

判别条件。如果$r_2=r_5$，则把$R_3$中的0传送到$R_1$而停止运算。如果$x\neq 0$、$y\neq 0$，则进行一般情形的乘法计算。实现的基本思想是：逐次作$r_3:=r_3+1$，每做一次，$x$计数器$R_4$中加1，当$r_1=r_4$时，表示已经把$r_3$增加到$r_3+x$了，于是可把$R_4$的内容清除，将$y$计数器加1，表示$r_3$已经增加了一个$x$。然后再逐次作$r_3:=r_3+1$，$x$计数器加1，…，直到$y$计数器的内容$r_4=y$（即$r_2=r_5$），表示已在$R_3$中加了$y$次的$x$，即实现了$r_3=x\cdot y$为止。其逻辑框图如图1·5所示，程序为：

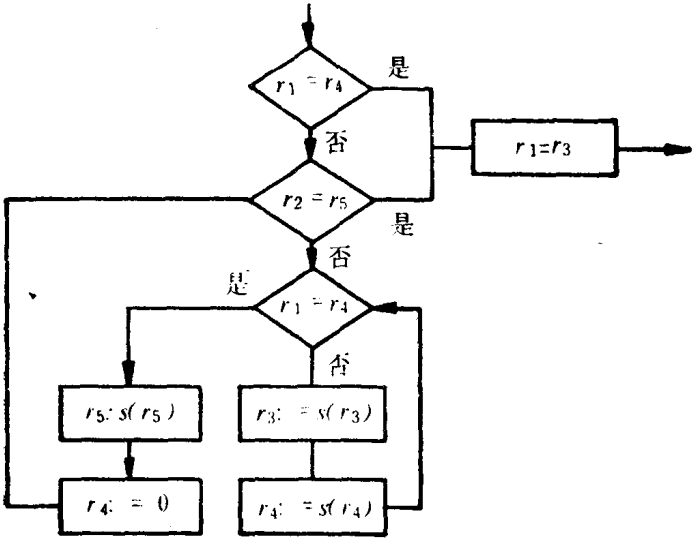


图 1·5

1.$J$（1，4，10） 2.$J$（2，5，10） 3.$J$（1，4，7）
4.$S$（3） 5.$S$（4） 6.$J$（1，1，3） 7.$S$（5）
8.$Z$（4） 9.$J$（1，1，2） 10.$T$（3，1）

【例1·4】函数$f(x)=\begin{cases}\frac{1}{2}x & \text{如果}x\text{是偶数}\\ \text{不定义} & \text{否则}\end{cases}$，试问$f(x)$

是否可计算？

解　函数$f(x)$的定义域是偶数集，因此令$R_2$、$R_3$作$2K$、$K$的计数器，初始数据$x$在$R_1$中，程序开始时，先判别$x$是否为偶数$2K$，即比较$r_1$与$r_2$是否相等。如果$r_1=r_2$，则把$R_3$中的$r_3$传送到$R_1$；如果$r_1\neq r_2$，则把$R_2$中加2，$R_3$中加1，进到下一个自变元$x+1$的计算。其逻辑框图如图1·6所示，程序为：

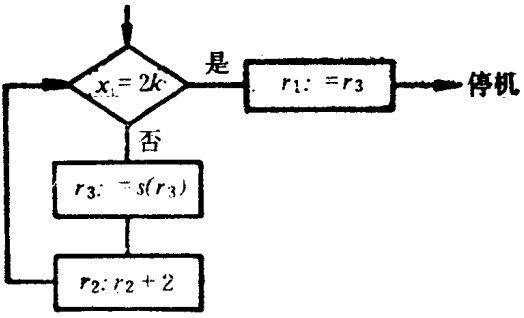


图 1·6

1.$J$（1，2，6）
2.$S$（3）
3.$S$（2）
4.$S$（2）
5.$T$（1，1，1）
6.$T$（3，1）

因为我们找到计算$f(x)$的程序，所以，它是可计算函数。

**注记1·1**　把例1·4的程序$P$中指令$I_6$修改为$T$（3，4），再增加指令$T$（4，1）而得程序$P'$，显然，$P$、$P'$计算同一个函数，以上所述这种修改程序的方法，可以连续使用多次，从而可以产生许多不同的程序，由此可见，一个函数可以有无穷多个程序来计算它。如果程序$P$、$P'$计算同一个函数，则$f_P=f_{P'}$。

**定义1·4**　如果对任一$x$，$O(x)=0$，则称$O(x)$为零函数；如果对任一$x$，$s(x)=x+1$，则称$s(x)$为后继函数；如果对任

意 $n$，$1 \leqslant i \leqslant n$，$P_i^n(x_1, \cdots, x_i, \cdots, x_n) = x_i$，则称$P_i^n(x_1, \cdots, x_i, \cdots, x_n)$为投影函数。$O(x)$、$S(x)$、$P_i^n(x_1, \cdots, x_i, \cdots, x_n)$一起称为基本函数。

**注记1·2** 设$n$维空间的坐标轴分别为$x_1, x_2, \cdots, x_n$，如果把$N^n$看成为$n$维空间内的格子点，则$P_i^n(x_1, \cdots, x_i, \cdots, x_n)$的意义是把格子点向第$i$条坐标轴$x_i$投影。所以称这种函数为投影函数。图1·7中是2维空间中的格子点（$N^2$内的点）向$x$轴投影的图形。

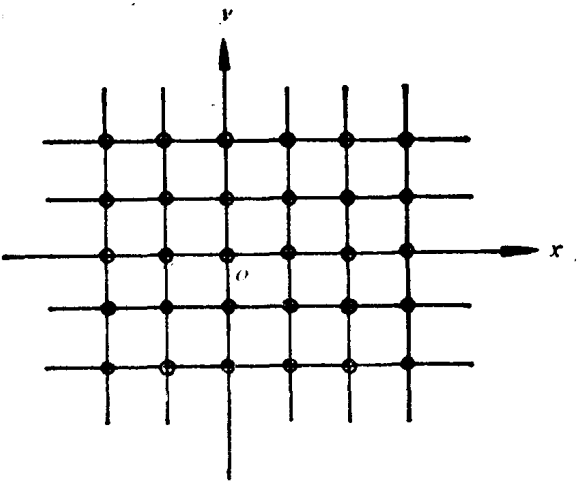


图 1·7

**定理1·1** 基本函数是可计算函数。

**证明** 程序$P_1 = Z(1)$，$P_2 = S(1)$，$P_3 = T(i, 1)$分别计算$O(x)$，$S(x)$，$P_i^n(x_1, \cdots, x_i, \cdots, x_n)$。根据定义1·3可知基本函数是可计算函数。

如果编出计算已知函数的程序，则说明已知函数是可计算的。这是证明函数可计算的基本方法。但是单靠这种方法是不够的。因此下面讨论从已知函数构造新函数的三种方法。并且证明，如果已知函数是可计算的，则通过这三种方法构造出来的函