迭代函数及其可计算性*

阎志欣 黄盛萍

(北京航空航天大学计算机科学与工程系 北京 100083) (南京大学计算机软件研究所 南京 210093)

摘要 带迭代算子的函数式程序设计语言是一种有坚实理论基础、高效的、实际有用的高级程序设计语言.该语言既可描述递归,又可描述迭代.本文证明了仅用迭代算子由已知迭代函数定义的部分迭代函数就可计算部分递归函数.这就等于证明任何图灵可计算的部分函数,都可用有坚实数学基础的部分迭代函数高效地进行计算.文中定义了复合算子、原始迭代和最小化迭代算子及部分迭代函数;证明了部分递归函数是部分迭代函数可计算的.

关键词 函数式语言,原始迭代算子,最小化迭代算子,部分迭代函数,计算能力.

程序设计语言的目的是对待求解问题的算法实现人—机通讯.用传统的指令式语言虽然可描述迭代,使得程序有高的时空效率,但程序缺乏数学特性,从而导致了不易理解、分析和证明等许多问题.[1]陈述式语言把算法的逻辑和控制分离,程序仅描述算法的输入输出关系,使得程序有良好的数学特性,易于理解、分析和证明.但由于其程序仅包含输入输出变量,难以描述迭代,从而难以构造出高效的执行系统.[2]近20年来各种陈述式语言被广泛研究[3~5],但由于程序的控制结构仍然是递归,因而时空效率问题一直是研究的重要问题.

为使程序设计语言既有良好的数学特性,又易于构造高效的执行系统,使程序有高的执行效率,文献[6]提出了带迭代算子的函数式计算模型,给出了基于该模型的程序设计语言的语法及操作语义.该类语言是一种有良好数学基础,既可描述递归又可描述迭代的高效的、实际有用的程序设计语言.任何一种程序设计语言,只有在其计算能力和图灵机等价时才可接受.[7]我们知道,递归函数的计算能力等价于图灵机已被证明,因此该类语言的计算能力是可保证的.但递归的时空效率很低,特别是空间效率.基于迭代的while程序是一种高效的指令式语言,它可计算部分递归函数已被证明.[7]然而文献[6]提出的语言,其迭代计算的能力尚未证明.本文证明了仅用迭代算子由已知函数定义的部分迭代函数可计算部分递归函数.这就等于证明任何图灵可计算的部分函数,都可用有坚实数学基础的迭代函数高效地计算它.

本文基于文献[6]中提出的带迭代算子的函数式计算模型,定义了复合算子、原始迭代和最小化迭代算子及由它们和几个初始函数产生的部分迭代函数;最后证明了部分递归

* 本文研究得到南京大学计算机软件研究所基金和航空科学技术基金资助.作者阎志欣,1936年生,教授,主要研究领域为计算,推理理论和程序设计语言.黄盛萍,女,1965年生,工程师,主要研究领域为计算理论和程序设计语言.

本文通讯联系人:阎志欣,北京 100083,北京航空航天大学计算机科学与工程系

本文1995—06—19收到修改稿

函数是部分迭代函数可计算的.

1　迭代函数

文献[6]描述了带迭代算子的函数式程序设计语言的语法.为精炼本文和方便讨论,下面仅列出对本文的讨论有关键作用的迭代表达式的定义.文中涉及到的其它概念,如项和谓词等,请参考文献[6].

定义 1. 让 $h:D^m\to D$ 是 m 元初始或定义函数,又称主迭代函数,$(t_1,\cdots,t_m)$ 是项表;$w:N^2\to\{k\mid 1\leqslant k\leqslant m\}$ 是自然数集上的 2 元初始函数,又称位置函数;P 是一个谓词;e,n 是自然数;则一个迭代表达式是一个形式如下的表达式:

$$\underset{i=e}{P\backslash w(i,n)\backslash h(t_1,\cdots,t_m)}$$

其中 i 是 P 和 w 的一个变量,又称迭代变量,并且允许出现在 $t_1,\cdots,t_m$ 中.$i=e$ 表示 i 的初值是 e,每步计算后 i 的值自动加 1;若以 I^d 表示该迭代表达式执行了从 $i=e$ 至 $i=d$ 若干次迭代后的阶段性计算结果,则该表达式应满足:

(1)若在 $i=e$ 下有 $w(e,n)=a$,则迭代表达式有初始结果:$I^e=t_a$.

(2)若在 $i=d$ 下 P 为假并且 $w(d,n)=a,w(d+1,n)=b$,则迭代表达式在 $i=d+1$ 时有结果:$I^{d+1}=h(t_1,\cdots,t_{a-1},I^d,t_{a+1},\cdots t_m)[i\leftarrow d+1]$ 并且 I^{d+1} 被存储在项表 $(t_1,\cdots,t_m)$ 的位置 b 处.

(3)如果存在一个 c 使得 P 在 $i=c$ 下为真,则对所有 $j,j\geqslant c$,迭代表达式有终止结果:$I^j=I^c$.

(4)若不存在任何 c 使得 P 在 $i=c$ 下为真,则迭代表达式无定义,即计算不终止.

其中 $X[y\leftarrow z]$ 表示若 y 在 X 中出现,则 X 中的 y 用 z 置换所得到的结果;若 y 不在 X 中出现,则置换为空,X 不变.迭代表达式中的所有置换是同时的.位置函数 w 是使得 $w(x,y)=z$ 并且 $1\leqslant z\leqslant m$ 成立的任何初始函数,用于指示迭代表达式中间计算结果的存放位置.特别有用的是自然数集上的常函数和 x 除以 y 的余数加 1 函数.上面 4 个条件由带迭代算子的函数式语言的解释器保证.[6]

带迭代算子的函数式计算模型计算的是任意域上的函数.为了研究其计算能力,下面我们仅考虑自然数集上的数论函数;首先给出 5 个初始函数和一个初始谓词;然后给出复合算子、原始迭代和最小化迭代算子的定义;最后给出原始、一般和部分迭代函数的定义.

定义 2. 下面自然数集 N 上的函数是迭代函数的初始函数:

(1)常函数 $C_k=k$,其中 $k\in N$;

(2)后继函数 $succ(x)=x+1$,其中 $x\in N$;

(3)先驱函数 $pred(x)=x-1$,其中 $x\in N$,若 $x=0$,则 $pred(0)=0$;

(4)投影函数 $u_i^n(x_1,\cdots,x_n)=x_i$,其中对所有 i 和 n,$1\leqslant i\leqslant n,x_i\in N$;

(5)模加 1 函数 $g(x,y)=z+1$,其中 $z=x-y*w$ 并且 $z\leqslant y$,并且 $x,y,z,w\in N$.

函数 g 即 x 除以 y 的余数加 1.它可由其它初始函数和初始谓词通过应用复合和原始

迭代算子产生，但 g 作为初始函数不失一般性，因增加 g 后初始函数的个数仍然是有限的. g 用以控制迭代表达式中间计算结果的存放位置，作为初始函数可有效地增加语言的表达能力和简化解释系统.

定义 3. 让符号"="表示比较 2 个自然数的相等关系，则$(x=y)$是初始谓词，其中 $x,y\in N$.

下面，我们把以函数作为变量产生新函数的运算称为算子.

定义 4. 假设 $h:N^k\to N$ 是 k 元函数，对每个 $j,1\leqslant j\leqslant k$，$g_i:N^n\to N$ 是 n 元函数. 让 C 是复合算子，则 C 被定义为：

$$C:C(h,g_1,\cdots,g_k)(x_1,\cdots,x_n)=h(g_1(x_1,\cdots,x_n),\cdots,g_k(x_1,\cdots,x_n))$$

若 $h,g_1,\cdots,g_k$ 是给定的函数，则应用复合算子定义的 n 元函数 f 被表示为：

$$f(x_1,\cdots,x_n)=h(g_1(x_1,\cdots,x_n),\cdots,g_k(x_1,\cdots,x_n))$$

并且函数 f 被称为应用复合运算由给定函数 $h,g_1,\cdots,g_k$ 得到的函数.

定义 5. 让 $h:N^{n+1}\to N$ 是 $n+1$ 元函数，$g_1:N^{n-1}\to N$ 是 $n-1$ 元函数，函数 $w(i,n)=g(i,n)$，或 $w(i,n)=C_k$，I 是原始迭代算子，则 I 被定义为：

$$I:I(h,g_1,w)(x_1,\cdots,x_n)=\backslash_{i=0}^{(i=x_1)}w(i,n)\backslash h(g_1(x_2,\cdots,x_n),pred(i),x_2,\cdots,x_n)$$

若 h,g_1,w 是给定的函数，则应用原始迭代算子定义的 n 元函数 f 被表示为：

$$f(x_1,\cdots,x_n)=\backslash_{i=0}^{(i=x_1)}w(i,n)\backslash h(g_1(x_2,\cdots,x_n),pred(i),x_2,\cdots,x_n)$$

并且函数 f 被称为应用原始迭代运算由给定函数 h,g_1,w 得到的函数.

为了引入最小化迭代算子，首先引入正则函数的概念.

定义 6. 让 $q:N^{n+1}\to N$ 是一个 $n+1$ 元全函数. 若对所有 n 元组 $(x_1,\cdots,x_n)\in N^n$，至少存在一个 $y\in N$，使得 $q(y,x_1,\cdots,x_n)=1$，则 q 被称为正则函数.

定义 7. 让 $q:N^{n+1}\to N$ 是一个 $n+1$ 元全函数，$X=(x_1,\cdots,x_n)\in N^n$，$I$ 是最小化迭代算子，则 I 被定义为：

$$I:I(q)(X)=\backslash_{i=0}^{(q(i,X)=1)}1\backslash succ(0)$$

若 q 是给定的函数，则应用最小化迭代算子由 q 定义的 n 元函数 f 被表示为：

$$f(x_1,\cdots,x_n)=\backslash_{i=0}^{(q(i,X)=1)}1\backslash succ(0)$$

并且函数 f 被称为应用最小化迭代运算由给定函数 q 得到函数.

定义 8. 一个函数被称为原始迭代函数当且仅当它可由初始函数和初始谓词出发，通过有限次应用复合和原始迭代运算而得到.

定义 9. 一个函数被称为一般迭代函数当且仅当它可由初始函数和初始谓词出发，通过

有限次应用复合、原始迭代和对正则函数应用最小化迭代运算而得到.

定义 10. 一个函数被称为部分迭代函数当且仅当它可由初始函数和初始谓词出发,通过有限次应用复合、原始迭代和对任何全函数应用最小化迭代运算而得到.

构造某个原始迭代函数,并不总是仅仅使用初始函数.例如,如果已经有了一组原始迭代函数 $f_1,f_2,\cdots,f_n$,那么只要限于运用复合和原始迭代运算,我们就可使用这些函数中的任何几个,与初始函数和初始谓词一起去得到其它的原始迭代函数.这个原则同样适用于一般迭代和部分迭代函数.显然,原始、一般和部分迭代函数的构造性定义是带迭代算子的函数式程序.

一般迭代函数,允许对正则函数应用最小化迭代运算,因而比原始迭代函数可产生一个较大的函数类.而部分迭代函数,允许对任何全函数应用最小化迭代运算,因而比一般迭代函数又可产生一个较大的函数类.显然,就计算能力来说,部分迭代函数大于一般迭代函数,而一般迭代函数大于原始迭代函数.自然,所有原始迭代和一般迭代函数都是全函数.

2　函数的迭代性

要知道一个由其它方式(如递归定义)给出的函数是一个迭代函数,就要给出其迭代定义.构造迭代定义时,先按复合、原始迭代和最小化迭代运算的结构给出这种定义的标准型,然后对其化简.

例:说明下面的加函数是原始迭代的:

$add(0,y)=y$;

$add(x+1,y)=succ(add(x,y))$

证明:因一个 n 元函数可看成一个 $n+m$ 元函数[7],则 1 元函数 $succ(x)$ 可看成 3 元函数.函数 add 可用初始函数 $succ,pred,u_3^3$ 及初始谓词 $=$,通过复合及原始迭代算子定义为如下的标准形:

$$add(x,y)=\mathop{\backslash 1}_{i=0}^{(i=x)}\backslash succ(u_1^3(u_1^1(y),pred(i),y))$$

这里,主迭代函数 $h(g_1(y),pred(i),y)=succ(u_1^3(u_1^1(y),pred(i),y))$,$g_1(y)=u_1^1(y)$. $w(i,n)=c_1=1$.化简标准形得:

$$add(x,y)=\mathop{\backslash 1}_{i=0}^{(i=x)}\backslash succ(y)$$

下面是一些函数的迭代定义(已经化简):

(1)乘函数是原始迭代的:

$mul(0,y)=0$;

$mul(x+1,y)=add(mul(x,y),y)$

$$mul(x,y)=\mathop{\backslash 1}_{i=0}^{(i=x)}\backslash add(0,y)$$

(2) 幂函数是原始迭代的:

$power(0,y)=1$;

$power(x+1,y)=mul(power(x,y),y)$

$$power(x,y)=\mathop{\backslash 1}_{i=0}^{(i=x)}\backslash mul(succ(0),y)$$

(3)阶乘函数是原始迭代的：

$fact(0)=1;$

$fact(x+1)=mul(fact(x),x+1)$

$$fact(x)=\underset{i=0}{\overset{(i=x)}{\backslash 1\backslash mul(succ(0),i)}}$$

(4)减函数是原始迭代的：

$sub(0,y)=y;$

$sub(x+1,y)=pred(sub(x,y))$

$$sub(x,y)=\underset{i=0}{\overset{(i=x)}{\backslash 1\backslash pred(y)}}$$

(5)符号函数是原始迭代的：

$sg(0)=0;$

$sg(x+1)=succ(0)$

$$sg(x)=\underset{i=0}{\overset{(i=1)}{\backslash 1\backslash succ(0)}}$$

(6)求 2 数相减的绝对值函数是原始迭代的：

$absub(x,y)=add(sub(x,y),sub(y,x))$ 同左

(7)*fibonacci* 函数是原始迭代的(串值迭代)：

$fib(0)=0;$

$fib(1)=1;$

$fib(x)=add(fib(x-2),fib(x-1))$

$$fib(x)=\underset{i=0}{\overset{(i=x)}{\backslash g(i,2)\backslash add(0,1)}}$$

3 可计算函数

为证明部分递归函数可由部分迭代函数计算，后面用到了结构归纳法.[8]下面定理给出了结构归纳的一般方法.

定理(Structural induction, Burstall). 让$(S,<)$是一个良序集，φ是 S 上的全谓词：如果$(\forall a\in S)\{[使得\ b<a\ 的(\forall b\in S)\varphi(b)]\supset(a)\}$，则$(\forall c\in S)\varphi(c)$.

自然数集就通常的<(小于)关系是良序集. 用结构归纳法证明断言的步骤是：首先对自然数集上的最小数 0，证明 $\varphi(0)$为真；然后假定对任意自然数 x，$\varphi(x)$为真，证明 $\varphi(x+1)$为真. 本节给出鉴别迭代函数计算能力的一个重要结果如下：

定理. 一个自然数集上的函数是部分递归的，则它是部分迭代函数可计算的.

证明：如果 $f:N^n\to N$ 是部分递归的，则我们要证明：f 是部分迭代函数. 为此，我们根据文献[7]中对部分递归函数的定义，归纳证明于函数的结构：

(1)所有初始递归函数，即零函数 $Z(x)\equiv 0$、后继函数 $Succ(x)=x+1$ 和投影函数 $U_i^n(x_1,\cdots,x_n)=x_i$，都是初始迭代函数，显然它们是部分迭代函数可计算的.

(2)让 $h,g_1,\cdots,g_k$ 是部分递归函数，其中 $h:N^k\to N$ 是 k 元函数，对所有的 i，$1\leqslant i\leqslant k$，$g_i:N^n\to N$ 是 n 元函数，通过复合运算得到的部分递归函数 f：

$$f(x_1,\cdots,x_n)=h(g_1(x_1,\cdots,x_n),\cdots,g_k(x_1,\cdots,x_n))$$

假设函数 $h,g_1,\cdots,g_k$ 是部分迭代函数，显然由 $h,g_1,\cdots,g_k$ 通过复合运算得到的部分迭代函数 f' 计算了函数 f：

$$f'(x_1,\cdots,x_n)=h(g_1(x_1,\cdots,x_n),\cdots,g_k(x_1,\cdots,x_n))$$

(3)由部分递归函数 h,g_1 通过原始递归运算得到的部分递归函数 f：

$f(0,x_2,\cdots,x_n)=g_1(x_2,\cdots,x_n)$;

$f(m+1,x_2,\cdots,x_n)=h(f(m,x_2,\cdots,x_n),m,x_2,\cdots,x_n)$

其中,$h:N^{n+1}\to N$ 是 $n+1$ 元函数,$g_1:N^{n-1}\to N$ 是 $n-1$ 元函数.假设函数 h,g_1 是部分迭代函数.我们要证明,下面的由部分迭代函数 h,g_1,w 和 $pred$ 及初始谓词=通过原始迭代运算得到的部分迭代函数 f' 计算了函数 f:

$$f'(x_1,\cdots,x_n)=\mathop{\backslash 1}_{i=0}^{(i=x_1)}\backslash h(g_1(x_2,\cdots,x_n),pred(i),x_2,\cdots,x_n)$$

让谓词 $\varphi(m)$是$\forall\ (m\in N)[f(m,x_2,\cdots,x_n)=f'(m,x_2,\cdots,x_n)]$,这是一个全谓词并且$(N,<)$是良序集[8],因此可用结构归纳法证明如下:

1)当 $m=0$ 时,证明 $\varphi(0)$为真.根据迭代表达式的定义 1 有:

$$f'(0,x_2,\cdots,x_n)=\mathop{\backslash 1}_{i=0}^{(i=0)}\backslash h(g_1(x_2,\cdots,x_n),pred(i),x_2,\cdots,x_n)=g_1(x_2,\cdots,x_n)$$

因 $f(0,x_2,\cdots,x_n)=g_1(x_2,\cdots,x_n)$,则 $f(0,x_2,\cdots,x_n)=f'(0,x_2,\cdots,x_n)$,即 $\varphi(0)$为真.

2)当 $m>0$ 时,假定 $\varphi(m)$为真,证明 $\varphi(m+1)$为真.根据迭代表达式定义 1 有:

$$\begin{aligned}f'(m+1,x_2,\cdots,x_n)&=I^{m+1}=h(I^m,pred(i),x_2,\cdots,x_n)[i\leftarrow m+1]\\&=h(f'(m,x_2,\cdots,x_n),pred(m+1),x_2,\cdots,x_n)\end{aligned}$$

因 $pred(m+1)=m$,并且根据归纳假设有 $f(m,x_2,\cdots,x_n)=f'(m,x_2,\cdots,x_n)$,则有:

$$f'(m+1,x_2,\cdots,x_n)=h(f(m,x_2,\cdots,x_n),m,x_2,\cdots,x_n)$$

因 $f(m+1,x_2,\cdots,x_n)=h(f(m,x_2,\cdots,x_n),m,x_2,\cdots,x_n)$,则有:

$$f(m+1,x_2,\cdots,x_n)=f'(m+1,x_2,\cdots,x_n)$$

因而 $\varphi(m+1)$为真.

(4)由任何 $n+1$ 元全函数 $q:N^{n+1}\to N$ 通过最小化运算得到的部分递归函数 f:

$$f(x_1,\cdots,x_n)=\mu y[q(y,x_1,\cdots,x_n)=1]$$

假设函数 q 是完全可计算的迭代函数.我们要证明,下面的由迭代函数 q 和初始函数 $succ$ 通过最小化迭代运算得到的部分迭代函数 f' 计算了函数 f:

$$f'(x_1,\cdots,x_n)=\mathop{\backslash 1}_{i=0}^{(q(i,X)=1)}\backslash succ(0)$$

证明分 2 种情况:

(1)若不存在自然数 m 使得$(q(m,X)=1)$为真,要证明 f 和 f'都无定义.显然 f 在 X 处无定义.根据迭代表达式定义 1,f'在 X 处也无定义,即不停机.

(2)若 m 是使得$(q(m,X)=1)$为真的最小自然数,要证明 $f(x_1,\cdots,x_n)=f'(x_1,\cdots,x_n)$.显然,当$(q(m,X)=1)$为真时,$f(x_1,\cdots,x_n)=m$.再根据迭代表达式的定义 1 有:

$$f'(x_1,\cdots,x_n)=I^m=succ(I^{m-1})$$

显然,迭代表达式:

$(q(i,X)=1)$
$\1\succ(0)$
$i=0$

对所有小于 m 并且使得$(q(d,X)=1)$为假的 d,$I^d=d$. 因 $m-1<m$ 并且 $q(m-1,X)=1$ 为假,则有 $I^{m-1}=m-1$. 所以,$f^l(x_1,\cdots,x_n)=succ(m-1)=m$.

参考文献

1 Backus John. Can programming be liberated from the Von Neumann style? A function style and its algebra of programs. Communications of the ACM, 1978,21(8):613～641.

2 Morris James H. Real programming in function languages. In: Darlington ed. Function Programming and Its Applications, 1982.

3 Guo Yike, Lok H C R. A classification scheme for declarative programming languages—sytax, semantics, and operational models. GMD—Studien, August 1990,182(8).

4 Darlington John, Guo Yike, Pull Helen. A design space for iterating declarative languages. In: Declarative Programming, Sasbachwalden 1991, 1992.

5 Darlington John, Guo Yike, Pull Helen. Introducing constraint functional logic programming. In: Declarative programming, Sasbachwalden 1991, 1992.

6 阎志欣. 带迭代算子的函数式程序设计. 软件学报,1996,7(增刊):239～248.

7 Kfoury A J, Moll Robert N, Arbib Michael A. A programming approach to computability. New York: Springer—Verlag, 1982.

8 Manna Z. Mathematical theory of computation. New York: McGRAW—HILL Inc., 1974.

ITERATION FUNCTIONS AND THEIR COMPUTABILITY

Yan Zhixin

(*Department of Computer Science Beijing University of Aeronautics and Astronautics Beijing* 100083)

Huang Shengping

(*Institute of Computer Software Nanjing University Nanjing* 210093)

Abstract Function programming language with iteration operator is an efficient, useful and practical, high—level programming language with sound theoretical foundation. It can represent recursion and iteration. This paper proved that partial iteration functions which are defined by iteration operators from given iteration functions can compute partial recursion functions. It means that any turing computable partial functions can be computed efficiently by iteration functions with sound theoretical foundation. In this paper, the authors defined the composition operator, primitive iteration operator, minimization iteration operator and partial iteration functions. Then they proved that partial recursion functions can be computed by partial iteration functions.

Key words Function language, primitive iteration operator, minimization iteration operator, partial iteration function, computability.