文章编号:1001 - 9081(2008)07 - 1854 - 04

非线性循环不变式的自动生成

毕忠勤,曾振柄,郭远华
(华东师范大学 上海市高可信计算重点实验室,上海 200062)
(zqbi@sei.ecnu.edu.cn)

摘 要:提出了一个自动生成非线性循环不变式的算法。循环不变式可以表示成一个带参数的多项式的形式,根据断言的归纳特性,将循环不变式的生成问题转变成一个约束求解问题,这个约束求解问题的每个解对应于一个循环不变式,如果约束求解问题仅有零解,则说明不存在该参数多项式形式的循环不变式。该算法在 Maple 中得到了实现,并通过一些实例说明了该算法的有效性。

关键词:程序验证; 循环不变式; 变迁系统; 约束求解问题
中图分类号: TP311.1 **文献标志码**:A

Automatic generation of non-linear loop invariants

BI Zhong-qin, ZENG Zhen-bing, GUO Yuan-hua
(*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai* 200062, *China*)

Abstract: This paper proposed an approach to automatically generate non-linear loop invariants. An invariant of a loop was hypothesized as a parameterized polynomial. Based on the inductive assertion's properties, we reduced the non-linear loop invariant problem to a numerical constraint solving problem. All the solutions to these constraints were the non-linear loop invariants of the program. The approach has been implemented in Maple. The implementation has been used to automatically discover nontrivial invariants for many programs.

Key words: program verification; loop invariant; transition system; constraint solving problem

0 引言

循环不变式是程序设计理论中的一个重要概念,在程序验证中起着非常重要的作用。程序不变式不仅可以用于分析程序的特性,还可以用于证明循环程序的正确性。

自从基于前、后断言和循环不变式的 Floyd-Hoare-Dijkstra 归纳断言法[1-3]被提出以来,归纳断言就被广泛地用于程序验证。循环不变式是指在循环体的执行前后都为真的断言。一个断言称其为归纳的是指当程序第一次到达该点时该断言为真,并且循环每次执行到这个点时该断言仍然为真。已经证明任何一个归纳断言都是不变式,并且所有求不变式的方法都是基于断言的归纳性的。因此,循环不变式的自动生成在验证程序的正确性中起着越来越重要的作用,并且作为本世纪挑战性的课题之一。

抽象解释[4-5]是循环不变式自动生成中用得最多的一种技术。抽象解释是使用另一个抽象对象域上的计算抽象逼近程序指称的对象域上的计算,使得程序抽象执行的结果能够反映出程序真实运行的部分信息。其产生不变式的主要思想是执行程序的一个抽象对象域上的计算过程直到一个断言不会随着程序的进一步执行而改变,从而得到循环不变式。但是,为了保证计算过程的终止性,在执行过程中引入了 widening/narrowing 推断算子,从而会产生弱的循环不变式。

近年来,随着计算机代数的发展,很多基于符号计算的方法被大量的用于循环不变式的自动生成,例如基于抽象解释、Gröbner 基和量词消去的各种技巧。文献[6]提出 Karr 算法计算一个程序的线性不变式。文献[5]利用抽象解释得到线性不等式作为程序的循环不变式。文献[7]首次证明了多项式形式的不变式具有理想的代数结构,并提出了一个基于不动点计算的循环不变式自动生成的算法,并在 Gröbner 基和消元理论的基础上实现了该算法。文献[8-9]提出了求固定次数的循环多项式不变式的算法,文献[10-11]通过将循环不变式的生成问题转换成数值约束求解问题,得到了一个基于 Gröbner 基的循环不变式自动生成算法。文献[12]基于多项式判别系统对循环不变式的自动生成也进行了有效的尝试。本文在 Sankaranarayanan 工作的基础上,得到了一个更加简单实用的算法,并在 Maple 下实现了该算法。实验结果表明该算法是非常有效的。

1 预备知识

给定实数域 $\mathbf{R}$,$x_1,x_2,\cdots,x_n$(简写为 x)为 n 个属于 $\mathbf{R}$ 的不同符号,称之为变元,对于 n 个非负整数 $\alpha_1,\alpha_2,\cdots,\alpha_n$,称 $x^\alpha = x_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ 为项,又设 $c \in \mathbf{R}$,形如 $cx^\alpha = cx_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ 的表达式称为单项式,其中 c 称为 cx^α 的系数,单项式 $cx^\alpha(c \neq 0)$ 的次数 $\deg(cx^\alpha) = \alpha_1 + \alpha_2 + \cdots + \alpha_n$,并令 $\deg(0) = -\infty$。多项式是单项式的有限和。系数在 $\mathbf{R}$ 中变元为 x 的多项式的集合记为 $\mathbf{R}[x]$。

定义 1 变迁系统。变迁系统 P 是一个五元组 $\langle V,L,l_0,\Theta,\Gamma\rangle$,其中:

1) V 是有限变量集合,某个时刻变量的值称为该时刻变量的状态;

收稿日期:2008 - 01 - 14;**修回日期**:2008 - 03 - 28。 **基金项目**:国家自然科学基金资助项目(90718041);国家 973 计划项目(2004CB318003);国家 863 计划项目(2007AA010302);华东师范大学 2008 年优秀博士生培养基金资助项目(20080029)。

作者简介:毕忠勤(1977 -),男,安徽安庆人,讲师,博士研究生,主要研究方向:程序验证、符号计算; 曾振柄(1963 -),男,甘肃皋兰人,教授,博士生导师,主要研究方向:计算机自动推理、人工智能软件; 郭远华(1978 -),男,湖北天门人,博士研究生,主要研究方向:计算机自动推理、逻辑证明。

2)L 是有限位置集合;

3)l_0 是初始位置;

4)Θ 是在 V 上描述初始状态的断言;

5)Γ 是状态变迁集。Γ 中的元素状态变迁 τ 是一个三元组$\langle l,l',\rho_\tau\rangle$,其中 $l,l'\in L$,分别表示变迁之前和之后的位置,ρ_τ 是一个变迁关系,它是 $V\cup V'$ 上的一个断言,V 表示当前状态变量集合,V' 表示变迁后的状态变量集合。

定义2　不变式。设 $P:\langle V,L,l_0,\Theta,\Gamma\rangle$ 是一个变迁系统。某个位置 $l\in L$ 上的不变式是指对于到达位置 l 的所有状态都为真的断言。一个变迁系统的不变式是指在变迁系统的所有位置都为真的断言。

定义3　代数断言。代数断言是指具有形如 $\wedge_i(p_i(x_1,x_2,\cdots,x_n)=0)$ 的断言,其中 $p_i\in\mathbf{R}[x_1,x_2,\cdots,x_n]$。

定义4　代数变迁系统。一个代数变迁系统是一个变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$,其中:

1)V 是有限变量$\{x_1,x_2,\cdots,x_n\}$ 集合;

2)L 是有限位置集合;

3)l_0 是初始位置;

4)Θ 是变量 V 上的初始代数断言;

5)Γ 是状态变迁集。Γ 中的元素状态变迁 τ 是一个三元组$\langle l,l',\rho_\tau\rangle$,其中 $l,l'\in L$,分别表示变迁之前和之后的位置,ρ_τ 是 $V\cup V'$ 上的一个代数断言。

例 1

同时计算 $GCD(a,b)$ 和 $LCM(a,b)$ 的源程序如下:

$(x,y,u,v):=(a,b,b,0)$

l_0: while true do

$$\left[\begin{array}{c}(x,y,u,v):=(x-y,y,u,u+v)\\ l_1:\quad\text{or}\\ (x,y,u,v):=(x,y-x,u+v,v)\end{array}\right]$$

end while

其对应的代数变迁系统可以表示成如下所示的形式:

$V=\{x,y,z,a,b\}$

$L=\{l_0,l_1\}$

$\Gamma=\{\tau_1,\tau_2\}$

其中:

$\tau_1:\langle l_1,l_0,(x'=x-y\wedge v'=u+v)\rangle$

$\tau_2:\langle l_1,l_0,(y'=y-x\wedge u'=u+v)\rangle$

$l_0=l_0$

$\Theta:(x=a\wedge y=b\wedge u=b\wedge v=0)$

定义5　归纳断言。一个断言 η 是归纳的当且仅当如下的两个条件满足:

初始条件(Initiation):初始位置 l_0 断言成立,即

$\Theta\vdash\eta(l_0)$

承接条件(Consecution):对于每一个状态变迁$\langle l,l',\rho_\tau\rangle$,有

$\eta(l)\wedge\rho_\tau\vdash\eta(l')$

众所周知,任何一个归纳断言都是不变式,并且现如今所有的不变式的生成过程都是利用归纳断言产生的。对于某个代数断言,假设它是某变迁系统的不变式,则要求其满足归纳断言的初始条件和承接条件。

定理1　一个多项式 $P\in\mathbf{R}[x_1,\cdots,x_n]$ 对于所有可能的 $x_1,\cdots,x_n$ 的值都为零当且仅当它的所有系数恒等于零。

对于某代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$,为了构造其不变式,本文先将其不变式假设成无量词的带参数多项式,然后根据归纳断言的初始条件和承接条件,构造出该多项式的参数约束条件,将不变式的生成问题转换成约束求解问题。如果约束求解问题可解,则根据该约束求解问题的解可以构造出原代数变迁系统的不变式,如果约束求解问题仅有零解,即所有的参数都等于 0,则表示原代数变迁系统不存在假设形式的不变式。

2　前人的工作

文献[10]中利用对 Gröbner 基的扩展,得到有关 Template 的扩展 Gröbner 基,通过范式的方法得到具有某种预先假设形式的代数变迁系统不变式。其算法的主要思想是:对于某代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$,首先假设其具有某种形式带参数的不变式 $I(\bar{x},\bar{u})=0$,其中 $\bar{x}$ 是代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$ 中 V 的变量,$\bar{u}$ 是参数。然后根据不变式需要满足归纳断言的初始条件和承接条件构造出有关 $\bar{u}$ 的约束关系。最后求解该约束关系,得到约束关系的解,每个解就对应着原代数变迁系统的一个代数不变式。文献[10]中为了得到不变式 $I(\bar{x},\bar{u})=0$ 满足归纳断言的承接条件时 $\bar{u}$ 需满足的约束关系,建立了如下的定义和定理得到了一个更强的承接关系。

定义6　设 τ 是一个状态变迁 $<l,l',\rho_\tau>$,η 是一个代数断言,定义如下逐渐增强的承接性:

1)η 对于 τ 是多项式倍(PS)承接当且仅当存在一个多项式 p 使得

$\rho_\tau\vdash(\eta(l')-p\cdot\eta(l)=0)$

2)η 对于 τ 是实数倍(CS)承接当且仅当存在一个实数 t 使得

$\rho_\tau\vdash(\eta(l')-t\cdot\eta(l)=0)$

3)η 对于 τ 是等值(CV)承接当且仅当

$\rho_\tau\vdash(\eta(l')-\eta(l)=0)$

4)η 对于 τ 是零承接(LC)当且仅当

$\rho_\tau\vdash(\eta(l')=0)$

由以上的定义可以看出这四种承接关系是逐渐增强的,LC 承接要求变迁后的代数断言不依赖于变迁前的代数断言,CV 要求变迁后的代数断言等于变迁前的断言,是 CS 断言中 $t=1$ 时的特殊情况,而 CS 承接是 PS 承接中多项式 $p=1$ 时的特殊情况。下面的定理给出了任何一个代数断言如果满足 PS 承接则其必然满足归纳断言的承接条件。

定理2　设 τ 是一个变迁$\langle l,l',\rho_\tau\rangle$,$\eta$ 是一个代数断言:

1) 如果 η 对于 τ 满足 CV 承接,则它也必然满足 CS 承接;

2) 如果 η 对于 τ 满足 CS 承接,则它也必然满足 PS 承接;

3) 如果 η 对于 τ 满足 PS 承接,则它也必然满足定义 5 中归纳断言的承接条件。

定理3　设 $G=\{g_1,\cdots,g_m\}$ 是状态变迁关系 Ψ 的 Gröbner 基,p_1,p_2 是多项式。令 $p'_i=NF_G(p_i)$,$(i=1,2)$,$\lambda\in\mathbf{R}$。如果 $p'_1-\lambda p'_2=0$,则 $\Psi\vdash(p_1-\lambda p_2=0)$。

根据定理 3,文献[10]给出了得到不变式满足归纳断言承接条件时有关 $\bar{u}$ 约束条件的算法:

1) 令 $f=NF_G(\eta(l_i))$,$g=NF_G(\eta(l_j)')$,其中 η 是一个代数断言,G 是某个状态变迁 ρ_τ 的 Gröbner 基,$\eta(l_i)$,$\eta(l_j)'$ 分别表示状态变迁 ρ_τ 前后的代数断言 η 对应的形式;

2) 引入一个 $\lambda\in\mathbf{R}$,计算 $\lambda f-g$,并令 $\lambda f-g$ 中有关 $\bar{x}$ 的各项的系数都等于 0,从而得到了一个有关 λ 和 $\bar{u}$ 的约束关系。

3)求解以上的约束关系,就可以得到原代数变迁系统的不变式。

3　算法描述

众所周知,计算 Gröbner 基的 Buchberger 算法的复杂度理论上是双指数的,效率比较低。本文在文献[10]的基础上进行了简化,得到了一个简单实用的算法,并通过若干实例验证

了程序的有效性。算法主要过程如下：

1)定义代数变迁系统的带参数不变式；

2)根据不变式必须满足归纳断言的初始条件和承接条件构造关于参数的约束关系；

3)求解约束关系得到原代数变迁系统的不变式。

下面通过一个实例详细介绍算法的主要过程。

3.1 定义不变式

对于某代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$，首先假设其具有某种形式带参数的不变式$I(\bar{x},\bar{u})=0$，其中$\bar{x}$是代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$中$V$的变量，$\bar{u}$是参数。

例2 对于例1的代数变迁系统，假设其存在一个不变式$I(\bar{x},\bar{u})$是V上次数不超过2的多项式，可以表示如下：

$$I(\bar{x},\bar{u})=a_0x^2+a_1x+a_2xy+a_3xu+a_4xv+a_5xa+a_6xb+a_7y^2+a_8y+a_9yu+a_{10}yv+a_{11}ya+a_{12}yb+a_{13}u^2+a_{14}u+a_{15}uv+a_{16}ua+a_{17}ub+a_{18}v^2+a_{19}v+a_{20}va+a_{21}vb+a_{22}a^2+a_{23}a+a_{24}ab+a_{25}b^2+a_{26}b+a_{27}$$

其中：$\bar{x}=\{x,y,u,v,a,b\}$，$\bar{u}=\{a_0,a_1,\cdots,a_{27}\}$。

3.2 构造约束关系

下面介绍根据不变式需要满足归纳断言的初始条件和承接条件构造$\bar{u}$的约束关系的过程。

3.2.1 初始条件

为了使得$\Theta\models(I(\bar{x},\bar{u})=0)$，将$\Theta$中初始断言的值代入$I(\bar{x},\bar{u})$将生成一个带参数的多项式$P$，令$P$中所有基于$V$的项的系数都为0，则得到一组约束关系。

例3 对于例1的代数变迁系统及例2中的不变式，进行以上运算得到如下的多项式P：

$$P=(a_0+a_{22}+a_5)a^2+(a_{16}+a_3+a_{24}+a_{11}+a_2+a_6)ab+(a_1+a_{23})a+(a_9+a_7+a_{12}+a_{25}+a_{13}+a_{17})b^2+(a_{14}+a_8+a_{26})b+a_{27}$$

令其所有项的系数等于0，得到如下的关于$\bar{u}$的约束关系1：

$$(\exists\bar{u})[(a_0+a_{22}+a_5)=0\wedge(a_{16}+a_3+a_{24}+a_{11}+a_2+a_6=0)\wedge(a_1+a_{23}=0)\wedge(a_9+a_7+a_{12}+a_{25}+a_{13}+a_{17}=0)\wedge(a_{14}+a_8+a_{26}=0)\wedge(a_{27}=0)]$$

3.2.2 承接条件

为了保证假设的不变式$I(\bar{x},\bar{u})=0$满足归纳断言的承接特性，即对每个状态变迁$\langle l,l',\rho_\tau\rangle$，有$(I(\bar{x},\bar{u})=0)\wedge\rho_\tau\models(I(\bar{x}',\bar{u})=0)$。首先需要列出该代数变迁系统中所有的状态变迁，然后对于每个状态变迁$\langle l,l',\rho_\tau\rangle$，利用$(I(\bar{x},\bar{u})=0)\wedge\rho_\tau\models(I(\bar{x}',\bar{u})=0)$确定其参数的约束关系。

根据定理2的第三条，对于某个代数变迁系统$\langle V,L,l_0,\Theta,\Gamma\rangle$，为了使得$(I(\bar{x},\bar{u})=0)\wedge\rho_\tau\models(I(\bar{x}',\bar{u})=0)$成立，只需要保证$(\exists t)(I(\bar{x}',\bar{u})-t\cdot I(\bar{x},\bar{u})=0)$成立。

为了得到原代数变迁系统的不变式的约束关系，考虑代数变迁系统的每个变迁使得它满足以上的公式，得出一个含有t的关于$\bar{u}$的约束关系。

例4 对于例1的代数变迁系统以及例2的不变式，首先考虑状态变迁τ_1，计算$I(\bar{x}',\bar{u})-t\cdot I(\bar{x},\bar{u})$得到如下的多项式：

$$(-ta_{22}+a_{22})a^2+(a_5-ta_5)ax+(-ta_{11}-a_5+a_{11})\cdot ay+(-ta_{16}+a_{16}+a_{20})au+(a_{24}-ta_{24})ab+(a_{20}-ta_{20})av+(a_{23}-ta_{23})a+(a_0-ta_0)x^2+(a_2-2a_0-ta_2)xy+(-ta_3+a_4+a_3)xu+(a_6-ta_6)xb+(a_4-ta_4)xv+(-ta_1+a_1)x+(-ta_7+a_7-a_2+a_0)y^2+(a_9-a_3-ta_9-a_4+a_{10})yu+(-a_6-ta_{12}+a_{12})yb+(-a_4+a_{10}-ta_{10})yv+(-ta_8+a_8-a_1)y+(-ta_{13}+a_{13}+a_{15}+a_{18})u^2+(-ta_{17}+a_{17}+a_{21})ub+(a_{15}-ta_{15}+2a_{18})uv+(a_{19}+a_{14}-ta_{14})u+(a_{25}-ta_{25})b^2+(a_{21}-ta_{21})bv+(a_{26}-ta_{26})b+(a_{18}-ta_{18})v^2+(a_{19}-ta_{19})v+(a_{27}-ta_{27})$$

根据零多项式定理，得到如下的关于$t,\bar{u}$的约束关系2：

$$(\exists t,\bar{u})[(a_0-ta_0=0)\wedge(-ta_1+a_1=0)\wedge(a_2-2a_0-ta_2=0)\wedge(-ta_3+a_4+a_3=0)\wedge(a_4-ta_4=0)\wedge(a_5-ta_5=0)\wedge(a_6-ta_6=0)\wedge(-ta_7+a_7-a_2+a_0=0)\wedge(-ta_8+a_8-a_1=0)\wedge(a_9-a_3-ta_9-a_4+a_{10}=0)\wedge(-a_4+a_{10}-ta_{10}=0)\wedge(-ta_{11}-a_5+a_{11}=0)\wedge(-a_6-ta_{12}+a_{12}=0)\wedge(-ta_{13}+a_{13}+a_{15}+a_{18}=0)\wedge(a_{19}+a_{14}-ta_{14}=0)\wedge(a_{15}-ta_{15}+2a_{18}=0)\wedge(-ta_{16}+a_{16}+a_{20}=0)\wedge(-ta_{17}+a_{17}+a_{21}=0)\wedge(a_{18}-ta_{18}=0)\wedge(a_{19}-ta_{19}=0)\wedge(a_{20}-ta_{20}=0)\wedge(a_{21}-ta_{21}=0)\wedge(-ta_{22}+a_{22}=0)\wedge(a_{23}-ta_{23}=0)\wedge(a_{24}-ta_{24}=0)\wedge(a_{25}-ta_{25}=0)\wedge(a_{26}-ta_{26}=0)\wedge(a_{27}-ta_{27}=0)]$$

对于状态变迁τ_2，依据上面的方法同样可以得到如下的关于$r,\bar{u}$的约束关系3：

$$(\exists r,\bar{u})[(-a_2+a_0+a_7-ra_0=0)\wedge(-ra_1-a_8+a_1=0)\wedge(-ra_2+a_2-2a_7=0)\wedge(-a_9+a_3-ra_3=0)\wedge(a_3-a_{10}-ra_4-a_9+a_4=0)\wedge(-ra_5+a_5-a_{11}=0)\wedge(a_6-a_{12}-ra_6=0)\wedge(a_7-ra_7=0)\wedge(a_8-ra_8=0)\wedge(-ra_9+a_9=0)\wedge(a_{10}+a_9-ra_{10}=0)\wedge(-ra_{11}+a_{11}=0)\wedge(-ra_{12}+a_{12}=0)\wedge(a_{13}-ra_{13}=0)\wedge(a_{14}-ra_{14}=0)\wedge(2a_{13}+a_{15}-ra_{15}=0)\wedge(-ra_{16}+a_{16}=0)\wedge(a_{17}-ra_{17}=0)\wedge(a_{15}+a_{18}+a_{13}-ra_{18}=0)\wedge(a_{14}+a_{19}-ra_{19}=0)\wedge(a_{20}-ra_{20}+a_{16}=0)\wedge(-ra_{21}+a_{17}+a_{21}=0)\wedge(-ra_{22}+a_{22}=0)\wedge(a_{23}-ra_{23}=0)\wedge(-ra_{24}+a_{24}=0)\wedge(a_{25}-ra_{25}=0)\wedge(a_{26}-ra_{26}=0)\wedge(a_{27}-ra_{27}=0)]$$

这样就将不变式的自动生成问题转化成了求解上述约束关系1 ∧ 约束关系2 ∧ 约束关系3的问题。

3.3 求解约束

以上生成的约束问题可以通过消去量词的方法进行计算，约束问题的解就对应于原代数变迁系统的不变式。

1951年A. Tarski[15]在其论文《初等代数与初等几何的判定方法》首次提出了量词消去方面的构想。1975年G. E. Collins[16]提出了较为实用的量词消去算法——柱形代数分解算法(CAD)。后人在他的工作的基础上作出了一些改进。近年来，Hong等人[17]在CAD的基础上提出了PCAD的算法，并且在其软件QEPCAD中实现了这个算法。在国内，杨路等人开发的DISCOVERER软件包中也实现了PCAD算法。本文中，主要是通过逐步将非线性约束分解转换成线性约束以及根据冲突将一些非线性约束去掉的方法求解。下面以第二节的例子来说明这个过程。

例5 对于例4中的约速关系2，首先通过分解$a_0-ta_0=(1-t)a_0=0$，得到$t=1$或$a_0=0,1-t\neq0$，对于第一种情况，将$t=1$代入原约束条件，可得到：

$$a_{\{0,1,2,4,5,6,9,15,18,20,21\}}=0$$

$$a_{10}-a_3=0$$

对于第二种情况，可以得到$\bar{u}$中的所有元素都为0。

对于约束关系3，我们同样可以得到，当$r=1$时，

$$a_{\{2,7,8,9,11,12,13,15,16,17,19\}}=0$$

$$a_3-a_{10}+a_4=0$$

综合约束关系1 ∧ 约束关系2 ∧ 约束关系3，可以得到如下的解：

$$\begin{cases} a_{\{0,1,2,4,\cdots,9,11,\cdots,23,25,26,27\}} = 0 \\ a_{10} - a_3 = 0 \\ a_{24} + a_3 = 0 \end{cases}$$

从而可以得到

$I(\bar{x},\bar{u}) = a_3(xu + yv - ab)$

即得到原代数变迁系统的不变式为:

$xu + yv - ab = 0$

4　实验结果

本算法以 Maple 11 为开发平台,在 P4 2.8 GHz,1 GB 内存的微机上得到了实现,并对一些例子进行了测试,得到了这些程序的循环不变式,验证了该算法的有效性。

例 6　下列循环语句是来源于[14] 的一个计算 $\sum y^5$ 的程序

```
(x,y) := (0,0)
while ? do
  (x,y) := (x + y^5, y + 1)
end while
```

通过运行程序,可以得到关于本程序的一个 6 次循环不变式:

$y^2 + 12x - 5y^4 - 2y^6 + 6y^5 = 0$

例 7　下列语句是一个具有多个程序变量的分解数 N 的程序块[13]

```
(r,x,y) := (R^2 - N, 2R + 1, 1);
while ? do
  (r,y) := (r - y, y + 2);
or
  (r,x) := (r + x, x + 2);
end while
```

通过运行程序,可以得到关于本程序的一个 2 次循环不变式

$4r - x^2 + 2x + y^2 - 2y + 4N = 0$

表 1 列出了本程序在一系列测试用例上所花费的时间以及得到的循环不变式的次数。(其中:Function 表示程序功能,Source 表示程序来源,NV 表示变元个数,NB 表示分支个数,DI 表示生成的不变式的次数,T 表示时间,来源为(*)的是作者为测试程序自己构造的变迁系统)

表 1　测试结果

Function	Source	NV	NB	DI	T/s
LCM&GCD	[10]	6	2	2	0.125
$a \times b$	[10]	4	1	2	0.064
Count	[10]	6	4	2	0.078
$\sum x$	[14]	2	1	1	0.031
$\sum x^2$	[14]	2	1	2	0.031
$\sum x^3$	[14]	2	1	3	0.031
$\sum x^4$	[14]	2	1	4	0.062
$\sum x^5$	[14]	2	1	6	0.125
LCM	[1]	6	2	2	0.047
Factor	[13]	5	2	2	0.141
Undef	(*)	4	2	2	0.062
Undef	(*)	4	2	1	0.032

从表 1 可以看出,该算法不论对于变元个数比较多的复杂系统,还是对于分支比较多的系统,都能够很好的得到原代数变迁系统的不变式。

5　结语

本文通过代数变迁系统的循环不变式必然满足归纳断言的初始条件和承接条件的特点,将循环不变式的自动生成问题转化成求解约束问题,然后由约束问题的解得到原代数变迁系统的循环不变式,并用 Maple 实现了该算法,实验结果显示该算法非常有效。但对于该算法中的有关问题还有待进一步的研究,特别是如何得到代数变迁系统循环不变式的多项式的最高次,从而能够进一步把该算法完备化,这也将是我们下一步的工作,另外对于如何自动生成多项式不等式作为循环不变式也是一个值得研究的问题,因为有些代数变迁系统本身就不存在多项式等式形式的循环不变式。

参考文献:

[1] DIJKSTRA E W. A discipline of programming[M]. New Jersey: Prentice-Hall Inc., 1976.

[2] FLOYD R W. Assigning meanings to programs[C]// Proceedings of Symposia in Applied Mathematics. [S.l.]: American Mathematical Society, 1967, 19: 19 - 32.

[3] HOARE C A R. An axiomatic basis for computer programming[J]. Communications of ACM, 1969, 12(10): 576 - 580.

[4] 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 Languages. New York: ACM Press, 1977: 238 - 252.

[5] 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.

[6] KARR M. Affine relationships among variables of a program[J]. Acta Information, 1976(6): 133 - 151.

[7] RODRIGUEZ-CARBONELL E, KAPUR D. Generating all polynomial invariants in simple loops[J]. Journal of Symbolic Computation, 2007, 42(4): 443 - 476.

[8] MULLER-OLM M, SEIDL H. Computing polynomial program invariants[J]. Information Processing Letters, 2004, 91(5): 233 - 244.

[9] MULLER - OLM M, SEIDL H. Precise interprocedural analysis through linear algebra[C]// ACM SIGPLAN Principles of Programming Languages. New York: ACM Press, 2004: 330 - 341.

[10] SANKARANARAYANAN S, SIPMA H B, MANNA Z. Non-linear loop invariant generation using Gröbner bases[C]// ACM SIGPLAN Principles of Programming Languages. New York: ACM Press, 2004: 318 - 329.

[11] COLON M, SANKARANARAYANAN S, SIPMA H. Linear invariant generation using non-linear constraint solving[C]// CAV 2003: Computer-Aided Verification, LNCS 2725. Heidelberg: Springer Verleg, 2003: 420 - 433.

[12] CHEN YING-HUA, XIA BI-CAN, YANG LU, *et al*. Generating polynomial invariants with discoverer and qepcad[C]// JONES C B, LIU Z, WOODCOCK J, eds. LNCS 4700. Heidelberg: Spring Verleg, 2007: 67 - 82.

[13] KNUTH D E. The art of computer programming: volume 2, seminumerical algorithms[M]. Mass: Addison Wesley, 1969.

[14] PETTER M. Berechnung von polynomiellen invarianten [D]. Fakultät für Informatik, Technische Universität München, 2004.

[15] TARSKI A. A decision method for elementary algebra and geometry [M]. Berkeley: University of California Press, 1951.

[16] COLLINS G E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition[C]// BRAKHAGE H, ed. Automata Theory and Formal Languages, LNCS 33. Heidelberg: Springer Verleg, 1975: 134 - 165.

[17] COLLINS G E, HONG H. Partial cylindrical algebraic decomposition for quantifier elimination[J]. Journal of Symbolic Computation, 1991, 12(3): 299 - 328.