文章编号:1000-5862(2020)06-0625-08

二叉树排序非递归算法推导及形式化证明

左正康[1],方 越[1],黄 箐[1],廖云燕[1],王 渊[2],王昌晶[1*]
(1. 江西师范大学计算机信息工程学院,江西 南昌 330022;2. 江西师范大学软件学院,江西 南昌 330022)

摘要: 非线性数据结构递归问题非递归算法的循环不变式的开发一直是形式化开发的难点. 研究二叉树类非递归算法的推导及形式化证明方法,对二叉树排序算法进行推导,得出非递归 Apla(Abstract Programming Language)算法及其精确而简单的循环不变式,然后用 Dijkstra-Gries 标准程序证明法证明算法的正确性,最后使用 PAR 平台 C++程序自动生成系统自动生成 C++代码. 实例的实验结果简化了算法程序的推导和证明过程,对递归问题非递归算法的循环不变式的探测具有一定的借鉴意义,而且对非线性数据结构算法程序的推导及形式化证明具有指导意义.

关键词: 二叉树类非递归算法; 循环不变式; PAR 平台; Dijkstra-Gries 标准程序证明法; 非线性数据结构
中图分类号: TP 311 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2020.06.14

0 引言

二叉树类非递归算法的推导及形式化证明一直是推导及形式化证明的难点. 递归算法是一种较好的解决方式,虽然递归算法代码简单,但其消耗时间长、占用空间较大. 因此,人们更倾向于使用高效的非递归算法来解决二叉树问题. 然而,非递归算法代码繁多,逻辑关系复杂不易理解,证明过程晦涩难懂,而且正确性得不到保证[1]. 如何能找到二叉树的非递归算法的循环不变式成为一大难点.

早期 D. Gries 对 E. W. Dijkstra 提出的构造循环不变式开发策略进行了补充和解释[2-3],提供了先序遍历二叉树非递归算法的循环不变式,但逻辑关系复杂,并且表达十分繁琐,没有较好的实用性和可行性. 因而人们期待能研发出一个更有效的算法设计方法,以此来开发和理解问题算法[4].

本文依据 PAR 方法开发循环不变式的新定义和新的开发策略[5-6],选取二叉树排序非递归算法作为实例,进行推导及形式化证明,对递归问题非递归算法的循环不变式进行探测,对非线性数据结构算法程序的推导及形式化证明具有指导意义.

1 相关工作

形式化方法基于严格数学基础,对计算机软硬件系统进行描述、开发和验证,可提高软件开发的正确性和可靠性. 虽然形式化方法发明了 40 多年,但是学术界对于形式化方法的定义和本质特征并没有统一的认识. G. Gargantini 等[7]认为许多从业者仍然不愿意采用正式的方法,除了众所周知的缺乏训练,这种怀疑主义主要是由于复杂的符号以及在系统开发的生命周期活动中缺乏支持开发人员的易于使用的工具. F. P. Brooks[8]根据形式化开发要从需求规约得到软件代码,提出了软件开发复杂性来源,将软件开发建立在一个有理论基础的工程过程中. Floyd-Hoare 逻辑[9-10]是一个经典的验证系统,该系统使用严格的数理逻辑推理来为计算机程序的正确性提供一组逻辑规则. 结合 Floyd-Hoare 逻辑的规则和公理,将已验证的程序转化为一组数学命题的证明. S. Owicki 等[11]提出一种通用的并发程序验证方法,作为顺序程序进行单独验证,并检查其是否存在干扰性,为此来保证验证过程是正确的.

收稿日期:2019-10-17
基金项目:国家自然科学基金(61862033,61762049,61902162),江西省自然科学基金(20202BABL202026,2020BABL202025,20202BAB202015)和国家留学基金(202008360094)资助项目.
作者简介:左正康(1980-),男,江西抚州人,教授,博士,主要从事软件形式化、可信软件方面的研究. E-mail:kerrykaren@126.com
通信作者:王昌晶(1977-),男,江西南昌人,教授,博士,博士生导师,主要从事软件形式化方法、形式规格说明方法和 Web 服务方面的研究. E-mail:wcj771006@126.com

寻找循环不变式的方法有很多种,基于循环不变式的构造技术最主要有 2 种: 参数化模型和约束求解[12-13]; 抽象解释[14-15]基于消元法对程序进行验证[16],可生成非线性循环不变式,并判定循环程序的终止性,如基于吴方法和 Dixon 结式.

二叉树是计算机科学中常见的数据结构. 二叉树的遍历分为迭代遍历和递归遍历. 递归算法更简单、更容易理解,但它需要更多的时间和空间. 毫无疑问,非递归算法的效率要高于递归算法. 与更多关注二叉树非递归算法的研究[17-18]不同,A. Loginov[19]验证了用破坏性指针操作遍历二叉树的 Deutsch-Schorr-Waite(DSW)算法,并使用测试向量泄漏评估(TVLA)方法来验证 DSW 算法的正确性. D. Gries[2-3]对 E. W. Dijkstra 提出的构造循环不变式开发策略进行了补充和解释,提供了先序遍历二叉树非递归算法的循环不变式.

本文基于 PAR 方法开发循环不变式的新定义和新的开发策略,推导及形式化证明二叉树排序非递归 Apla 算法,使用 PAR 平台 C++程序自动生成系统将 Apla 算法自动生成 C++可执行程序.

2 预备知识

2.1 PAR 方法和 PAR 平台

为简化算法过程以此得到简单的算法表达式、实现算法程序开发方法的普遍适用性的目标. 于是,本实验室研究团队提出了 PAR 方法[20],它是一个基于分划和递推的方法. PAR 方法是一个统一的算法设计方法,支持形式化推导和开发所用的构件[21]. PAR 平台是支撑 PAR 方法所开发的平台,提高了程序的正确性和简洁性. PAR 方法及 PAR 平台包括算法规约和描述算法语言的算法设计语言 Radl[22-23]、实现形式化开发的抽象程序设计语言 Apla 以及 PAR 平台程序自动生成系统(可生成 Java、VB. net 和 C++等可执行程序).

对需求模型的细化,可以基于循环不变式的新定义和新的开发策略,采用 Dijkstra-Gries 标准算法和程序正确性证明方法,关于二叉树类模型对相关算法和程序进行形式化或自动验证; 还可以对定义的模型进行细化,进而得到算法模型、抽象程序模型和可执行语言程序模型. 这些模型也可以由模型自动转换器自动生成[24]. PAR 架构如图 1 所示. 有 2 种方法可以生成代码,第 1 种方法适用于处理可量化问题,即目标或问题具体明确,可以清晰度量,根据不同的情况,可表示成具体的统计数字的问题. PAR 平台可以将 SNL 需求模型自动转换为 Radl 规范模型,并半自动转换为 Radl 算法模型,再自动转换为 Apla 程序模型,最后转换为可执行程序. 第 2 种方法适用于处理不可量化问题,即无法表示成具体的统计数字的问题. 用户可以直接设计 Apla 程序并给出形式化证明或定理证明,再将其转为可执行程序.

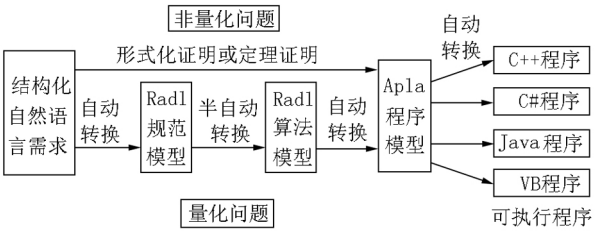

图 1 PAR 架构

2.2 循环不变式

循环前后均为真的谓词被称为循环不变式[25]. 理解掌握算法程序和实现推导、验证及自动生成均需要循环不变式[26]. 循环不变式是形式化开发算法程序的重要环节[27].

2.3 Apla 语言

抽象程序设计语言 Apla(Abstract Programming Language)[28-30]由递推关系描述的 Radl 算法自动生成,形式化开发算法程序,以此实现算法的自动化,提高了算法程序的可靠性和开发效率. Apla 具有泛型机制,可用于顺序、分布式、并行并发等程序开发[31]. Apla 具有功能抽象和数据抽象,使程序的清晰易懂,便于形式化推导和验证,更易于转换成 Java、VB. net 和 C++等可执行程序[32]. 并且,利用 Apla 提供的预定义 ADT,可以直观地进行程序设计,让关系数据库更容易访问. 这些特性可极大地缩短算法程序的设计过程和代码长度,便于程序开发、理解和验证,提高开发效率和可靠性.

2.4 序列

序列(list)是被排成一列的对象(或元素). 访问元素所做的操作依赖于具体的应用问题. 遍历序列在数据结构的二叉树搜索中经常用到. 算法设计语言 Radl 提供了序列的预定义 ADT 类型,包括相关数据和操作描述,具体如下所示:

```
Specify ADT list(sometype data, [size])
//data 为序列的元素类型,size 为序列的元素个数
var h,t: integer: =0,0;
//h 表示序列头的域名,t 表示序列尾的域名
```

var S,T: list: = Ⅰ, Ⅰ; //S 和 T 是序列
x: data; i,j: integer; //x 为序列中的元素
Operator:
Ⅰ //序列为空
[x] //序列中有一个元素 x
#(S) //计算 S 中元素的个数
S [i] //S 的第 i 个元素,$S.h \leqslant i \leqslant S.t$
S [i..j] //S 的子序列,$S.h \leqslant i,j \leqslant S.t$
$S \uparrow T$
//S 的尾和 T 的头用"↑"运算合成一个新序列
Endspec.

2.5　二叉树

二叉树(btree)是一种树型结构. 算法设计语言 Radl 提供了二叉树的预定义 ADT 类型,设 rnode 为二叉树的根结点记录类型,btree 为二叉树类型,data 为二叉树结点值的参数类型,定义为

type rnode = rnode (sometype data) = record
t: data//data 表示二叉树结点 t 的值域类型
l: btree
r: btree
end.

下面给出本文中所用的二叉树 ADT 相关数据和操作描述:

Specify ADT btree (sometype data, [size])
var T: btree: = % //二叉树 T 为空
n: rnode; //n 是二叉树的一个结点
Operator:
$T.d$//产生二叉树 T 的根结点
$T.l$//产生二叉树 T 的左子树
$T.r$//产生二叉树 T 的右子树
$n + T$
//把结点 n 加到 T 中,使 T 成为和 T 结构相同的新树
ReadNode(n) //建立结点 n
WriteNode(n) //输出结点 n
Endspec.

3　二叉树类算法推导方法研究与应用

3.1　二叉树类算法推导方法研究

3.1.1　**开发步骤**　结合各类二叉树非递归算法的推导及形式化证明,总结出 5 个步骤[33]:

(i) 构造形式化规范来明确二叉树类算法的工作目标.

AQ: 给定一个有限的二叉树 T;
AR: $X = T$ 的二叉树类问题产生的节点序列.

(ii) 把二叉树进行划分,得到一定数目的子树,子树满足规模比二叉树小、结构和二叉树相同这 2 大特征,再把子树进行相同方式分解,直到求解出每个子树.

(iii) 用量词转换法推导出序列的递推关系 $S_i = F(S_j)$,并把初值赋值给函数和变量.

(iv) 开发循环不变式[34],根据变量中元素与递推关系中子树的关系及递推关系中对子树的解的描述,采用递归方式定义序列(栈或者队列)中的内容.

(v) 依据递推关系和循环不变式,导出 Apla 算法程序.

3.1.2　**改进循环不变式**　E. W. Dijkstra 等[35]根据通过弱化后置断言 R 可以得到 ρ 这一原理,给出了 4 个循环不变式的传统开发策略; D. Gries 更对 E. W. Dijkstra 提出的构造循环不变式开发策略进行了补充和解释,提供了先序遍历二叉树非递归算法的循环不变式:

ρ: $0 \leqslant c \leqslant \#p \wedge preorder(p) = b$ [0: $c-1$] | $preorder(r_0)$ | $\cdots$ | $preorder(r_{|r-1|})$.

通过观察,可看出此循环不变式用"…"表示显得非常冗长,表达不清晰、不易理解. 于是,文献[5-6]提出了基于 PAR 方法开发循环不变式的新定义和新的开发策略.

定义 1　称变量的值随循环的改变而改变为循环变量.

定义 2　循环变量在集合 A 中,且在每次循环前后在集合 A 中的所有元素的变化规律均为真的谓词被称为循环 Do 的循环不变式.

循环不变式新策略:

策略 1　保证循环程序的验证正确,观察所得信息,根据其变化规律分析问题的数学性质、实际背景和程序特征,即为所求循环不变式.

策略 2　运用可靠地算法设计方法确定解决问题的总策略和全部所需循环变量,对每一变量的变换规律进行描述,以此考察求解问题的相关数学性质和实际背景,即为求得的循环不变式. 递归定义序列中的内容需在递推关系中子解个数超过 1 的情况下来引进堆栈作用的序列变量.

通过循环不变式的新定义和新的开发策略,提供了新的先序遍历二叉树非递归算法的循环不变式:

ρ: $Pre(T) = X_2 \uparrow Pre(q) \uparrow F(S)$.

比较 2 个先序遍历二叉树非递归算法的循环不变式,可以较清晰地发现新的循环不变式更精确、简单. 便于二叉树类非递归算法的推导.

3.1.3 递推关系 通过分析被求解问题的背景知识和相关数学特性,确定二叉树类问题求解序列的递推关系和所需的全部循环变量,用谓词精确表达它们的变化规律.

用序列实现的栈和队列表达了二叉树类问题的 2 类递推关系.

对于栈,可以看成是一个序列 $S[0\cdots\#S-1]$,并规定以 $S[\#S-1]$ 端为栈顶,以 $S[0]$ 端为栈底,则栈 S 的常用操作可用如下序列的部分操作加以实现:

测试栈 S 是否为空: $\#S=0$;

取栈 S 的栈顶之素: $X:=S[\#S-1]$;

对栈 S 实施进栈操作: $S:=S\uparrow[X]$;

对栈 S 实施出栈操作: $S:=S[0\cdots\#S-2]$.

对于队列,也可以看成是一个序列 $q[0\cdots\#q-1]$,并规定以 $q[0]$ 端为队头,以 $q[\#q-1]$ 端为队尾,则队列的常用操作可以用如下序列的操作加以实现:

测试队列是否为空: $\#q=0$;

引用队头元素: $X:=q[0]$;

对队列 q 实施进队: $q:=q\uparrow[X]$;

对队列 q 实施出队: $q:=q[1\cdots]$.

将序列及其运算封装在一起构成抽象数据类型,可以方便地进行程序设计. 对于本文的实例,使用栈的存储结构模拟递归的思想实现.

3.2 二叉树排序算法推导

二叉排序树,又称二叉查找树,亦称二叉搜索树. 它可以是一棵空树; 也可以是具有下列性质的二叉树:

1) 若左子树不为空,则左子树上所有结点的值均小于它的根结点的值;

2) 若右子树不为空,则右子树上所有结点的值均大于它的根结点的值;

二叉树排序的基本原理如图 2 所示,排序过程如下:

(i) 输入一个无序的整数序列;

(ii) 通过 PAR 平台的预定义的操作,建立二叉排序树,序列第 1 个元素作为根结点,序列中下一个元素小于根结点的值放在左边,大于根结点的值放在右边;

(iii) 对二叉排序树进行中序遍历的算法,并动态显示结果;

(iv) 得到从小到大的有序的整数序列.

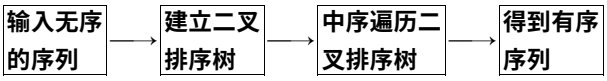

图 2 二叉树排序算法

3.2.1 建立二叉排序树 建立二叉排序树是 PAR 平台的预定义的操作,可以通过 Apla 程序快速地建立二叉排序树:

```
const size = 10; //10 个字符结点
var
    t: tree; //t 代表二叉排序树
    i: integer; //i 整数类型
    n: integer; //n 整数类型
do i≤size→readln(n);
//建立 10 个结点的二叉排序树
t,i:=t+n,i+1; od;
//t+n 为整数类型的结点 n 插入二叉排序树 t.
```

3.2.2 中序遍历二叉排序树 中序遍历是在二叉树排序中的一个步骤,通过对二叉排序树进行中序遍历,最终得到有序序列. 中序遍历(Inorder Traversal) 是二叉树遍历的一种. 对中序遍历非递归算法进行形式化推导,得到 Apla 算法,便于程序的开发和自动生成可执行程序. 该形式化推导由如下 5 个步骤组成[36]:

(i) 对求解问题算法程序规约.

(ii) 划分原问题,得到 $\mathrm{In}(T)=\mathrm{In}(T.l)\uparrow[T.d]\uparrow\mathrm{In}(T.r)$.

(iii) 寻找递推关系,得到中序遍历二叉树非递归算法的总策略. 设 3 个变量 X、S、q. 序列变量 X 用于存放已访问的结点标志序列,当有限二叉树遍历结束,则 $X=\mathrm{In}(T)$; S 用于存放尚待遍历 T 的子树的根及其右子树; q 用于存放正准备遍历 T 的子树.

(iv) 基于开发循环不变式的新策略,X、S、q 满足下面的等式,构成所需的循环不变式:

ρ: $\mathrm{In}(T)=X\uparrow\mathrm{In}(q)\uparrow F(S)$.

(v) 依据递推关系和循环不变式,推导出 Apla 算法程序:

```
Procedure inorder(T: btree(char,50); var X: list(char,50));
{
PQ: 给定一个有限的二叉树 T
PR: X = In(T)
}
var S: list(char,50);
var q: btree(char,50);
begin
```

X,S,q: = ▮, ▮,T;
{
ρ: In(T) = X ↑ In(q) ↑ F(S) ;
}
do $q \neq \% \rightarrow q,S$: = $q.l$, [q] ↑ S;
▮ $q = \% \wedge S \neq$ ▮ $\rightarrow q,S,X$: = S [h] . r, S [h + 1. . t] , X ↑ [S [h] . d] ;
od;
end.

基于所得 Apla 算法，使得中序遍历二叉树非递归算法的代码简单实用.

4　形式化证明

用 Dijkstra-Gries 的标准程序证明法证明上述 Apla 算法程序的正确性[20].

(i) 证明 ρ 在循环执行前为真:

语句 S_0 为: X,S,q: = ▮, ▮,T

$Q \Rightarrow wp(S_0,\rho) \equiv$

In(T) ⇒

(X ↑ In(q) ↑ F(S)) $^{X,S,q}_{▮,▮,T} \equiv$

In(T) ⇒ ▮ ↑ In(T) ↑ F(▮) ≡

In(T) ⇒In(T) ≡

true.

(ii) 证明 ρ 为循环不变式:

(a) 证明 ρ 在执行第 1 个循环分支前后为真.

条件 C_1 为 $q \neq \%$

语句 S_1 为 q,S: = $q.l$, [q] ↑ S

$\rho \wedge C_1 \Rightarrow wp(S_1,\rho) \equiv$

In(T) = X ↑ In(q) ↑ F(S) ∧ $q \neq \% \Rightarrow$ (X ↑ In(q) ↑ F(S)) $^{q,S}_{q.l,\ [q]\ \uparrow S} \equiv$

In(T) = X ↑ In(q) ↑ F(S) ∧ $q \neq \% \Rightarrow X$ ↑ In($q.l$) ↑ F([q] ↑ S) ≡

In(T) = X ↑ In(q) ↑ F(S) ∧ $q \neq \% \Rightarrow X$ ↑ In($q.l$) ↑ [$q.d$] ↑ In($q.r$) ↑ F(S) ≡

In(T) = X ↑ In(q) ↑ F(S) ∧ $q \neq \% \Rightarrow X$ ↑ In(q) ↑ F(S) ≡true.

(b) 证明 ρ 在执行第 2 个循环分支前后为真.

条件 C_2 为 $q = \% \wedge S \neq$ ▮

语句 S_2 为 if 语句

$\rho \wedge C_2 \Rightarrow wp(S_2,\rho) \equiv$

In(T) = X ↑ In(q) ↑ F(S) ∧ $q = \% \wedge S \neq$ ▮ ⇒

(X ↑ In(q) ↑ F(S)) $^{q,S,X}_{S\ [q].r,\ S\ [h+1..t],\ X\ \uparrow\ [S\ [q].d]} \equiv$

In(T) = X ↑ In(q) ↑ F(S) ∧ $q = \% \wedge S \neq$ ▮ ⇒ X ↑ [S [h] . d] ↑ In(S [h] . r) ↑ F(S(h + 1. . t)] ≡

In(T) = X ↑ F(S) ∧ $S \neq$ ▮ ⇒ X ↑ F(S) ≡

true.

(iii) 证明后置断言 R 在循环终止时必须为真:

$\rho \urcorner (C_1 \vee C_2) \Rightarrow R \equiv$

In(T) = X ↑ In(q) ↑ F(S) ∧ $q = \% \wedge S =$ ▮ ⇒ In(T) = $X \equiv$

In(T) = X ↑ In(%) ↑ F(▮) ∧ $q = \% \wedge S =$ ▮ ⇒In(T) = $X \equiv$

In(T) = $X \wedge q = \% \wedge S =$ ▮ ⇒In(T) = $X \equiv$

true.

(iv) 循环的终止性成立.

5　PAR 平台自动生成可执行程序

Apla 算法具有集合、序列、树等抽象数据类型，考虑到 C + + 部件库中没有相应的类型与其对应. 因此，本课题组研究团队前期已构造了一个程序转换系统，可以转换一些简单的数据类型和基于简单数据类型的操作，所涉及的语句也十分简单，被称为“核心转换器”[30]**. 正因为如此，才可实现从 Apla 抽象数据类型到 C + + 可重用部件库的转换，以此保证了生成的 C + + 程序的正确性. PAR 平台 C + + 程序自动生成系统总体结构如图 3 所示.**

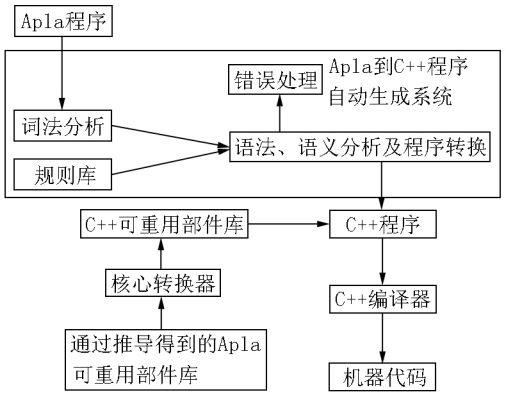

图 3　PAR 平台 C + + 程序自动生成系统总体结构图

通过 PAR 平台 C + + 程序自动生成系统，可将推导并证明的 Apla 算法作为 PAR 平台 C + + 生成系统源语言，自动生成对应的 C + + 程序[31].

在 C + + 程序的自动生成系统中，左边的代码为二叉树排序非递归算法的 Apla 程序，右边为通过 PAR 平台自动生成的C + + 程序. 输入 10 个无序的整数序列，并将其构成二叉排序树，对二叉排序树进行中序遍历，遍历结果为 10 个有序的整数序列(见图 4) .

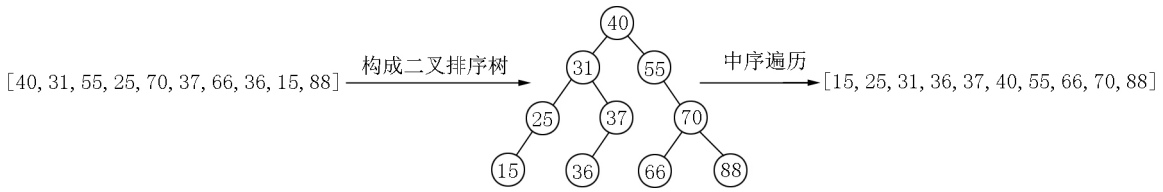

图 4 二叉树排序算法实例

通过验证,自动生成的 C + + 程序正确(见图 5). 将 Apla 程序转换成一个可执行程序,这样就可以通过编译得到运行结果,由于将编译成机器代码的工作交给了第 3 方编译器去做,因而实现了算法程序的机器无关性,转换系统的可靠性也易于得到保证. 通过 Apla 和 C + + 代码比较,发现 Apla 程序明显相应地简单明了,PAR 平台将预定义的组合数据类型(集合、列表、二叉树、图、关系数据和其他预定义的数据类型)以 ADT 的形式进行封装,并且可以用作标准数据类型,因此仅仅有 4 行关键代码,极大地提高算法和程序的开发效率. 并且,PAR 平台集词法分析、语法分析、语义一致性分析、转换、编译、运行为一体,通过严格的测试和静态验证分析,可以完全地确保转换过程中的等价关系没有发生改变就能得到所需要的可执行语言程序. PAR 平台的自动生成工具极大地提高了开发的效率和可靠性.

```
二叉树排序——请输入 10 个整数:
40 31 55 25 70 37 66 36 15 88
排序结果为:
15,25,31,36,37,40,55,66,70,88
```

图 5 自动生成的二叉树排序 C + + 程序的运行结果

6 结束语

本文研究二叉树类非递归算法的推导及形式化证明方法,以二叉树排序非递归算法作为实例,先使用 PAR 方法推导及形式化证明,再用 Dijkstra-Gries 标准程序证明法证明算法的正确性,最后使用 PAR 平台将抽象程序设计语言 Apla 自动生成正确的 C + + 代码. 相比国内外现有研究,本文具有如下特点:

1) 基于划分和递推的算法程序设计方法,简化算法过程得到简单的算法表达式,实现算法程序开发方法的普遍适用性;

2) 基于 PAR 方法开发的循环不变式的新定义和新的开发策略[38-40],通过递归定义技术得到循环不变式,克服了传统的二叉树算法程序不易开发循环不变式的困难;

3) 利用 Apla 提供的预定义 ADT,可以直观地进行程序设计;对“序列”和“二叉树”进行预定义 ADT,相比较其他任何一种解此类问题的可执行语言程序都简洁、清晰易懂,提高算法和程序的开发效率和可靠性,使得形式化推导和证明能快速和方便地进行;

4) 开发了 PAR 平台 C + + 程序自动生成系统,可将推导并通过证明的 Apla 算法程序作为系统源语言,自动生成对应的 C + + 程序;将二叉树排序非递归算法的 Apla 程序自动生成 C + + 程序,极大地提高了开发的效率和可靠性.

PAR 方法是一种实用型的形式化方法[41-43],本文中实例的成功推导及形式化证明对递归问题非递归算法的循环不变式的探讨有一定的参考价值,对非线性数据结构算法程序的推导及形式化证明具有一定的指导意义[44-45].

7 参考文献

[1] Zhu Zhenyuan, Zhu Cheng. Non-recursive implementation of recursive algorithm [J]. Journalof Chinese Computer Systems, 2003, 23(3): 567-570.

[2] Gries D. The science of programming [M]. New York: Springer, 1981.

[3] Dijkstra E W. A discipline of programming [M]. Upper Saddle River: Prentice Hall, 1976.

[4] Xue Jinyun. Two new strategies for developing loop invariants and their applications [J]. Journal of Computer Science and Technology, 1993, 8(3): 147-154.

[5] Xue Jinyun, Zheng Yujun, Hu Qimin, et al. PAR: a practicable formal method and its supporting platform [EB/OL]. [2019-05-17]. https://link.springer.com/chapter/10.1007/978-3-030-02450-5_5.

[6] 左正康,游珍,薛锦云. 后序遍历二叉树非递归算法的推导及形式化证明 [J]. 计算机工程与科学, 2010, 32(3): 119-123.

[7] Gargantini G, Riccobene A E, Scandurra P. Combining formal methods and MDE techniques for model-driven system design and analysis [J]. International Journal on Advances in Software, 2010, 3(1/2): 1-18.

[8] Brooks F P. No silver bullet: essence and accidents of software engineering [J]. IEEEComputer, 1987, 20(4): 10-19.

[9] Floyd R W. Assigning meaning to programs [EB/OL]. [2019-06-12]. https://link.springer.com/chapter/10.1007%2F978-94-011-1793-7_4.

[10] Hoare C A R. An axiomatic basis for computer programming [J]. Communications Ofthe ACM,1969,12(10):117-126.

[11] Owicki S,Gries D. An axiomatic proof technique for parallel programs I [J]. Act a Informatica,1976,6(4):319-340.

[12] Sankaranarayanan S,Sipma H B,Manna Z. Non-linear loop invariant generation using grobner bases [C]//ACM SIGPLAN Principles of Programming Languages. NewYork: ACM Press,2004:318-329.

[13] Colon M,Sankaranarayanan S,Sipma H. Linear invariant generation using non-linearconstraint solving [C]//Computer-Aided Verfication,LNCS 2725. Heidelberg: Springer Berleg,2003:420-433.

[14] Coust P,Coust R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints [C]//ACM Principles of Programming Language. New York: ACMPress,1997:238-252.

[15] Cousot P,Halbwachs N. Automatic discovery of linear restraints among the variables of a program [C]//ACM Principles of Programming Languages. New York: ACM Press,1978:84-97.

[16] 余伟. 基于消元法生成非线性循环不变式 [J]. 电子技术与软件工程,2014,16(8):264-266.

[17] Arora N,Kumar Tamta V,Kumar S. Modified non-recursive algorithm for reconstructing a binary tree [J]. International Journal of Computer Applications,2012,43(10):25-28.

[18] Das V V. A new non-recursive algorithm for reconstructing a binary tree from its traversals [EB/OL]. [2019-05-13]. https://ieeexplore.ieee.org/document/5656782.

[19] Loginov A,Reps T,Sagiv M. Automated verification of the Deutsch-Schorr-Waitetree-traversal algorithm [M]. Berlin: Static Analysis,Springer,2006:261-279.

[20] 胡启敏,薛锦云,游珍,等. PAR 平台中若干软件构件形式化验证技术研究 [J]. 计算机工程与科学,2018,40(2):268-274.

[21] Xue Jinyun. Developing the generic path algorithmic program and its instantiations using PAR method [C]//The Second Asia Workshop on Programming Languages and Systems. Daejeon: APLAS,2001:159-169.

[22] 石海鹤,薛锦云. 基于 PAR 的算法形式化开发 [J]. 计算机学报,2009,32(5):982-991.

[23] 石海鹤,薛锦云. 基于 PAR 的排序算法自动生成研究 [J]. 软件学报,2012,23(9):2248-2260.

[24] Xue Jinyun. PAR: a model driven engineering platform for generating algorithms and software [C]//Programming: Logics,Models,Algorithms and Concurrency to recognize Jayadev Misra´s Accomplishments. Austin: University of Texas Press,2016:150-154.

[25] Xue Jinyun. A unified approach for developing efficient algorithmic programs [J]. Journal of Computer Science and Technology,1997,12(4):314-320.

[26] 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.

[27] Gosling J,Joy W,Steele G,et al. The Java language specification [M]. 3rd Edition. New Jersey: Addison Wesley Press,2005.

[28] You Zhen,Xue Jinyun,Zuo Zhengkang. Unified formal derivation and automatic verificationof three binary-tree traversal non-recursive algorithms [J]. Cluster Computing,2016,19:2145-2156.

[29] 谢武平,薛锦云. Radl 算法到 Apla 程序的生成系统 [J]. 计算机研究与发展,2014,51(4):856-864.

[30] 赖勇. APLA 到 C++自动程序转换系统的研制 [D]. 南昌: 江西师范大学,2002.

[31] Bryant R E,Clarke E M,Grumberg O. Research on automatic verification of finite-state concurrent systems [J]. Annual Review of Computer Science,1987,2(1):269-290.

[32] Yang Qinghong,Li Yunqing. A program development method based on program correctness proof theory [J]. Computer Application Research,2001,18(2):11-13.

[33] Havelund K. Pressbu rger t. Model checking programs using Java path finder [J]. International Journal on Software Tools forTechnology Transfer,2000,2(4):366-381.

[34] Xue Jinyun,Gries D. Developing a line algorithm for cubing a cycle permutation [J]. Science of Computer Programming,1988,11(2):161-165.

[35] 夏鲸. 并发分布式事务处理机制在 PAR 平台中的设计与实现 [D]. 南昌: 江西师范大学,2018.

[36] 游珍. Isabelle 定理证明器的剖析及其在 PAR 方法/PAR 平台中的应用 [D]. 南昌: 江西师范大学,2009.

[37] Sutton A,Maletic J I. Emulating C++0*x* Concepts [J]. Science of Computer Programming,2013,78(9):1449-1469.

[38] Li Mengjun. Formal characterization and verification of loop invariant based on finite difference [EB/OL]. [2012-06-30]. https://cs.nyu.edu/acsys/wing2012/abstracts/menghun.pdf.

[39] Furia C A,Meyer B. Inferring loop invariants using postconditions [M]//Blass A,Dershowitz N,Peisig W. Fields of logic and computation. Berlin,Heidelberg: Springer Verlag,2010:277-300.

[40] 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.

[41] CCF 形式化方法专业委员会. 形式化方法的研究进展

与趋势 [M]. 北京: 机械工业出版社,2018: 1-68.
[42] CCF 软件工程专业委员会. 软件分析: 技术、应用与趋势 [M]. 北京: 机械工业出版社,2016: 56-114.
[43] 张健,张超,玄跻峰,等. 程序分析研究进展 [J]. 软件学报,2019,30(1): 80-109.
[44] 马晓星,刘譞哲,谢冰,等. 软件开法方法发展回顾与展望 [J]. 软件学报,2019,30(1): 3-21.
[45] 王戟,詹乃军,冯新宇,等. 形式化方法概貌 [J]. 软件学报,2019,30(1): 33-61.

The Derivation and Formal Proof of Binary Tree Sorting Non-Recursive Algorithm

ZUO Zhengkang[1],FANG Yue[1],HUANG Qing[1],LIAO Yunyan[1],WANG Yuan[2],WANG Changjing[1*]
(1. College of Computer Information Engineering,Jiangxi Normal University,Nanchang Jiangxi 330022,China;
2. College of Software,Jiangxi Normal University,Nanchang Jiangxi 330022,China)

Abstract: The development of loop invariants for recursive problems of nonlinear data structures are always difficult problems in formal development. In this paper,an approach for the derivation and formal proof of binary tree non-recursive algorithm are researched,and the non-recursive Apla (Abstract Programming Language) algorithm of binary tree sorting algorithm and its exact and simple loop invariant are derived. Then,the correctness of the algorithm is proved by Dijkstra-Gries standard proving technique. In the end,the PAR platform C++ program automatic generation system automatically generates C++ code. The experimental results of the example simplify the derivation and proof of the algorithm program and are useful for the direction for the exploration of loop invariant of non-recursive algorithm for recursive problems,which has guiding significance for the formal proof of algorithm program for nonlinear data structure.

Key words: binary tree class non-recursive algorithm; loop invariant; PAR platform; Dijkstra-Gries standard proving technique; nonlinear data structure

(责任编辑: 冉小晓)

(上接第 624 页)

The Generic Stability of Common Element under Information Constraint of Set-Valued Preference

LU Meihua[1],WENG Xianjie[2]
(1. School of Science,Jiangxi University of Technology,Nanchang Jiangxi 330022,China;
2. Institute of Scientific and Technical Information of Jiangxi Province,Nanchang Jiangxi 330046,China)

Abstract: In order to use the set-valued mapping method to describe preferences,the set-family common element is proposed,and some stability results of the set-family common element under the information constraint mechanism are studied. The equal continuity in the information mechanism is proposed,and generic stability conclusion of the common element under the set-valued preference information constraint is obtained. This research is of great significance,especially when the incomplete information expansion game is equivalently generalized,the strategy chain is more complicated,and the common element of a set family can initially deal with the information mechanism problem under the information constraint.

Key words: information constraint; common element of a set family; generic stability

(责任编辑: 曾剑锋)