文章编号:1673-0062(2016)04-0106-06

一种基于 GEP 的程序不变量动态发现方法

郭培甲,吴取劲*

(南华大学 计算机科学与技术学院,湖南 衡阳 421001)

摘　要:利用 GEP(基因表达式编程)获取最优解的函数发现能力和数学理论的精确计算能力获得较客观的程序不变量的预置形式,能够进行目前技术没有处理的函数型程序不变量的发现生成工作.在函数型程序不变量范围内,有效地克服了目前程序不变量发现技术中存在的两个缺点,即计算盲目性与程序不变量形式预置的主观性.整个工作可以理解为是对目前程序不变量发现技术的一个扩展,它拓广了原有技术可发现的程序不变量种类,增大了从程序运行轨迹数据中发现更多不变量的可能性.

关键词:程序不变量;GEP 函数型程序不变量

中图分类号:TP311　　**文献标志码**:B

A Method of Detecting Likely Program Invariants Based on GEP

GUO Pei-jia, WU Qu-jing*

(School of Computer Science and Technology, University of South China, Hengyang, Hunan 421001, China)

Abstract: It introduces a new algorithm based on GEP which can find Functional Program Invariants from data formed by observing program executing. In the field of detecting program invariants, the algorithm improves possibility of mining more invariants which can not be found by the technology in existence and can get successfully polynomial functional invariants.

key words: program invariants; GEP; functional program invariants

0　引　言

程序不变量(Invariants)是指软件程序运行时保持不变性质的逻辑断言.严格来讲,在动态分析方法中程序不变量应该称为似然程序不变量,它描述的是在程序的多次运行中保持不变的性质.由于采集的数据总是有限的,所以似然程序不变量也许是真正的不变量,也许是伪不变量.动态发现程序不变量研究的主要目的是通过观察和分析程序运行数据获取程序不变量,提供给程序员作为提高程序质量的重要依据.

通过程序的实际运行轨迹发现其不变性质的

收稿日期:2016-10-10

作者简介:郭培甲(1994-),男,宁夏中卫人,南华大学计算机科学与技术学院.主要研究方向:软件工程. * 通讯作者.

动态分析方法近年来得到了长足的发展[1-4],其中影响最大的是 Ernst 等人的系列研究工作.Daikon 是 Ernst 等开发的一个动态发现程序不变量的工具,其基本思想是依据专家主观预置不变量集合并对该集合进行违例排除,它的不变量都是关于数据行为合约的断言,每个断言中变量个数限制在 3 个以下(含 3 个).Daikon 主要检查单个变量是否恒为常量、是否恒不为 0、是否恒满足某个函数(如奇数 odd、求模 mod 等)、是否局限在较小的集合或区间内等,检查两个或三个变量之间是否有相等关系、不等关系、次序关系、线性关系等[5-8].分析表明目前的似然程序不变量动态发现方法比较粗糙,只能检测一些直观的来自于程序员经验所及的不变量形式,这难免会遗漏一些重要的程序特征.主要体现在两个方面,一个是检验遍历的盲目性,即针对所有预置似然不变量进行全方位的遍历测试,其计算量很大且存在大量的无用计算;另一个方面是预置似然不变量形式为人工预测,具有较大的主观性.因此需要寻找可行的方法尽可能地从程序运行轨迹数据中进行客观地分析,发现出程序中的各种不变性质,减少因遗漏造成的不良影响,同时减少盲目性造成的大量无用计算.

1　基于 GEP 的程序不变量动态发现方法核心思想

程序不变量的形式多种多样,因此在研究时需要将程序不变量进行分类,降低复杂程度以便于分别进行分析处理.总体上讲,可以将程序不变量划分成函数依赖型与非函数依赖型两种类别进行研究,其中函数依赖关系是数学研究的一个经典领域,有很多的理论、方法可以利用,本文讨论的内容集中在这一领域.

对于函数型程序不变量表达式的获取过程,由于工作的基础在于采集到的程序运行轨迹数据,能够理解为是一个依据实验数据进行建模的过程,因而可以在实验建模理论的指导下进行分析.当然,如果能够获得一定数学理论的指导,那肯定可以降低计算的复杂度,减少建模的工作量,可是在实验数据中所反映的函数类型事先是不清楚的,这时候只能够让计算机通过计算和比较,从庞大的数据点集合中寻找可能存在的函数表达式,基因表达式编程(GEP)方法比较好地体现了这种建模思想,它利用遗传算法进行迭代计算,获取误差最小的函数表达式,也就是所谓最优解,因而为我们提供了一条解决问题的途径,即最优解有能力反映真实解的函数类型.但必须强调的是程序运行轨迹数据具有的一个重要特点:采集过程的零噪声,它要求获得的最终结果应该是一个精确解,这也是程序不变量的特点,Ernst 在他的相关论文对此有详细的论述,即程序不变量不能容纳任何违例[3].

基于上述分析,对于函数型程序不变量发现工作可以进一步划分成三个步骤进行:1)函数型不变量存在性判定,即利用数学关系数据理论明确哪些变量之间存在函数依赖关系,从而降低后续工作的盲目性,提高发现效率;2)利用存在性判定结果使用 GEP 进行程序不变量形式的发现,这里又分为两部分工作:函数依赖型程序不变量和非函数依赖型程序不变量的形式启发信息的获取.依据获得的最接近数据内部真实规律的类型信息和相关的参数,形成较客观的程序不变量的预置形式信息;3)利用上一步获得的具有启发意义的信息,即函数的基本类型和相关的参数,寻找可能的数学理论指导进行函数表达式的精确确定.

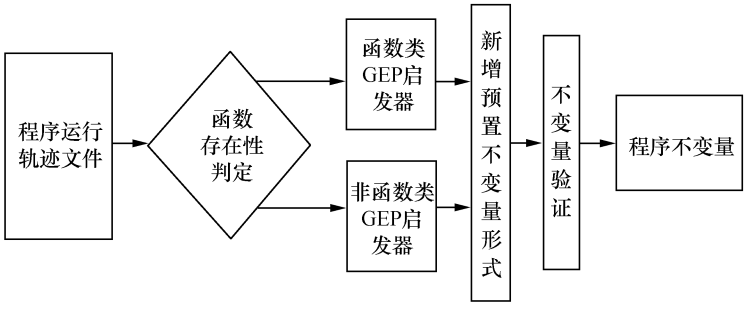

图 1　基于 GEP 的程序不变量动态发现方法框架图

Fig.1　Framework of dynamic detecting method of program invariants based on GEP

2 函数型不变量的存在性判定

函数依赖在关系数据库中是一种通用约束形式.函数依赖一般被用于在数据库设计的时候表达关系模式上各属性之间的语义约束,被用来作为一种约束指导数据库设计的概念与细节.

函数依赖定义:设 $R(U)$ 是一个属性集 U 上的关系模式,X 和 Y 是 U 的子集.若对于 $R(U)$ 的任意一个可能的关系 r,r 中不可能存在两个元组在 X 上的属性值相等,而在 Y 上的属性值不等,则称"X 函数确定 Y"或"Y 函数依赖于 X",记作 $X\rightarrow Y$.

进行函数依赖不变量的存在性判定工作的主要目的是降低函数形式推导时的盲目性.比如设某程序观测变量有 N 个,分别是 $X_1,X_2,\cdots,X_N$,但只存在一个二元函数不变量形式 $X_1=X_2+2$,若依据函数依赖定义检测已经确定 X_1 和 X_2 之间存在函数依赖关系,则后续的函数表达式确定工作只需要在 X_1 和 X_2 之间进行,而不需要检测 C_N^2 组数据.这样就大大提高了后继工作的效率.文献[9]中有详细的实验数据支持这一方法对于整体工作效率的提高具有优越性的结论.

3 发现算法

算法的任务是为了从程序运行轨迹数据中发现程序不变量启发信息进而确定程序不变量形式.要解决的问题包括如何获取程序运行轨迹数据、程序运行轨迹数据以何种形式存放、区分出非函数型和函数型似然程序不变量、确定函数型似然程序不变量的表达形式.

因此完整的算法完成的工作应该是进行程序的编配、运行编配后程序提取程序运行轨迹并存入数据库中、利用关系数据理论明确程序运行轨迹数据中哪些变量之间存在函数依赖关系、形成函数型和非函数型程序不变量两个处理分支(对后者留下处理接口)、在存在函数关系的变量集合中使用 GEP 进行函数挖掘获得最接近数据内部真实规律的函数的类型和相关的参数、寻找可能的数学理论指导进行函数表达式的精确确定、使用数据库 SQL 语言的 SELECT 进行最终结果的验证.

3.1 算子定义问题

为了从某个应用领域中的软件程序内发现不变量,在应用领域中有很多特殊规律经过前人的总结已经成为业界的共识,在算法设计过程中应该给出机制,使得算法的用户能够根据自己的需要自由地选择基本算子.

基本算子可以划分成三类:算符算子、标准函数算子和自定义算子.

算符算子:{+,-,*,/,@,mod}//@ 为取负

标准函数算子:{x,1/x,sin(x),cos(x),tan(x),ex,log(x),ln(x),abs(x),sqrt(x)……}.

自定义算子:提出的目的是为了满足在特定应用领域中的特殊要求,允许用户根据自己的需要设定算子,这一机制将增大算法的灵活性.

算法设计中,算符算子、标准函数算子和自定义算子共同形成算子库,在算法程序运行过程前除了提供默认算子外,还应提供机制允许用户选择程序执行时使用的算子.

3.2 SQL 语句池

本文设计的算法是将程序运行轨迹数据存放于数据库中的,利用了关系数据理论进行了函数依赖关系存在性判定工作.SQL 语言作为关系数据库中最普遍使用的语言,它的查询功能十分强大,这一点可以充分地用来进行函数关系的验证工作.比如,对于函数 $y=f(x)$,执行

SELECT y FORM 轨迹数据库中(x,y) WHERE y=f(x);

若选择结果为全集,即轨迹数据中不存在 $y=f(x)$ 的违例,因而可以认为 $y=f(x)$ 是一个似然不变量.

在算法设计中,采用的是开放性质的 SQL 语句池机制.可以事先定义好典型性质的不变量的检测 SQL 语句存储在语句池中,也可以根据需要随时构造 SQL 语句并存入语句池中.这一方法除了可以用在函数型程序不变量的检测上,同样可以用在非函数型程序不变量的检测上,这一机制为我们将两大类型的程序不变量发现工作进行整合提供了一条有效的途径.

3.3 算法流程

算法 1:基于 GEP 的程序不变量发现算法总程序

输入:分析对象程序;

输出:程序不变量.

Begin

初始化算子库和 SQL 语句池;

获得运行轨迹数据文件;

对运行轨迹数据文件进行格式转换,得到轨迹文件数据库数据;

依据函数依赖定义检测获得函数依赖

```
集合;
    If 函数依赖集合=
    Then
      {
        转非函数型程序不变量处理程序接口;
        处理完毕,算法结束
      }
    Else
      {
        用户从算子库中选择基本算子或使用程序自动选取的默认算子;
        调用 GEP 算法获得函数类型等启发性信息;
        Switch 依据启发性信息确定存在的函数的类别
        {
        Case 为多项式函数:
          {
            提取启发信息中多项式的最高幂次 n;
            确定多项式形式 y=a_n x^n+a_{n-1} x^{n-1}+…+a_1 x+a_0;
            从轨迹数据中提取 n+1 组数据,使用待定系数法解联立方程组确定系数 an,an-1,…,a1,a0;
            将确定的多项式函数 y=f(x)填入 SQL 语句池中;
            执行 SELECT y FORM 轨迹数据库 WHERE y=f(x);
            If 选择结果为全集
            Then output y=f(x)
            Else output y=f(x)不成立      //这里可以考虑转换成条件不变量
            Fi
            算法结束
          }
        Case 为三角函数:{略}
        Case 为指数函数:{略}
          …
        Default :
            {非多项式的函数型程序不变量接口;
              //需要根据不同的类型采取不同的处理方法;
                算法结束}
        }
        End Switch
      }
    Fi
End
```

4　启发信息的获取实验

在本文的算法中,函数表达式启发信息指的是函数的类型信息以及相关参数信息.对于原始数据集中蕴含的不变量形式而言,进行挖掘之前是不知道其类型的,为此这里选择了几个不同形式的函数作为实验的测试函数,使用 GEP 原理进行函数表达式的发现.

4.1　标准多项式型函数的实验数据

传统的 GEP 对最高次数较高的标准多项式型函数的发现率不高,好在除了传统的 GEP 方法外,目前已有一些改良的 GEP 算法可以提高发现的成功率,如对于多项式函数,文献[9]中提出并实现了一种基于 GEP 的并列函数关系挖掘算法 PPM(Parallel Polynomial Mining),其核心思想是文献[10]中提出的重叠基因表达,通过实验证明基于此思想的多基因进化技术在速度上是单基因 GEP 算法的 2.5~9.4 倍,在高次函数发现的成功率方面提高至少一个数量级.

对于多项式程序不变量,传统的多项式挖掘 PM 主要是针对现有的数据以单一的多项式进行匹配,而 PPM 则寻求多个并列函数因子的乘积 $f_1(x)\times f_2(x)\times\cdots\times f_n(x)$ 来进行匹配.作为其特例,当 $f(x)$ 和每个 $f_i(x)$ 都是多项式函数时,这一方法实现了用多个低阶多项式来拟合一个具有较高阶的多项式.

表 1 是文献[10]中使用 PPM 算法实现的部分实验数据,从表 1 中分析实验 1 至实验 3 的结果可以获得两个方面的信息,一方面是 GEP 具有能力从实验数据中获得其内部蕴含的函数关系的基本类型信息,另一方面是获得的结果表达式并不是一个完全精确的结果,根据程序不变量的定义不能以 GEP 给我们的结果作为最终的输出.

其中实验 1 的结果是精确的,实验 2 与实验 3 存在误差,但是这三个实验均给出了重要的启发信息,即多项式函数的最高次数.

表 1 PPM 算法实现的部分实验数据

Table 1 The partial data originating from PPM algorithm experiment

实验序号	测试表达式	结果表达式集
1	$Y=2x^6-x^5-5x^4+8x^3+1$	$2.00x^3+3.00x^2-1.00x+1.00$; $1.00x^3-2.00x^2+1.00x+1.00$
2	$Y=12x^7+2x^6+27x^5+52x^4+24x^3+70x^2+39x+5$	$2.99x^2-1.00x+5.00$; $4.00x^2+6.01x+0.98$; $1.00x^3-1.00x^2+1.99x+1.00$
3	$Y=2x^{10}+x^9-x^8+18x^7-46x^6+87x^5-58x^4+79x^3-90x^2+34x-24$	$1.00x^2-2.01x+2.00$; $1.00x^3-0.99x^2+3.00x-4.00$; $2.00x^2+0.98x+2.99$; $1.02x^3+3.00x^2+1.00$

上述内容是应用 GEP 原理及技术获得的基于误差可信度量的函数表达式.从结果中可以看出存在有一定误差值,而对于程序不变量来说,存在误差意味着存在违例,也即程序不变量不成立.尽管如此,上述的工作还是得到了非常重要的启发性信息,即函数的类型信息为标准多项式以及多项式函数的最高阶.可以据此在成熟的数学理论中选取拉格朗日插值方法或待定系数方法直接得到唯一的准确的多项式函数表达式形式.

至此,对于程序不变量中标准多项式函数型程序不变量这一类程序中重要的不变性质可以使用本文的方法完全精确地获得.相对于 Daikon 工具,本文的函数型程序不变量的发现方法获得了 Daikon 没有处理的函数型程序不变量(Daikon 只能处理线性多项式形式),整个工作可以理解为是对目前程序不变量发现技术的一个扩展,增大了从程序运行轨迹数据中发现更多不变量的可能性.

4.2 扩展多项式型函数

多项式函数形式定义为 $y=P(f(x))$,$f(x)$ 为标准函数算子或自定义算子,当 $f(x)\neq x$ 时,$y=P(f(x))$ 成为非标准的扩展多项式.对于 $y=P(f(x))$ 上的处理在本质上与 $y=P(x)$ 上的处理是相同的.在将来的工作中将寻求实验结论给予支持.

4.3 其它类型函数的实验

实验环境为:Intel(R)4CPU,1.80 GHz;APS3.0;采样数均为 2 000,取值为 10 000 以内的随机数.表 2 为基于 GEP 的非多项式型函数表达式发现的实验结果数据.

表 2 的实验结果给出的信息也是两个方面的信息,一方面是 GEP 具有能力从实验数据中获得其内部蕴含的函数关系的基本类型信息,另一方面是获得的结果表达式并不都是完全精确的结果.

表 2 基于 GEP 的非多项式型函数表达式发现的实验结果

Table 2 Results of detecting non polynomial function expressions based on GEP

实验一	测试表达式	$z=x^3+2xy+y^2$
	结果表达式	$z=x^3+2xy+y^2$
	可信度	1 000
	用时/s	11
实验二	测试表达式	$Y=2x^2+\mathrm{sqrt}(x)+6$
	结果表达式	$Y=2x^2+sqrt(x-(0.024\ 323\ 580\ 699\ 795\ 254-sqrt(x)/9.406\ 067))+5.947\ 205$
	可信度	999.999 999 762 129
	用时/s	28
实验三	测试表达式	$Y=x+\log(x)$
	结果表达式	$Y=x+\log(x)+\cos(10.967\ 713)/(14.482\ 27*(x+9.487\ 488)$
	可信度	999.999 934 562 546
	用时/s	26
实验四	测试表达式	$Y=\sin(2x+5)$
	结果表达式	$Y=\cos(2x-15.420\ 165)$
	可信度	999.998 459 799 398
	用时/s	356

对于实验一,直观上可看得出得到的是一个完全精确的结果,因为 GEP 可信度评估可出的是零误差,所以可以直接放入算法 2 描述的 SQL 语句池中进行验证.

实验二、三存在一定的误差,仔细分析可以发现其误差值是很小的,在可信度上也有反映.

对于实验四而言,使用三角函数公式 $\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$ 和 $\cos(\alpha-\beta)=\cos\alpha\times\cos\beta+\sin\alpha\sin\beta$ 进行处理可以得到

$Y=\sin(2x+5)=0.283\ 662\ 185\ 463\ 226\ 25\times\sin(2x)-0.958\ 924\ 274\ 663\ 138\ 5\cos(2x)$

$Y=\cos(2x-15.420\ 165)=0.283\ 841\ 737\ 461\ 836\ 22\times\sin(2x)-0.958\ 871\ 142\ 581\ 027\ 9\cos(2x)$

两者的误差也是很小的.

对于上述实验的结果,均可以做出合理的假设并使用数据进行验证,不过需要根据不同的函数类型分别寻找相对应的数学理论进行指导.

如对于实验四,得到的是三角函数类型表达式 $Y=\cos(2x-15.420\ 165)$,从程序不变量的分析角度出发,若认为计算机能提供足够的计算精度,综合考虑分析对象程序的应用场合,程序员在设计程序中使用 15.420 165 这样一个小数作为常量的可能性相对是小的,这时可以考虑进行合理的假设,先使用 $\cos(\alpha-\beta)=\cos\alpha\cos\beta+\sin\alpha\sin\beta$ 进行处理可以得到

$Y=\cos(2x-15.420\ 165)=0.283\ 841\ 737\ 461\ 836\ 22\times\sin(2x)-0.958\ 871\ 142\ 581\ 027\ 9\cos(2x)$

依据三角函数 $\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\times\sin\beta$、$\sin(\alpha-\beta)=\sin\alpha\cos\beta-\cos\alpha\sin\beta$、$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$、$\cos(\alpha-\beta)=\cos\alpha\cos\beta+\sin\alpha\sin\beta$ 寻找合适的整数或小数部分为 0.5 的常数,形成表达式后再用原始数据进行验证.当然不一定能够得到最后的精确解,但所做的假设并非完全盲目的,还是有一定的合理性.

对于非标准多项式型函数的表达式,为了去除误差获得最后的确切解依旧是一个尝试过程.另外,对于如实验三包含对数、指数等其它标准函数的情况的处理过程类似于上述的讨论.

5　结　论

本文内容给出了一种新的思路进行程序不变量的发现工作,利用 GEP 获取最优解的函数发现能力和数学理论的精确计算能力获得较客观的程序不变量的预置形式,能够进行目前技术没有处理的函数型程序不变量的发现生成工作.在函数型程序不变量范围内,有效地克服了目前程序不变量发现技术中存在的两个缺点,即计算盲目性与程序不变量形式预置的主观性.整个工作可以理解为是对目前程序不变量发现技术的一个扩展,它拓广了原有技术可发现的程序不变量种类,比如大于二次阶的多项式函数、简单复合函数等,增大了从程序运行轨迹数据中发现更多不变量的可能性.

参考文献:

[1] ERNST M D. Static and dynamic analysis: synergy and duality [C]//ICSE Workshop on Dynamic Analysis. Portland: ICSE Press, 2003: 24-27.

[2] MOCK M. Dynamic analysis from the bottom up [C]// ICSE Workshop on Dynamic Analysis. Portland: ICSE Press, 2003: 13-16.

[3] 周霞, 沈炯. 多目标免疫 GEP 算法及其在多项式 NARMAX 模型辨识中的应用[J]. 控制与决策, 2014, 29(6): 1009-1015.

[4] 邓超强, 曾凡平, 武飞, 等. 程序不变量到断言的自动转换方法研究及其应用[J]. 计算机应用与软件, 2012, 29(11): 177-180.

[5] MICHAEL D. Ernst. dynamically discovering likely program invariants[D]. Seattle: University of Washington, 2000.

[6] 唐常杰, 彭京, 张欢, 等. 基于基因表达式编程的知识发现的三项新技术——转基因、重叠基因表达和回溯进化[J]. 计算机应用, 2005, 25(9): 1978-1981.

[7] 刘树锟, 阳小华. 程序不变量检测技术[J]. 计算机工程与科学, 2011, 33(3): 108-112.

[8] 阳小华, 黄彩霞. 基于 GEP 的线性指数函数型程序不变量动态发现方法[J]. 南华大学学报(自然科学版), 2012, 26(1): 63-67.

[9] 刘杰, 阳小华. 基于关系数据理论的函数型程序似然不变量动态检测方法[D]. 衡阳: 南华大学, 2008.

[10] 汪锐, 唐常杰. 基因表达式编程在多项式函数分解和并列函数关系挖掘中的研究和应用[D]. 成都: 四川大学, 2005.