文章编号:1000-5862(2020)03-0307-06

2 类数列问题循环不变式开发策略研究与应用

古素梅,杨庆红*
(江西师范大学计算机信息工程学院,江西 南昌 330022)

摘要:该文通过对组合数学中 Catalan 数列问题和 Fibonacci 数列问题进行深入研究,利用归纳推理、组合数学中的加法和乘法原理等方法得到问题求解函数,使用变量记录算法求解过程中子问题的解,并约束循环变量的变化范围,获得问题求解算法的循环不变式,由此得到了 2 类数列问题循环不变式的统一开发策略. 以二叉树的形态数问题和阶梯问题为例,利用所提策略开发循环不变式,并基于循环不变式展示了这 2 类数列问题算法程序的形式化推导过程.

关键词:数列问题;循环不变式;可信软件;归纳推理
中图分类号:TP 311 文献标志码:A DOI:10.16357/j.cnki.issn1000-5862.2020.03.15

0 引言

随着计算机应用的日益普及,计算机软件的正确性和可靠性在各个领域中都受到高度重视,尤其在一些关键领域(如市场经济、交通安全、航空航天等领域)中更是至关重要[1]. 算法是软件的核心,它在软件中占有不容忽视的地位和作用. 而形式化方法是保证算法正确性和可靠性的有效途径之一[1-2].

大多数复杂算法需要使用循环结构加以实现,而循环不变式在循环程序设计中占有至关重要的地位,它是理解、证明和开发一个算法程序的关键[3-4]. 循环不变式的开发一直是形式化领域中最具挑战性、最富创造性、最困难的问题之一,寻找循环不变式开发策略一直存在较多难点. 许多学者和形式化领域专家一直致力于循环不变式开发策略的研究,分别从不同角度提出了众多循环不变式开发技术和策略[5-6].

E. W. Dijkstra 提出构造循环不变式开发策略,然后 D. Gries[7] 对该策略进行了补充和解释,并称其为标准策略. 该策略实质上是利用气球原理,通过弱化后置断言,得到循环不变式. 标准策略及对标准策略的扩充方法能得到的循环不变式在形式上都必须与后置断言非常相似,而实际上,许多循环程序的循环不变式与其后置断言却并不形似[8-9]. 此外该策略针对简单问题比较有效,对一些具有复杂数据结构的算法程序或精巧算法程序循环不变式的开发存在较多困难[10-12]. M. Janota[13] 介绍了一种基于断言的循环不变式自动探测技术. 该技术从程序规约和静态分析循环体 2 个方面来推导循环不变式. 但在该方法中循环不变式的探测能力与程序中间断言有密切关系,而程序中间断言的获取又是一个难点,因而该探测技术未能得到较好地推广和应用. 薛锦云[14] 通过对大量算法程序的研究分析,给出循环不变式的一个更加确切的定义:"给定循环语句 DO 和它的所有循环变量的集合 A,一个反映 A 的变化规律且在循环体 S 执行前后均为真的谓词,称为循环语句 DO 的循环不变式",同时给出循环不变式开发的新策略. 该策略对于循环不变式开发具有宏观的指导意义. 但是不同问题的数学性质和求解方法不一样,因此非常有必要对该策略做进一步深入的研究,将该策略进一步细化.

本文主要对组合数学问题[15] 中的 2 类数列问题进行深入研究,分析这 2 类问题的数学性质和求解特征,提出了 2 类数列问题循环不变式的统一开发策略. 该策略首先通过归纳推理方法以及组合数学中的加法原理和乘法原理等方法获得原问题和子问题之间应该满足的函数关系,得到问题求解函数;然后使用变量记录算法求解过程中子问题的解,并约束循环变量的变化范围,由此获得问题求解算法

收稿日期:2019-09-16
基金项目:国家自然科学基金(61662035)资助项目.
通信作者:杨庆红(1968-),女,江西南昌人,教授,主要从事软件形式化和智能教育软件的研究. E-mail:yangqh120@163.com

的循环不变式. 本文以二叉树的形态数问题和阶梯问题为例,利用所提出的策略开发循环不变式,并基于循环不变式展示了这 2 类数列问题算法程序的形式化推导过程.

1 2 类数列问题循环不变式开发策略

在组合数学中有 3 类典型数列问题,分别是 Stirling 数列、Catalan 数列和 Fibonacci 数列问题. 本文通过对组合数学中的这 3 类数列问题进行了深入研究,发现 Catalan 数列和 Fibonacci 数列问题均可以利用数学分析法从规模最小的子问题出发,通过归纳推理、组合数学中的加法原理和乘法原理等方法获得原问题的解和若干子问题的解之间满足的函数关系,得到问题的求解函数.

进一步的研究表明: 这 2 类数列问题在算法设计过程中都是从最小规模的子问题出发,利用所获得的问题求解函数不断求解更大规模子问题的解,直到得出原问题的解. 因此,在算法设计时每一步的求解都必须使用变量记录当前已求出的子问题的解,这是该 2 类数列问题在算法设计过程中的一个共性特征. 根据这一特征,可以提炼出这 2 类数列问题循环不变式的统一开发策略. 步骤如下:

(i) 利用归纳推理、组合数学中的加法原理和乘法原理等数学方法获得问题求解函数 f;

(ii) 根据问题求解函数 f,分析在问题求解函数 f 中所包含的子问题个数;

(iii) 若原问题中子问题个数大于 3,则使用数组变量记录当前每一个已求子问题的解; 若子问题个数小于等于 3,则分别用简单变量记录已求子问题的解;

(iv) 约束循环变量的变化范围,即可获得问题求解算法的循环不变式.

2 形式化开发算法程序的过程

复杂的算法程序一般采用循环结构加以实现. 而循环结构通常由 3 个部分构成: 循环的初始化语句序列 S_0、循环条件 $Guard$ 和循环体 S. 本文将采用 Dijkstra 算法形式化推导方法,根据问题的前置断言和后置断言以及使用本文提出的策略开发的循环不变式完成算法中这 3 个部分的形式化推导过程. 具体过程如下:

(i) 通过对 2 类数列问题数学性质和求解特征进行数学分析,利用组合数学中的加法原理和乘法原理得到问题求解函数 f.

(ii) 根据问题需要实现的功能,使用(i) 中得到的求解函数 f,形式化描述问题的程序规约 (Q,R),其中 Q 表示前置断言,R 表示后置断言.

(iii) 根据本文提出的策略得到问题求解算法的循环不变式 ρ.

(iv) 利用(iii) 中得到的循环不变式 ρ,结合(ii) 中给出的 Q 和 R,形式化开发问题求解的算法程序,步骤为

(a) 根据 $Q \Rightarrow WP("S_0",\rho)$ 求出循环的初始化语句序列 S_0;

(b) 根据 $\rho \wedge \neg\ Guard \Rightarrow R$ 求出循环条件 $Guard$.

(c) 根据每次循环体执行前后 ρ 应为真的性质求出循环体 S.

(d) 根据上述(a)、(b) 和(c) 的推导得出最终算法程序.

{Q: 前置断言}
S_0;
{ρ: 循环不变式}
while ($Guard$)
    S
{R: 后置断言}.

3 开发实例

本小节将通过 2 个实例,利用第 1 小节的开发策略开发问题算法程序的循环不变式,利用第 2 小节中的开发步骤,基于循环不变式推导出未知算法程序.

3.1 二叉树的形态数问题

问题描述: 已知二叉树有 n 个节点,求二叉树有多少种形态.

(i) 使用归纳推理法寻找问题求解函数. 从最小规模的子问题开始讨论,在讨论中,定义一个 2 元组 (l,r),其中 l 表示根节点左子树的节点数,r 表示根节点右子树的节点数. 使用函数 $f(i)$ 表示 i 个节点的二叉树所构成的形态数目. 使用归纳推理法对原问题进行数学分析,过程为

(a) 当二叉树有 0 个节点时,只有 1 种形态,$f(0)=1$;

(b) 当二叉树有 1 个节点时,只有 1 种形态,$f(1)=1$;

(c) 当二叉树有 2 个节点时,固定 1 个节点作为根节点,剩余节点去构建根节点的左、右子树,包含 2 种情况: (0,1)、(1,0),则 $f(2)=f(0)f(1)+$

$f(1)f(0)=2$.

(d) 当二叉数有 3 个节点时,先固定 1 个节点作为根节点,剩余节点去构建左、右子树,包含 3 种情况:(0,2)、(1,1)、(2,0),则 $f(3)=f(0)f(2)+f(1)f(1)+f(2)f(0)=5$.

(e) 二叉树有 4 个节点时,先固定一个节点作为根节点,剩余节点去构建左右子树,包含 4 种情况:(0,3)、(1,2)、(2,1)、(3,0),则 $f(4)=f(0)\cdot f(3)+f(1)f(2)+f(2)f(1)+f(3)f(0)=14$.

根据组合数学中的加法原理和乘法原理可求 n 个节点的二叉树形态数目,求解公式为

$$f(n)=f(0)f(n-1)+f(1)f(n-2)+\cdots+f(n-1)f(0)\ (n\geqslant 2). \tag{1}$$

经过以上推导,可以得到的问题求解函数为

$$f(n)=\begin{cases}1, & n=0,\\ 1, & n=1,\\ \sum\limits_{k=0}^{n-1}f(k)f(n-k-1), & n\geqslant 2.\end{cases} \tag{2}$$

(ii) 形式化描述问题的程序规约.

Q 表示给定 n 个节点的二叉树 T;

R 表示求 T 所具有的形态数 $f(n)$,其中 $f(n)$ 定义如(2)式所示.

(iii) 根据第 1 小节的开发策略得到求解算法的循环不变式.

根据(1)式获得的问题求解函数可知,原问题的解建立在多个子问题的基础上,即在求 $f(j)$ 的解时,$f(j)$ 是建立在 $f(j-1)$、$f(j-2)$、$\cdots$、$f(2)$、$f(1)$、$f(0)$ 这些子问题的基础上. 由于原问题的解是建立在多个子问题的基础上,采用数组变量 $a[i]$($i=1,2,\cdots,n$) 记录当前已经求解的每一个子问题的解,并约束循环控制变量的变化范围即可得出问题求解算法的循环不变式,其描述为

$$\rho: 2\leqslant i\leqslant n+1\wedge(\forall j: 2\leqslant j<i: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1.$$

(iv) 利用(iii)中得到的循环不变式开发算法程序. 过程如下:

(a) 根据 $Q\Rightarrow WP("S_0",\rho)$ 求循环的初始化语句 S_0.

由于循环变量 i 从 2 开始循环,因此初始化语句 S_0 包含 $i:=2$,设 S_0 由 S_1 和 $i:=2$ 组成,则

$$WP("i:=2",\rho)\equiv \text{true}\wedge\text{true}\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1\equiv a[0]=f(0)=1\wedge a[1]=f(1)=1.$$

由 $Q\Rightarrow WP("S_1",WP("i:=2",\rho))$ 得出 S_1 为简单赋值语句 $a[0]:=1$ 和 $a[1]:=1$,因此初始化语句 S_0 为 $a[0]:=1;a[1]:=1;i:=2$.

(b) 根据 $\rho\wedge\Rightarrow\neg\ Guard\Rightarrow R$ 求出循环条件 $Guard$. 这里

$$\rho\wedge\neg\ Guard\Rightarrow R\equiv 2\leqslant i\leqslant n+1\wedge(\forall j: 2\leqslant j<i: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)\cdot f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1\wedge\neg\ Guard\Rightarrow f(n)=(\sum k: 0\leqslant k<n: f(k)f(n-k-1)).$$

由上式可知,当 $i=n+1$ 时可得到后置断言 R 中 $f(n)$ 的结果,因此 $\neg\ Guard$ 取 $i\geqslant n+1$,则 $Guard$ 为 $i<n+1$.

(c) 根据每次循环体执行前后 ρ 为真的性质求出循环体 S.

通过 $\{\rho\wedge Guard\}S\{\rho\}$ 为真求循环体 S. 由于在整个循环过程中始终有循环控制变量 i 在递增,推断循环体 S 中有语句 $i:=i+1$. 假定循环体 S 是由 S' 和语句 $i:=i+1$; 组成,则 $\{\rho\wedge Guard\}S'\{\rho_{i+1}^{i}\}$ 为真. 这里

$$\rho\wedge Guard\equiv 2\leqslant i\leqslant n+1\wedge(\forall j: 2\leqslant j<i: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1\wedge i<n+1\equiv 2\leqslant i<n+1\wedge(\forall j: 2\leqslant j<i: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1.$$

$$\rho_{i+1}^{i}\equiv(2\leqslant i\leqslant n+1\wedge(\forall j: 2\leqslant j<i: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1)_{i+1}^{i}\equiv 2\leqslant i+1\leqslant n+1\wedge(\forall j: 2\leqslant j<i+1: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1\equiv 1\leqslant i\leqslant n\wedge(\forall j: 2\leqslant j<i+1: a[j]=f(j)\wedge f(j)=(\sum k: 0\leqslant k<j: f(k)f(j-k-1)))\wedge a[0]=f(0)=1\wedge a[1]=f(1)=1.$$

根据 ρ_{i+1}^{i} 以及 $(\rho\wedge Guard)$ 的值可知,S' 执行的功能是计算 $f(i)$ 的值并把它存入到数组 $a[i]$ 中. 根据(2)式可知,$f(i)=f(0)f(i-1)+f(1)f(i-2)+\cdots+f(i-2)f(1)+f(i-1)f(0)$; 又已知 $f(0)$,$f(1)$,$f(2)$,$\cdots$,$f(i-1)$ 分别存入到 $a[0]$,$a[1]$,$a[2]$,$\cdots$,$a[i-1]$ 中,则 S' 为

$$a[i]=a[0]a[i-1]+a[1]a[i-2]+\cdots+a[i-2]a[1]+a[i-1]a[0].$$

(d) 根据以上推导获得算法程序为

```
begin
a[0]:=1; a[1]:=1; i:=2;
do i < n+1 →
a[i]=a[0]a[i-1]+a[1]a[i-2]+…+a[i-2]a[1]+a[i-1]a[0];
i:=i+1;
od;
write(a[n]);
end.
```

3.2 阶梯问题

问题描述: 已知 1 个阶梯共有 n 级,某人每步可走 1 级、2 级或者 3 级阶梯,求走完 n 级阶梯的方案数.

(i) 使用归纳推理法寻找问题求解函数.

从最小规模的子问题开始讨论,在讨论中,定义 $i=x_1+x_2+\cdots+x_r$,其中 i 表示阶梯级数,x_1 表示第 1 步走的阶梯级数,x_2 表示第 2 步走的阶梯数,…,x_r 表示第 r 步走的阶梯级数. $i=x_1+x_2+\cdots+x_r$ 表示经过 r 步走完 i 级阶梯. 函数 $f(i)$ 表示走完 i 级阶梯的全部方案. 由于规定每一步只能走 1 级、2 级或 3 级阶梯,因此分 3 种情况进行讨论,即最后一步走 1 级阶梯、最后一步走 2 级阶梯和最后一步走 3 级阶梯. 使用归纳推理法对原问题进行分析,过程如下:

(a) 当有 1 级阶梯时,只有 1 种走法,$f(1)=1$.

(b) 当有 2 级阶梯时,走法为 $2=1+1$(只有 1 种情况)、$2=2$(只有 1 种情况),$f(2)=2$.

(c) 当有 3 级阶梯时,走法为 $3=1+1+1=2+1$(只有 2 种情况)、$3=1+2$(只有 1 种情况)、$3=3$(只有 1 种情况),$f(3)=2+1+1=4$.

(d) 当有 4 级阶梯时,走法为 $4=1+1+1+1=1+2+1=2+1+1=3+1$(有 $f(3)$ 种情况)、$4=1+1+1+2=2+2$(有 $f(2)$ 种情况)、$4=1+3$(有 $f(1)$ 种情况),$f(4)=f(3)+f(2)+f(1)=7$.

(e) 当有 5 级阶梯时,走法为 $5=1+1+1+1+1=1+2+1+1=2+1+1+1=3+1+1=1+1+2+1=2+2+1=1+3+1$(有 $f(4)$ 种情况)、$5=1+1+1+2=2+1+2=1+2+2=3+2$(有 $f(3)$ 种情况)、$5=1+1+3=2+3$(有 $f(2)$ 种情况),$f(5)=f(4)+f(3)+f(2)=13$.

通过归纳推理法对原问题进行数学分析,得出问题求解隐含的数学性质,即在求走完 $m(m\geqslant 4)$ 级阶梯的方案数时,若最后 1 步走 1 级阶梯,则方案数为 $f(m-1)$;若最后 1 步走 2 级阶梯,则方案数为 $f(m-2)$;若最后 1 走 3 级阶梯,则方案数为 $f(m-3)$. 根据组合数学的加法原理得到走完 m 级阶梯的所有方案数,具体求解公式为

$$f(m)=\sum_{k=1}^{3}f(m-k),\ m\geqslant 4.$$

根据以上推导可得原问题求解函数 f 为

$$f(m)=\begin{cases}1, & m=1,\\ 2, & m=2,\\ 4, & m=3,\\ \sum_{k=1}^{3}f(m-k), & m\geqslant 4.\end{cases}\tag{3}$$

(ii) 形式化描述问题程序规约.

Q 表示已知阶梯共有 n 级.

R 表示求走完 n 级阶梯的方案数 $f(n)$,其中 $f(n)$ 的定义如(3)式所示. 则

$$f(n)=\begin{cases}1, & n=1,\\ 2, & n=2,\\ 4, & n=3,\\ \sum_{j=1}^{3}f(n-j), & n\geqslant 4.\end{cases}$$

(iii) 根据第 1 小节的开发策略得到问题求解算法的循环不变式.

根据(i)中获得的问题求解函数可知,在原问题的解是建立在 3 个子问题的基础上. 即在求 $f(m)$ 时,$f(m)$ 是建立在 $f(m-1)$、$f(m-2)$、$f(m-3)$ 这 3 个子问题的基础上. 分别使用简单变量 f_1、f_2、f_3 和 f 记录 $f(m-3)$、$f(m-2)$、$f(m-1)$ 和 $f(m)$ 的值,并约束循环控制变量的变化范围即可得出问题求解算法的循环不变式 ρ,其描述为

ρ: $f_1=f(m-3)\wedge f_2=f(m-2)\wedge f_3=f(m-1)\wedge f=f(m)\wedge f(m)=f(m-1)+f(m-2)+f(m-3)\wedge 4\leqslant m\leqslant n$.

(iv) 利用(iii)中得到的循环不变式开发算法程序.

(a) 根据 $Q\Rightarrow WP("S_0",\rho)$ 求循环初始化语句序列 S_0.

由于循环变量 m 从 4 开始循环,因此循环初始化语句 S_0 包含 $m:=4$,设 S_0 由 S_1 和 $m:=4$ 组成,由于

$WP("m:=4",\rho)\equiv f_1=f(1)\wedge f_2=f(2)\wedge f_3=f(3)\wedge f=f(4)\wedge f(4)=f(3)+f(2)+f(1)\wedge \text{true}\equiv f_1=f(1)\wedge f_2=f(2)\wedge f_3=f(3)\wedge f=f(4)\wedge f(4)=f(3)+f(2)+f(1)$.

由于 $f(1)=1$,$f(2)=2$,$f(3)=4$,因此有

$WP("m:=4",\rho)\equiv f_1=1\wedge f_2=2\wedge f_3=4\wedge f=f(4)\wedge f(4)=7\equiv f_1=1\wedge f_2=2\wedge f_3=4\wedge f=7$,并且 $Q\Rightarrow WP("S_1",WP("m:=4",$

ρ)). 可以得出 S_1 为简单赋值语句 f_1: = 1; f_2: = 2; f_3: = 4; f: = 7,则初始化语句 S_0 为

S_0: f_1: = 1; f_2: = 2; f_3: = 4; f: = 7; m: = 4.

(b) 根据 $\rho \wedge \neg Guard \Rightarrow R$ 求循环条件 $Guard$. 这里

$\rho \wedge \neg Guard \Rightarrow R \equiv f_1 = f(m-3) \wedge f_2 = f(m-2) \wedge f_3 = f(m-1) \wedge f = f(m) \wedge f(m) = f(m-1) + f(m-2) + f(m-3) \wedge 4 \leqslant m \leqslant n \wedge \neg Guard \Rightarrow f(n) = f(n-1) + f(n-2) + f(n-3)$.

由上式可知,当 $m = n$ 时可得到后置断言 R 中 $f(n)$ 的结果,因此 $\neg Guard$ 为 $m \geqslant n$,从而得出 $Guard$ 为 $m < n$.

(c) 根据每次循环体执行前后 ρ 应为真的性质求循环体 S.

通过 $\{\rho \wedge Guard\} S \{\rho\}$ 为真,求出循环体 S. 由于在整个循环过程循环控制变量 m 始终是在递增,所以在循环体中考虑语句 m: = m + 1. 设循环体 S 是由 S' 和语句 m: = m + 1 组成,则 $\{\rho \wedge Guard\} S' \{\rho\}_{m+1}^{m}$ 为真. 并且

$\rho \wedge Guard \equiv f_1 = f(m-3) \wedge f_2 = f(m-2) \wedge f_3 = f(m-1) \wedge f = f(m) \wedge f(m) = f(m-1) + f(m-2) + f(m-3) \wedge 4 \leqslant m \leqslant n \wedge m < n \equiv f_1 = f(m-3) \wedge f_2 = f(m-2) \wedge f_3 = f(m-1) \wedge f = f(m) \wedge f(m) = f(m-1) + f(m-2) + f(m-3) \wedge 4 \leqslant m < n$,

$\rho_{m-1}^{m} \equiv (f_1 = f(m-3) \wedge f_2 = f(m-2) \wedge f_3 = f(m-1) \wedge f = f(m) \wedge f(m) = f(m-1) + f(m-2) + f(m-3) \wedge 4 \leqslant m \leqslant n)_{m+1}^{m} \equiv f_1 = f(m-2) \wedge f_2 = f(m-1) \wedge f_3 = f(m) \wedge f = f(m+1) \wedge f(m+1) = f(m) + f(m-1) + f(m-2) \wedge 4 \leqslant m+1 \leqslant n \equiv f_1 = f(m-2) \wedge f_2 = f(m-1) \wedge f_3 = f(m) \wedge f = f(m+1) \wedge f(m+1) = f(m) + f(m-1) + f(m-2) \wedge 3 \leqslant m \leqslant n-1$.

根据 ρ_{m+1}^{m} 以及 $\rho \wedge Guard$ 的值可知,S' 完成的功能是计算 $f(m+1)$ 的值. 已知 $f(m+1) = f(m) + f(m-1) + f(m-2)$,根据(3)式可知 $f(m) = f(m-1) + f(m-2) + f(m-3)$,同时由于 $f(m-3)$、$f(m-2)$、$f(m-1)$ 和 $f(m)$ 的值分别用简单变量 f_1、f_2、f_3 和 f 记录,则循环体 S 为

f_1: = f_2;

f_2: = f_3;

f_3: = f;

f: = $f_1 + f_2 + f_3$.

(e) 根据以上推导获得算法程序.

考虑到问题完整性,将原问题包含的 3 种特殊情况也利用 if 语句在程序最前面判断后,直接输出相应结果. 得到原问题的算法程序为

```
if(n == 1) printf("1\n");
else if(n == 2) printf("2\n");
    else if(n == 3) printf("4\n");
        else{
            int m = 4;
            int f1 = 1,f2 = 2,f3 = 4,f = 7;
            while(m < n)
            {
            f1 = f2;
            f2 = f3;
            f3 = f;
            f = f1 + f2 + f3;
            m = m + 1;
            }
        printf("%d\n",f);}.
```

4　总结和展望

本文提出了 2 类数列问题循环不变式的统一开发策略,在本策略中问题求解函数是通过数学归纳以及组合数学中的加法和乘法原理等数学方法获取的,因而有效地保证了问题求解函数的正确性,进而保证了该开发策略的有效性.

传统的标准开发策略虽适用的范围较广,但主要针对较简单的算法问题有效; 而本文所提出的统一开发策略,主要适用于 2 类复杂的组合数学问题. Janota 的自动探测技术适用于已知算法程序循环不变式的开发,而本文的开发策略可以开发未知算法的循环不变式. 薛锦云[14] 提出的策略对于循环不变式开发具有宏观的指导意义,而本文的策略是对其宏观策略的一个具体的细化. 目前,笔者已经使用本文提出的循环不变式开发策略开发了握手问题、排队找零问题、出栈问题等 Catalan 数列问题以及青蛙跳台阶问题、兔子繁殖问题等 Fibonacci 数列问题的循环不变式,为具有Catalan数列和 Fibonacci 数列性质的组合数学问题循环不变式开发提供了统一的开发途径.

循环不变式的开发是一种创造性劳动,寻找循环不变式开发策略一直存在较多难点. 本文提出的循环不变式统一开发策略主要适用于 Catalan 数列和 Fibonacci 数列这 2 类组合数学问题循环不变的开发. 是否适用于其它组合问题循环不变式的开发,今后还需做进一步深入研究.

5 参考文献

[1] 万松松,薛锦云,谢武平. 循环不变式开发技术研究[J]. 计算机工程与科学,2010,32(9):84-88,94.

[2] 刘自恒,曾庆凯. 一种自适应的循环不变式生成方法[J]. 计算机工程,2013,39(6):76-81.

[3] 王晓东,吴英杰,傅仰耿,等. 算法归纳设计策略与循环不变式[J]. 福州大学学报:自然科学版,2004,32(4):387-392.

[4] 杨黄磊,薛锦云. 一类单元赋值语句型循环不变式的开发方法研究[J]. 江西师范大学学报:自然科学版,2014,38(4):378-382.

[5] 潘建东,陈立前,黄达明,等. 含有析取语义循环的不变式生成改进方法[J]. 软件学报,2016,27(7):1741-1756.

[6] 许欢,王以松. 用 daikon 发现循环不变式[J]. 贵州大学学报:自然科学版,2012,29(4):77-81.

[7] Gries D. A note on a standard strategy for developing loop invariants and loops[J]. Science of Compute Programming,1982,2(3):207-214.

[8] 黄小明. 循环不变式自动探测[D]. 南昌:江西师范大学,2017.

[9] Zhai Juan,Wang Hanfei,Zhao Jianhua. Post-condition-directed invariant inference for loops over data structures[EB/OL]. [2019-05-11]. https://ieeexplore.ieee.org/document/6901659.

[10] 王树义,邹伟松,刘恒. 基于知识的循环不变式生成方法[C]//中国计算机学会. 2001 全国软件技术研讨会论文集. 大连:大连出版社,2001:38-40.

[11] 李彬,翟娟,汤震浩,等. 自动合成数组不变式[J]. 软件学报,2018,29(6):1544-1565.

[12] Hou Chunyan,Wang Jinsong,Chen Chen,et al. Loop invariant generation for non-monotone loop structures[EB/OL]. [2019-05-11]. https://www.researchgate.net/publication/326103698_Loop_Invariant_Generation_for_Non-monotone_Loop_Structures.

[13] Janota M. Assertion-based loop invariant generation[EB/OL]. [2019-05-11]. http://hdl.handle.net/10344/2127.

[14] Xue Jinyun. Two new strategies for developing loop invariants and their applications[J]. Journal of Computer Science and Technology,1993,8(2):147-154.

[15] 游颖. 算法形式化方法在 3 类组合数学问题求解中的应用研究[D]. 南昌:江西师范大学,2017.

The Research and Application of Loop Invariant Development Strategy for Two Kinds of Sequence Problems

GU Sumei, YANG Qinghong*

(College of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

Abstract: Based on the combination of mathematics in the Catalan sequence and the Fibonacci sequence problems in-depth study, using the method of inductive reasoning, and the methods of addition and multiplication principle of combinatorial mathematics problem solving function, the variable problem of neutron log algorithm process of solution and the change of constraint loop variable range are used, the loop invariant of problem solving algorithm is obtained, giving the problem of two kinds of sequence loop invariant of the unification of the development strategy. Taking binary tree morphological number problem and ladder problem as examples, the proposed strategy is used to develop loop invariants.

Key words: sequence problem; loop invariant; trustworthy software; inductive reasoning

(责任编辑:冉小晓)