软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(12):4464–4475 [doi: 10.13328/j.cnki.jos.006341]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

PROPER: 一个概率程序终止性与正确性分析工具*

赵旭慧[1], 邓玉欣[1], 符鸿飞[2]

[1](华东师范大学 上海市高可信计算重点实验室, 上海 200062)

[2](上海交通大学, 上海 200240)

通信作者: 邓玉欣, E-mail: yxdeng@sei.ecnu.edu.cn

摘 要: 概率程序将概率推理模型与图灵完备的编程语言相结合, 统一了对计算和不确定性知识的形式化描述, 能够有效地处理复杂的关系模型和不确定性问题. 提供了一种用于分析仿射概率程序的工具 PROPER. 一方面, 它有助于定性和定量地分析仿射概率程序的终止性, 可以验证该概率程序是否以概率 1 终止, 估计期望终止时间的上限, 并计算步数N, 使得N步后给定程序的终止概率呈指数下降; 另一方面, 它可以估计一个断言成立的概率区间, 这有助于分析变量不确定性对概率程序结果的影响. 通过实验表明, PROPER 对分析各种仿射概率程序是有效的.

关键词: 概率编程; 终止性; 断言分析; 程序验证

中图法分类号: TP311

中文引用格式: 赵旭慧, 邓玉欣, 符鸿飞. PROPER: 一个概率程序终止性与正确性分析工具. 软件学报, 2022, 33(12): 4464–4475. http://www.jos.org.cn/1000-9825/6341.htm

英文引用格式: Zhao XH, Deng YX, Fu HF. PROPER: Tool for Analyzing Termination and Correctness of Probabilistic Programs. Ruan Jian Xue Bao/Journal of Software, 2022, 33(12): 4464–4475 (in Chinese). http://www.jos.org.cn/1000-9825/6341.htm

PROPER: Tool for Analyzing Termination and Correctness of Probabilistic Programs

ZHAO Xu-Hui[1], DENG Yu-Xin[1], FU Hong-Fei[2]

[1](Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China)

[2](Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Probabilistic programs combine probabilistic reasoning models with Turing complete programming languages, unify formal descriptions of calculation and uncertain knowledge, and can effectively deal with complex relational models and uncertain problems. This study provides a tool for analyzing termination and assertions for affine probabilistic programs, namely, PROPER. On one hand, it can help to analyze the termination property of affine probabilistic programs both qualitatively and quantitatively. It can check whether a probabilistic program terminates with probability 1, estimate the upper bound of expected termination time, and calculate the number of steps after which the termination probability of the given probabilistic program decreases exponentially. On the other hand, it can estimate the correct probability interval for a given assertion to hold, which helps to analyze the influence of uncertainty of variables on the results of probabilistic programs. The experiments show that PROPER is effective for analyzing various affine probabilistic programs.

Key words: probabilistic programming; program verification; termination; assertion estimation

在传统逻辑中, 知识的表示和推理是确定和机械的, 能够根据已知事实推测出未知结果. 但现实世界中存在大量的不确定性问题, 需要我们用先决条件知识和演绎推理来预测结果. 概率是表示不精确和不完备知识的重要工具, 为此, 概率编程的方法被提出来. 也就是说, 可以通过概率推理对不确定性问题进行决策.

概率程序是一类部分事实带概率的逻辑程序. 目前已有几种实用的概率编程语言, 如 Edward[1],

* 基金项目: 国家自然科学基金(62072176, 61832015, 61802254)
收稿时间: 2021-01-28; 修改时间: 2021-02-27; 采用时间: 2021-04-03

Anglican[2], WebPPL[3]和 Church[4]等, 它们可用于构建生成模型, 对隐藏的中间过程进行推理, 估计某些事件发生的可能性. 此外, 还可以用于量化预测中的不确定性, 在商业、军事、科学研究及日常生活等各个领域均有广泛应用. 目前, 随着对高可信软件需求的不断增加, 概率程序的分析与验证也逐渐受到了学术界和工业界的广泛关注. 一般来说, 程序验证是一个不可判定的问题. 鉴于这一困难, 一个更现实的考虑是验证某些重要的、可以自动验证的性质. 我们设计与实现了一个分析仿射概率程序终止性与断言正确性的工具 PROPER, 该工具理论基于文献[5–7].

终止性是保证软件正确性的基础. 未能终止的程序会导致系统无响应, 从而导致系统故障. 对于终止性, 我们分别从定性与定量两个方面进行分析.

- 定性分析旨在验证仿射程序是否以概率 **1** 终止(几乎肯定终止);
- 定量分析旨在计算期望终止时间(期望问题), 以及使终止的概率呈指数下降的步数, 即集中界限问题(concentration problem).

我们采用的方法是合成多项式秩上鞅(synthesize polynomial ranking supermartingales)[8], 它是线性秩上鞅的扩展.

对于断言分析, 我们的目标是推算在程序结束时给定断言正确的概率区间. 它在不完备知识处理系统、元推理、风险分析、认知搜索引擎等人工智能领域具有一定的潜在价值. 我们通过静态分析的方法, 探索尽可能多的、不同的程序路径, 从而分析断言在每条路径结束时的正确率. 当探索到的程序路径覆盖率足够高时, 可以视为是对整个程序进行分析.

综上所述, 本文的工作主要包括以下几个部分.

(1) 定义并解析概率程序. 支持十余种概率分布, 便于构建概率模型;

(2) 将终止性验证归约为线性规划问题. 分别从定性与定量两个方面进行分析;

(3) 将分析断言正确性问题归约为多面体求解问题. 特别值得一提的是, 它可以处理具有无限状态的概率程序.

PROPER 支持计算仿射概率程序的终止时间集中界限, 并集成了终止性验证、期望终止时间分析和断言正确性分析的功能, 这是概率程序分析的 3 个主要特征.

本文的源代码和实验数据均可在 https://github.com/Healing1219/PROPER 中获取.

1 预备知识

概率程序是对经典命令式程序的扩展, 通过概率分布生成整型或浮点型随机值. 本节将介绍仿射概率程序的语法和语义.

1.1 语 法

我们定义了一种具有足够表达力且易于理解的概率编程语言, 如图 1 所示. 为了提高可读性, 以不同的字体区分关键字和终止符, 对于关键字使用粗体, 对于终止符使用大写. 概率程序与经典命令式程序中的语法相似, 主要区别在于: 存在一组随机值生成器, 可以用来模拟不同的概率分布(离散分布、二项式分布、泊松分布、整数均匀分布、实数均匀分布、指数分布、伽玛分布、贝塔分布、几何分布). 语句主要由赋值、条件分支(if/if-else)和循环(while)这 3 种类型组成. 在仿射概率程序中, 每个变量的赋值(或判断条件)都是线性或仿射形式, 比如 $y=x+5$(或 $y>2x$), 不支持变量之间的乘法、除法与指数等非线性运算. 变量分为程序变量和采样变量两类: 程序变量是普通的变量, 可以由整型、浮点型、布尔型或算术表达式赋值, 布尔变量主要作为判断条件用于条件分支和循环语句中; 采样变量由随机值生成器产生, 在程序运行过程中被赋予随机值. 由同一随机值生成器产生的采样变量总体服从连续或离散的概率分布. 断言是关于变量的布尔表达式, 在复合断言的情况下, 我们支持逻辑运算符“与”(&&).

program	:=	typeSpecifier **main** {stmt*}
stmt	:=	assign | condStmt | loop
assign	:=	intAssign | realAssign
condStmt	:=	ifStmt | ifElseStmt
ifStmt	:=	**if** LP boolExpr RP stmt*
ifElseStmt	:=	ifStmt **else** stmt*
loop	:=	**while** LP boolExpr RP stmt*
intAssign	:=	intVar EQUAL intExpr
realAssign	:=	realVar EQUAL realExpr
intRandom	:=	**uniformInt**(intVar, intVar)
		| **Binomial**(intVar, realVar)
		...
realRandom	:=	**uniformReal**(realVar, realVar)
		| **Gaussian**(realVar, realVar)
		...
intExpr	:=	intConst | intRandom | intExpr ± intExpr
		| intConst ∗ intExpr | intExpr / intConst
realExpr	:=	realConst | realRandom | realExpr ± realExpr
		| realConst ∗ realExpr | realExpr / realConst
boolExpr	:=	**true** | **false** | boolExpr ∧ boolExpr
		| intExpr relop intExpr
		| realExpr relop realExpr
relop	:=	< | > | ≥ | ≤ | ==

图 1 概率语言的语法规范

1.2 语 义

我们使用控制流图来表达概率程序的语义. 形式上, 以五元组$(L,X,R,\rightarrow,\perp)$表示, 其中,

- L 是一组有限标号的集合, 用来表示程序的位置. 程序中每一条语句都有唯一的标号, 例如, 标号 l_0 表示起始位置;
- $X=\{x_0,\ldots,x_n\}$表示一组程序变量的集合;
- $R=\{r_0,\ldots,r_m\}$表示一组采样变量的集合;
- $\rightarrow$表示转换关系, 其元素形式为三元组(l,α,l'), 其中: l 表示当前位置, l'表示下一个执行位置, α是从标号 l 到标号 l'需要服从的转换规则. 如果标号 l 处是赋值语句, 则α是一个更新函数 $f:\boldsymbol{R}^{|X\cup R|}\rightarrow\boldsymbol{R}^{|R|}$; 如果标号 l 处是条件分支或循环语句, 则α是一个谓词;
- $\perp$是一个不属于集合 L 的特殊标号, 以 $l_\perp$的形式表示, 意味着程序的终止.

显然, 任何概率程序都能以控制流图的形式表示. 设$(L,X,R,\rightarrow,\perp)$为与概率程序 P 相关的控制流图. 集合$X=\{x_0,\ldots,x_n\}$和$R=\{r_0,\ldots,r_m\}$分别表示程序变量与采样变量的集合, v_x 和 v_r 表示对集合 X 和 R 的赋值向量, 且按照所有变量定义的顺序排序. 变量的赋值满足函数 $f:v_{x_{i+1}}=f(v_{x_i},v_r)$.

为了便于标记, 我们假设格局(configuration)是一个二元组(l,v_x), 其中, $l\in L\cup\{\perp\}$表示在程序 P 中的位置. 我们用 (l_0,v_{x_0}) 表示程序 P 的初始格局. 程序根据转换关系(l,α,l')更新其状态. 程序 P 的执行状态可用前缀为 $(l_0,v_{x_0})...(l_k,v_{x_k})$ 的无限序列表示. 如果存在从 (l_0,v_{x_0}) 开始到 (l_k,v_{x_k}) 的有限路径, 则表示从起始格局 (l_0,v_{x_0}) 可以到达格局 (l_k,v_{x_k}). 按照惯例, 我们假设当程序终止时, 它永远停留在标号 $l_\perp$的位置.

假设当前格局为 (l_i,v_{x_i}). 根据不同类型的语句, 转换关系可以分为以下几种情况.

(1) 赋值语句: x=exp, 其中, x 是一个变量, 而 exp 是一个常量、变量、算术表达式或函数调用. 根据转换关系(l_i,α,l_{i+1}), P 的格局转变成 $(l_{i+1},v_{x_{i+1}})$, 其中, $v_{x_{i+1}}=f(v_{x_i},v_r)$, α是一个更新函数 $f:\boldsymbol{R}^{|X\cup R|}\rightarrow\boldsymbol{R}^{|R|}$;

(2) 条件分支语句: if α l_a else l_b. 如果变量赋值 v_x 满足判断条件α, 则 P 的格局变为 (l_a,v_{x_i}); 否则变为 (l_b,v_{x_i});

(3) 循环语句: while α $\{l_{i+1}\ldots l_m\}$. 如果变量赋值 v_x 满足判断条件α, 则 P 的格局变为 (l_{i+1},v_{x_i}); 否则变成 (l_{m+1},v_{x_i}), 其中, l_{m+1} 表示退出循环的标号. 如果整个程序结束, 则 $l_{m+1}=l_\perp$.

例 1: 如图 2 所示, 左边展示了一个概率程序, 右边是其控制流图. 在程序中, t, p 和 h 是程序变量,

Binomial(1,0.5)和 *unifInt*(0,10)是概率分布. *Binomial*(1,0.5)表示二项分布, 以相等的概率生成 0 或 1, 第 1 个参数表示实验次数, 第 2 个参数表示实验成功的概率; *unifInt*(0,10)表示整数均匀分布, 以相等的概率生成 0 到 10 之间的整数. 最左边的数字 1~6 表示程序标号, 省略了标号 l_0 处 h 和 t 的初始化. 该概率程序模拟了龟兔赛跑的场景. 假设乌龟的初始位置是 t, 兔子的初始位置是 h. 乌龟总是以 1 的速度向前移动. 当 $p>0$(即有 0.5 的概率), 兔子以 0–10 之间的一个随机数向前移动. 当兔子超过乌龟时, 程序终止.

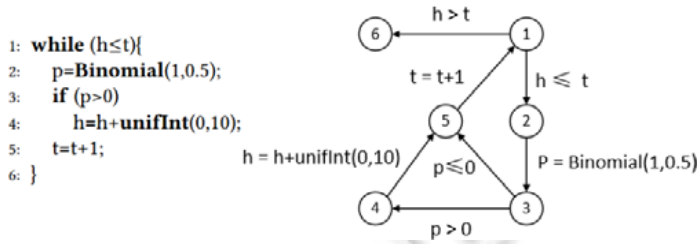


图 2 概率程序及其控制流图

2 终止性分析

终止性分析是程序验证的重要组成部分. 如果一个循环程序对所有可以进入循环的初值都终止, 那么这个程序就称为终止的(terminating). 确保终止是程序许多特性的必要条件, 例如完全正确性.

一般来说, 一个程序的终止性是不可判定的. 但对于某些子类程序, 终止性是可以验证的. 本文将传统的线性秩上鞅推广到多项式秩上鞅, 并通过合成多项式秩上鞅来验证仿射概率程序的终止.

PROPER 基于算法 1[5,6]分别对终止性进行定性和定量地分析.

2.1 定性分析

本节旨在分析仿射概率程序是否会以概率 **1** 终止(几乎肯定终止). 下面我们会介绍不变量(invariant)、预期望(pre-expectation)和多项式秩上鞅(polynomial ranking supermartingale)的概念.

定义 1. 不变量(invariant)是程序运算过程中在标号 l 处始终成立的谓词. 对于程序 P 中每个标号 l, 都有关于变量 X 的谓词函数 $I(l)$, 使 v_x 满足 $I(l)$. 在 PROPER 中, 我们仅支持线性不变量.

例 2: 我们考虑与例 1 中相同的程序, 其不变量 $I(l)$为

$$I(1)=h\leqslant t+9,\ I(j)=h\leqslant t,\ j=2,3,\ I(4)=h\leqslant t\wedge p>0,\ I(5)=h\leqslant t+10,\ I(\perp)=t\leqslant h.$$

定义 2. 预期望(pre-expectation)是一个多项式函数. 对于任意函数 $g:\boldsymbol{R}^{|X|}\to\boldsymbol{R}$, 我们有 $pre_g(l,v_x)$:

$$pre_g(l,v_x)=\begin{cases}\boldsymbol{E}(g(l,f(v_x,v_r))) & (1)\\ \mathbf{1}_{v_{x|=\alpha}}\cdot g(l_a,v_{x_i})+\mathbf{1}_{v_{x|\neq\alpha}}\cdot g(l_b,v_{x_i}) & (2)\\ g(l,v_x) & (3)\end{cases}$$

- 等式(1)适用于标号 l 处为赋值语句的情况, 是关于函数 g 的数学期望. 其中, v_r 服从其累计分布函数, 标号 l'表示下一个执行的语句位置, $f(v_x,v_r)$表示更新函数;
- 等式(2)适用于标号 l 处为条件分支或循环语句的情况. 当变量赋值 v_x 满足判断条件α时, $\mathbf{1}_{v_{x|=\alpha}}$ 的值为 1; 否则, $\mathbf{1}_{v_{x|=\alpha}}$ 的值为 0. 标号 $l_a(l_b)$表示判断条件为真(假)的下一个执行的语句位置;
- 等式(3)适用于程序终止时, 标号 $l=l_\perp$.

例 3: 我们再次考虑例 1 中的程序. 假设多项式 $g(l,v_x)$的最高次数为 2, 如下式所示:

$$g(l_n,h,t,p)=a_{n0}+a_{n1}\cdot h+a_{n2}\cdot h^2+a_{n3}\cdot ht+a_{n4}\cdot hp+a_{n5}\cdot t+a_{n6}\cdot t^2+a_{n7}\cdot tp+a_{n8}\cdot p+a_{n9}\cdot p^2,\ n=1,\ldots,5.$$

其中, a_{ni} 表示线性系数. 根据上面公式, 我们可以计算预期望. 对于标号 l_2 处的赋值语句:

$$pre_g(l_2,h,t,p)=\boldsymbol{E}(g(l_3,h,t,Binaomial(1,0.5)))=(a_{30}+0.25a_{39}+a_{38})+(a_{31}+0.5a_{34})\cdot h+a_{32}\cdot h+a_{33}\cdot ht+(a_{35}+0.5a_{37})\cdot t+a_{36}\cdot t^2.$$

对于标号 l_3 处的条件分支语句, 如果 $p>0$ 成立, 其下一步执行标号 l_4 处的语句:

$$pre_g(l_3,h,t,p)=g(l_4,h,t,1)=(a_{40}+a_{48}+a_{49})+(a_{41}+a_{44})\cdot h+a_{42}\cdot h^2+a_{43}\cdot ht+(a_{45}+a_{47})\cdot t+a_{46}\cdot t^2.$$

否则, 下一步执行标号 l_5 处的语句:

$$pre_g(l_3,h,t,p)=g(l_5,h,t,0)=a_{50}+a_{51}\cdot h+a_{52}\cdot h^2+a_{53}\cdot ht+a_{55}\cdot t+a_{56}\cdot t^2.$$

定义 3. 多项式秩上鞅(polynomial ranking supermartingale)是一个最高次数为 d 的多项式函数 $g(l,v_x)$, 满足以下条件.

- C1: 对于任意位置 l, $g(l,v_x)$是最高次数不超过 d 的多项式;
- C2: 对于非终止位置 $l\neq\perp$且变量 v_x 满足 $I(l)$时, 我们有 $g(l,v_x)\geqslant 0$;
- C3: 对于终止位置且 $K<0$, 我们有 $g(l,v_x)=K$;
- C4: 对于非终止位置 $l\neq\perp$且变量 v_x 满足 $I(l)$时, 我们有 $pre_g(l,v_x)\leqslant g(l,v_x)-\varepsilon$.

在上式中, $\varepsilon\geqslant 0$ 且 $K\leqslant-\varepsilon$. C1 是多项式条件; C2 为非终止条件, 表示在每个非终止位置上, 多项式秩上鞅的值都是非负的; C3 是终止条件, 它表示终止位置上, 多项式秩上鞅的值必须小于 0; C4 是秩上鞅有界差异条件(RSM difference condition), 它表示预期望不超过多项式秩上鞅与ε之间的差值. 从以上 4 个条件可以看出, $g(l,v_x)$是一个下界为 K 的函数. 所以本方法不能处理没有下界的分布, 例如高斯分布.

例 4: 根据例 1 中的程序、例 2 中的不变量 $I(l)$及例 3 中给出的预期望 $pre_g(l,v_x)$, 并根据以上多项式秩上鞅满足的条件, 我们可以得到以下不等式.

- (C4, 标号 1): $g(l_2,h,t,p)\leqslant g(l_1,h,t,p)-\varepsilon$;
- (C4, 标号 2): $\boldsymbol{E}(g(l_3,h,t,Binaomial(1,0.5)))\leqslant g(l_2,h,t,p)-\varepsilon$;
- (C4, 标号 3): $g(l_4,h,t,p)\leqslant g(l_3,h,t,p)-\varepsilon$, $g(l_5,h,t,p)\leqslant g(l_3,h,t,p)-\varepsilon$;
- (C4, 标号 4): $\boldsymbol{E}(g(l_5,h+unifInt(0,10),t,p))\leqslant g(l_4,h,t,p)-\varepsilon$;
- (C4, 标号 5): $g(l_1,h,t,p)\leqslant g(l_5,h,t,p)-\varepsilon$;
- (C2): $g(l_i,h,t,p)\geqslant 0$, for $1\leqslant i\leqslant 5$.

我们观察到: 通过量词消去法可以将 C2 和 C4 中的全称量词命题分解为合取式的形式, 但这种方法需要较强的计算能力. 为了减少计算代价, 我们基于 Handelman 定理[9]进行求解, 该定理用正线性函数来表示紧凸多面体上(compact convex polyhedra)的多项式. 下面, 我们简要介绍一些关于 Handelman 定理的基本定义.

设Γ是线性函数(1 阶多项式)的有限集合Γ的幺半群, 被定义为 $Monoid(\Gamma):=\left\{\prod_{i=1}^{n}k_i \mid n\in N,k_i,...,k_n\in\Gamma\right\}$.

Handelman 定理. 如果$\{\Gamma\}$是有界(compact)的, 则多项式 $h\in R[X]$的定义如下:

$$h=\sum_{i=1}^{m}c_i\times f_i,$$

其中, $m\in\boldsymbol{N}$, $c_i,\ldots,c_m\geqslant 0$ 且 $f_i,\ldots,f_m\in Monoid(\Gamma)$.

算法 1 描述了终止性分析的完整过程, 如下所示.

算法 1. 终止性分析.

输入: 概率程序 P;

输出: isT 判断程序是否终止, ET 终止期望时间与使程序终止的概率呈指数下降的步数 N.

1) 多项式 $g(l,v_x)$由程序变量 X 上所有单项式的集合构成, 最高次数不超过 d. 当 $l=\perp$时, $g(l,v_x)=K$;
2) 遍历抽象语法树, 分别对每个标号 l 计算不变量 $I(l)$和预期望 $pre_g(l,v_x)$;
3) 构建多项式 $H=g(l,v_x)-pre_g(l,v_x)-\varepsilon$, 其中, $\varepsilon\geqslant 0$ 且 $K\leqslant\varepsilon$;
4) 根据 Handelman 定理, $H'=\sum_{i=1}^{m}c_i\times f_i$, 其中, c_i 是非负实数, $f_i\in Monoid(\Gamma)$,

$$Monoid(\Gamma)=\left\{\prod_{i=1}^{d}I(l_i) \mid d\in\boldsymbol{N},I(l_1),...,I(l_n)\in\Gamma\right\};$$

5) 线性规划求解. H 的系数 a_{ni} 与 H'的系数 c_i 正好相等, 且 $c_i\geqslant 0$. 如果可以解出 a_{ni}, 则程序将以概率 1

终止(*isT*=true), 且继续执行第 6 步;

6) 计算期望终止时间的上界: $ET(P)\leqslant UB(P)=(g(l_0,v_{x_0})-K)/\varepsilon$;

7) 计算差异有界多项式秩上鞅区间[*a*,b]: $a\leqslant g(l',f(v_x,v_r))-g(l,v_x)\leqslant b$;

8) 根据公式 $\boldsymbol{P}(T_p\geqslant N)\leqslant \mathrm{e}^{\frac{-2(\varepsilon(N-1)-g(l_0,v_{x_0}))^2}{(N-1)(b-a)^2}}$ 计算步数 *N*.

在 PROPER 中, 终止性的定性分析可以参考算法 1 中的第 1 步~第 5 步. 理论上, 多项式 $g(l,v_x)$的最高次数 *d* 可以取任意正整数. 为简单起见, 在 PROPER 中, 我们设定 $g(l,v_x)$的最高次数为 2. 然后, 通过遍历抽象语法树来计算不变量 $I(l)$和预期望 $pre_g(l,v_x)$. 显然, $g(l,v_x)$是关于程序变量的多项式, 如果在标号 *l* 处它是一个赋值语句、条件分支或循环语句, 则预期望 $pre_g(l,v_x)$也是一个多项式(依据定义 2). 如果在标号 *l* 处于程序终止位置, $pre_g(l,v_x)=g(l,v_x)$, 即 $pre_g(l,v_x)$的值为负数 *K*(依据定义 3 中的 C3). 在 $pre_g(l,v_x)$和 $g(l,v_x)$都已知的情况下, 多项式 *H* 很容易被构建(依据定义 3 中的 C4). 然后, 我们利用 Handelman 定理和在步骤 2 中计算出的不变式 $I(l)$构造多项式 *H*′. Handelman 定理要求 *Monoid*(*I*)中每个多项式 k_i 必须是正线性组合. 通过移项, 我们可以将不变量构造为 $I(l)\geqslant 0$ 的形式. 由于不变量 $I(l)$是线性的, 且 *Monoid*(*I*)中的元素最多是 *d* 个不变量 $I(l)$的连乘, 因此, *H*′最高次数也为 *d*, 其系数与第 3)步中 *H* 的系数相对应. 最后一步进行线性规划求解, 计算出多项式 $g(l,v_x)$的系数. 如果存在一个解, 则表明概率程序将以概率 **1** 终止(几乎肯定终止).

另外, 在工具 PROPER 中, ε取值为 1, *K* 取值为–1. 下面, 我们在表 1 中展示例 1 的结果.

表 1 关于例 1 的结果

标号	不变量	$g(l,v_x)$
1	$h\leqslant t+9$	$3\cdot t-3\cdot h+27$
2	$h\leqslant t$	$3\cdot t-3\cdot h+26$
3	$h\leqslant t$	$3\cdot t-3\cdot h-14\cdot p+32$
4	$h\leqslant t\wedge p=1$	$3\cdot t-3\cdot h+17$
5	$h\leqslant t+10$	$3\cdot t-3\cdot h+31$
6	$t<h$	–1

表 1 第 1 列表示程序的标号; 第 2 列给出了每个标号 *l* 对应的不变量, 当程序从起始位置执行时, 到达标号位置变量赋值满足该谓词; 第 3 列为解出系数后的多项式 $g(l,v_x)$. 每执行一步, 其期望至少会减少一个ε. 从表格中我们可以看到: 从标号 1 到标号 2, 上鞅的期望值减少了 1. 类似地, 在从标号 2 到标号 3, 因为 $p=\boldsymbol{E}(Binomial(0.5))=0.5$, 也满足该条件. 当循环条件不满足时, 程序结束, 期望值为 *K*.

2.2 定量分析

本小节目的是推算期望终止时间上界, 并计算步数 *N*, 在 *N* 步后给定程序的终止概率呈指数下降. 具体步骤如算法 1 中第 6 步–第 8 步所示.

根据前面的步骤, 我们可以得到多项式模板 $g(l,v_x)$的系数. $g(l,v_x)$的存在, 保证了期望终止时间有限上界的存在. 当已知变量初始值为 v_{x_0}, 则多项式秩上鞅在初始位置的值是 $g(l_0,v_{x_0})$, 终止位置 *l* 的值为 *K*, 两个相邻位置的差值至少为ε. 因此, 当程序 *P* 几乎确定终止时, 我们可以得到其期望终止时间的上界为

$$ET(P)\leqslant UB(P)=\frac{g(l_0,v_{x_0})-K}{\varepsilon}.$$

然后我们重点讨论了集中问题, 即在 *N* 步后程序终止的概率呈指数下降. 我们的主要思想是基于 Azuma 不等式[10]、Hoeffding 不等式[11,12]和 Bernstein 不等式[12,13]. 在概率理论中, Azuma 不等式给出了有界鞅差的集中结果. Hoeffding 不等式是 Azuma 不等式的一个特例, 它提出了随机变量偏离期望值的概率上界. Bernstein 不等式是 Hoefffing 不等式的推广, 它不仅能处理自变量, 而且能处理弱自变量(weak independent variables). 根据算法 1 第 7 步中的公式 $a\leqslant g(l',f(v_x,v_r))-g(l,v_x)\leqslant b$, 我们可求出差异有界多项式秩上鞅(difference-bounded pRSM)区间[*a*,*b*].

结合 Hoeffding 不等式可知: 如果 $\varepsilon(N-1)>g(l_0,v_{x_0})$ 满足, 则不等式 $\boldsymbol{P}(T_p\geqslant N)\leqslant \mathrm{e}^{\frac{-2(\varepsilon(N-1)-g(l_0,v_{x_0}))^2}{(N-1)(b-a)^2}}$ 成立.

例 5: 再次考虑例 1, 假设初始值 t=30 和 h=5 时, 我们有 $ET(P)\leqslant\frac{3\cdot30-3\cdot5+27-K}{\varepsilon}=103$, 其中, K 和 ε 分别为–1 和 1. 存在鞅差区间为[–28,14]. 当集中终止界限的阈值 $\boldsymbol{P}(T_p\geqslant N)\leqslant10^{-3}$ 时, 步数 N 约为 6 296.

3 断言分析

本节对给定断言进行分析, 估计其在程序终止时的正确概率区间. 具体算法见文献[7], 分为两个步骤.

第 1 步是确定一个具有较高覆盖率的有限路径集 $S=\{s^1,\ldots,s^i,\ldots,s^n\}$, 其中, s^i 表示执行序列 $l_0\to\ldots\to l_k\to\ldots\to l_\perp$. 循环语句的控制流可能产生无限条路径, 因此, 我们采用符号执行的方法, 它可以在不需要指定输入的情况下, 系统地探索尽可能多的执行路径, 确保高覆盖率. 程序所有的执行路径都能以树状图的结构表示, 如图 3 所示. 每个框代表一个执行状态:

- 在框的第 1 行中, X 表示从变量到符号表达式映射的集合; π表示路径约束, 即执行该条程序语句时, 变量 X 应满足的约束条件;
- 在框的第 2 行中, stmt 表示下一步即将执行的程序语句.

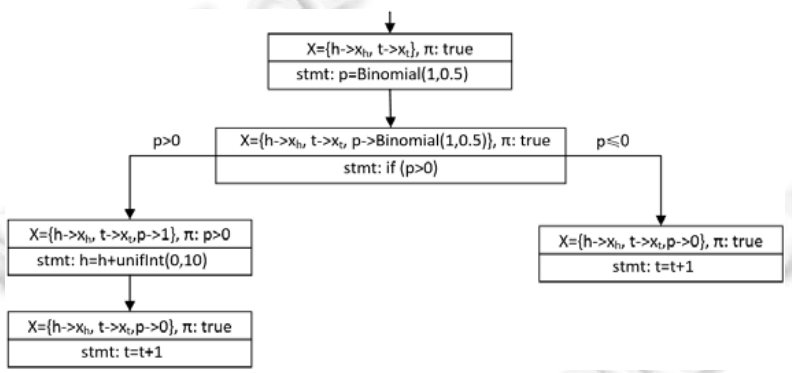


图 3 例 1 的符号执行树

符号执行的关键思想是: 将输入用符号值来表示, 同时将程序变量表示为符号表达式, 那么程序计算的输出值就是一个符号输入值的函数, X 与π在符号执行的过程中会不断更新. 同时, 为避免无关步骤的干扰, 我们对执行序列进行剪枝操作. 剪枝操作可以消去无关约束, 减少计算工作量. 例如: 对于条件分支语句, 如果不论执行 if 分支还是 else 分支, 都不会对断言中相关变量产生影响, 则该条件分支就可被剪枝. 假设路径集 S 的覆盖率有一个下界 c(如 95%), 则我们可以将在路径集 S 上对给定断言φ正确性的估计作为对整个程序的近似, 并可知未探测路径的干扰最多为 1–c. 这样, 我们就能处理无限状态的概率程序.

第 2 步是计算路径集 S 的覆盖率和给定断言正确的概率区间. 我们知道, 路径约束是关于实数变量 $\boldsymbol{Q}=\{q_1,\ldots,q_n\}$和整数变量 $\boldsymbol{Z}=\{z_1,\ldots,z_n\}$的谓词, 因此, 路径覆盖率的计算可以归约为根据单个变量的分布从实数集或整数集中提取样本点满足断言φ的概率推算问题. 同理, 给定断言ψ也可以被视为一组约束条件, 将它与路径约束相结合, 组成一组新的线性不等式$\varphi\wedge\psi$. 我们可以将其归约为凸多面体体积求解问题, 每个变量 q_i(或 z_i)都可看作是凸多面体的一个维数. 由于精确概率的计算与 n 维凸多面体的体积密切相关, 当维数到达一定数量时, 计算会变得非常困难, 时间复杂度很高[14]. 通常, 整个计算可能会由于某个过程出错或内存不足而导致失败. 在权衡计算效率和准确性后, 我们将工作重点放在计算其上下界上, 而不是计算某一精准的值.

计算路径覆盖率与断言正确区间的总体思想是, 用多个超立方体来逼近给定的多面体. 通过重要性采样的思想, 不断选取最有影响力的维数, 并沿选定的维数进行分解, 直到找到超立方体的边界并进行计算, 如算法 2 所述. 算法输入最大递归深度 *maxDepth* 和由变量和约束组成的结构体 p. 函数 *isDecomposable*(·)用于确

定当前凸多面体是否可以分解, 只有当该凸多面体至少存在两个维数时才能执行分析.

函数 *selectBranchDimension*(·)基于重要性采样(importance sampling)的思想, 选择一个无界或半有界的维数(semibounded dimension), 再或宽度最大的维数. 函数 *branchAndBound*(·)根据所选维数进行分解. 这里, *nBranch* 是维数分解的段数, 默认值为 2. 当各个维数都有界(*isABox*(·)=true)或递归深度超过 *maxDepth* 时, 将计算上下界并返回结果. 函数 *computeBoundingPro-bability*(·)用于计算上下界. 由于不存在重复路径, 因此断言正确的概率等于每条路径终止后正确概率之和. 路径覆盖率计算方法同上. 假设路径覆盖率的计算结果为 $[p_1,p_2]$, 断言正确率为 $[p_1', p_2']$, 它表示尚未探索的路径最多为 $1-p_1$. 断言正确率的上界需要通过添加 $1-p_1$ 来修正, 因此, 最终断言正确率的区间为 $[p_1,p_2-p_1+1]$.

算法 2. 断言正确性分析 *mcBoundProbability*.

输入: *p*=(*vars*,*contraints*),*maxDepth*;

输出: 概率区间 $[p_1,p_2]$.

if *isABox*(·)||*maxDepth*≤0 **then**

1) $[p_1,p_2]$+=*computeBoundingProbability*(·);
2) **end if**
3) **if** *isDecomposable*(*p*) **then**
4) *dim*=*selectBranchDimension*(·);
5) *polys*=*branchAndBound*(*dim*,*nBranch*);
6) **for** *p* in *polys*:
7) *mcBoundProbability*(*p*,*maxDepth*−1);
8) **end for**
9) **end if**

4 实现与实验评估

PROPER 采用 Java 语言开发编写, 并使用 lingo 与 lpsolve 库, 以支持线性和非线性方程组求解. 架构如图 4 所示, 由前端、后端和推理系统这 3 个部分组成.

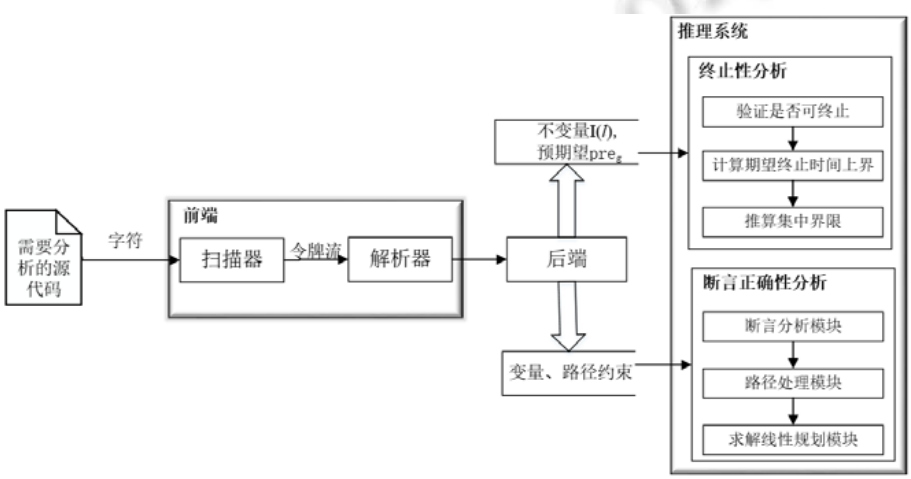


图 4 PROPER 的架构

前端的工作流程是基于解释器的原理, 包括扫描器和解析器两部分, 主要作用是将源代码转换成有效的中间表示并逐行执行. 扫描器通过读取源代码, 其转换成令牌流来执行词法分析, 主要功能是把字符拆分成我们需要的单词、数字与标识符等. 通过有限状态自动机, 从左至右逐个字符地对源代码进行扫描, 把作为字

符串的源程序转换为单词符号串的中间表示, 若能成功地从自动机初态扫描至终态, 则被视为可接受字符. 解析器的主要功能是接收令牌流并构造抽象语法树, 保证其语法结构在语义上是合法的. 先判断由扫描器产生的令牌流是否符合特定的模式: 如果符合, 再把这些模式与函数调用、变量调用、数学运算之类的表达式相关联; 如果有语句与图 1 中给出的语法规则不匹配, 则返回语法错误的相关提示并终止执行.

后端根据抽象语法树逐行执行代码, 每次读入一条语句, 并且根据这条语句的语义执行相应操作, 直到所有语句均被执行. 在遍历语法树的同时, 我们需要对每条程序语句进行分析, 收集相关的信息, 并需要对信息进行一些处理(比如约束合并), 处理完成后, 用于下一阶段的推理系统分析. 对于终止性分析, 我们需要计算对应于每个标号 l 处的不变量 $I(l)$和预期望 pre_g 表达式, 用于合成多项式上鞅; 对于断言分析, 我们需要知道执行这条路径时产生的采样变量和路径约束, 用于构造线性规划模型, 计算路径覆盖率和断言正确率.

终止性分析主要包括 3 个模块: (a) 验证程序终止性; (b) 计算期望终止时间上界; (c) 推算终止时间集中界限. 期望终止时间与集中界限的问题是建立在给定概率程序以概率 1 终止的基础上, 所以, 只有当在第 1 步中确保该概率程序终止, 才会继续执行后面两步. 多项式秩上鞅 $g(l,v_x)$的存在是概率程序终止的充分条件, 在已知多项式模板 $g(l,v_x)$的系数, 给定初始值 v_{x_0} 与 K 值时, 期望终止时间的上界根据公式 $\frac{g(l_0,v_{x_0})-K}{\varepsilon}$ 即可计算得. 推算终止时间集中界限问题的重点在于求出鞅差区间, 其可以视为凸优化问题, 即在给出等式约束与不等式约束的情况下, 求解出目标函数的最大值与最小值.

断言正确性分析主要包括 3 个模块: (a) 断言分析; (b) 路径处理; (c) 求解线性规划. 断言分析模块主要用于检查给定断言是否符合规范, 并记录断言所涉及的相关变量. 路径处理模块主要是通过符号执行收集路径并进行路径剪枝. 如果某条路径的执行序列中包含标号 l_k 且标号 l_k 处的语句执行不会影响断言相关变量,则可对其进行剪枝, 将标号 l_k 从执行序列中删除. 经剪枝处理后, 再判断路径集 S 中是否已经包含与该执行序列相同的路径: 如没有, 则将其加入到路径集 S 中. 如果连续 K 个迭代都没有产生新路径, 则表示路径收集完成. K 值的选择是根据贝叶斯因子检验, 在路径覆盖率至少为 95%、置信度为 0.99 的条件下, 我们将 K 固定为 90. 在符号执行的同时, 会计算每条路径的约束条件, 给定断言也可视为一组约束条件. 因此, 我们将计算路径覆盖率和断言正确的概率区间这两个问题转化为线性规划问题.

由于各个部分功能都是模块化的, 每个类或函数都有各自的职责, 且相互之间的耦合性较低. 按照设计模式中的开闭原则, 当需要增加新功能时, 可以直接扩展接口或类, 不必对原有代码进行修改.

PROPER 可适用于 Windows 和 Linux 操作系统, 在 Windows 系统中支持用户图形界面操作. 如图 5 所示, 分别展示了终止性验证和断言正确性分析的运行界面.

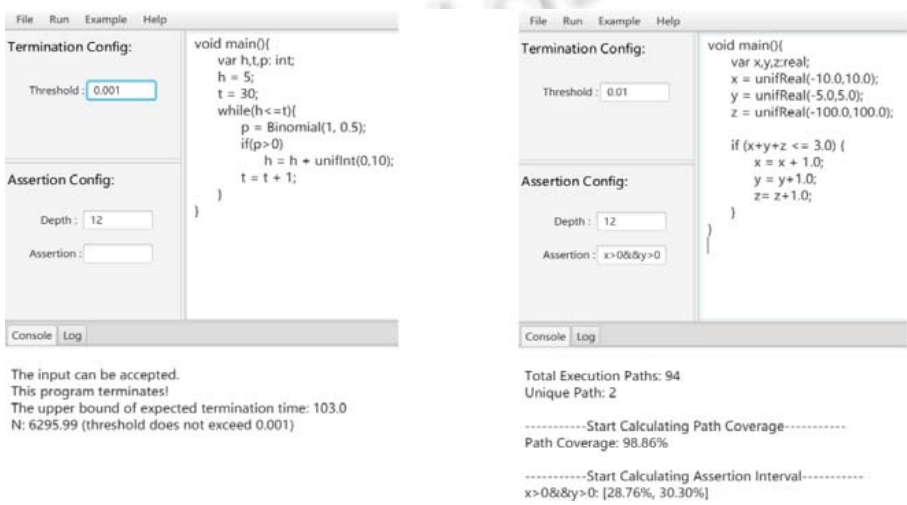


图 5 PROPER 的用户界面

界面顶部是菜单栏，点击鼠标左键，会以下拉条的形式显示下拉内容: “File”中主要包含导入和保存文件的功能; “Run”中包括语法检查、终止性验证和断言正确性分析的功能; “Example”中包含一些能直接运行的实例. 界面左上部分可设置集中终止界限的阈值，如设为 0.01，则表示当程序执行 N 步后，有 0.99 的概率终止. 左下部分可设置递归深度和断言. 在界面右半部分可以写入概率程序，也可通过外部文件导入. 界面底部展示实验分析的结果.

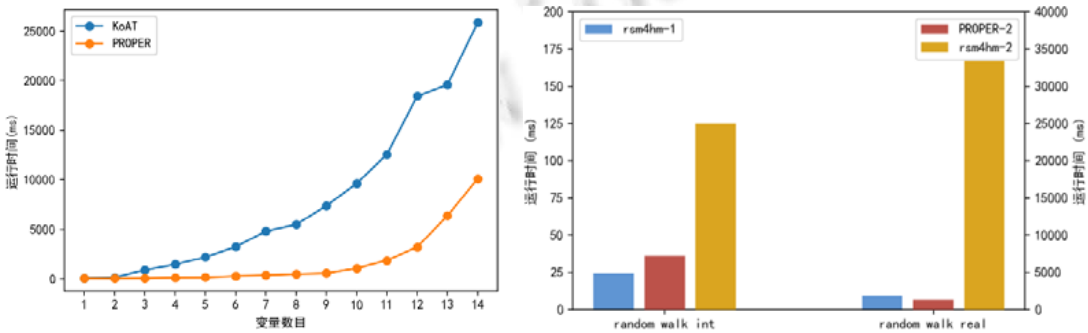
PROPER 能够处理各种基本结构的概率程序，例如 while 循环(如 Simple)、多层嵌套的 while 循环(如 NestedLoop)、while 循环中有 if 条件(如 Tortoise-Hare)、while 循环中有 if-else 条件分支(如 RandomWalk)、while 循环中有嵌套条件语句(如 Bitcoin mining)和 while 循环中有嵌套条件分支语句(如 Gambler2). 另外，我们也尝试了生成 4 935 个约束(线性方程)和 7 561 个参数的大规模程序(如 Herman). 这些示例在 Github 上可以找到.

表 2 中展示了终止性分析的实验结果，其中，v_x 是各变量的初始值; $g(l,v_x)$是求解出的多项式秩上鞅; $UB(P)$是期望终止时间的上界; N表明在N步后，给定程序的终止概率呈指数下降(我们设置的阈值为 0.01). 为便于阅读，我们保留到小数点后两位.

表 2 终止性分析

Ex.	v_x	$g(l,v_x)$	$UB(P)$	N
Simple	x=100	6·x	601	1474.82
NestedLoop	x=1, m=2	1040·m−1040·x+208	1 249	158141.27
Award	$bonus$=0	−4.0·$bonus$+440	441	4522.28
RandomWalk	$position$=0	−20·$position$+120	121	4695.66
Gambler	$money$=3	400·$money$+400	1 601	1506477.30
Gambler2	$money$=10	45.37·$money$+226.86	908.44	3358.51
Bitcoin mining	$coin$=10	5.32·$coin$	54.18	6.39E7
Tortoise-Hare	h=5, t=30	3·t−3·h+27	103	4264.27
Cars	s=0, $s0$=10	1.90·$s0$−1.90·s+0.24·s_0^2+3.81	47.67	165.90
Cars2	s=0, $s0$=10	3.05·$s0$−3.05·$s0$+0.38·s^2−0.76·s·$s0$+0.38·s_0^2+6.10	75.67	259.12
Herman	$process_i$=0 (i=1,…,5), p=0.5 $notStable$=0, $count$=0	−56·$notStable$+280	281	957.93
Carton	$Total$=0.0, $nPacked$=0, $nPer$=5, h=0, $count$=0, l=0	45.31·$nPerCarton$−45.31·$nPacked$+45.31	272.85	787.55

另外，我们将工具 PROPER 与 KoAT[15]和 rsm4hm[16]进行了对比，对比实验结果如图 6 所示. KoAT 用于推导期望终止时间，仅支持以 x:=x+p_i·c_i的形式更新变量赋值，所有概率 p_i之和为 1，相当于每次循环迭代中所有变量的更新服从离散联合分布.



(a) 与 KoAT 对比结果 (b) 与 rsm4hm 对比结果

图 6 实验对比

下面我们分别按照两者的语法规则编写相同语义的程序，在运行环境为 Ubuntu 18.04 LTS 的虚拟机上进行实验. 实验结果如图 6(a)所示，其中，横坐标表示与循环条件相关的变量数目，纵坐标表示验证终止性且推算出期望终止时间的运行时间(ms). rsm4hm 关注集中界限问题，用于计算尾概率，它给定迭代步数 N，输出尾

概率; 而 PROPER 是给定尾概率(即集中终止界限的阈值), 输出需要迭代的步数 *N*. rsm4hm 通过可以秩上鞅逼近高阶矩(higher moments)产生更紧的界, 但每个标号 *l* 处的不变量需要用户在例子中标明. 在都仅考虑一阶矩的情况下, 我们对比了文献[15]中提供的随机游走的例子. 如图 6(b)所示: 蓝色柱状表示 rsm4hm 利用线性一阶矩计算的运行时间, 其高度依据左侧刻度; 红色柱状和黄色柱状分别表示 PROPER 和 rsm4hm 利用二次一阶矩计算的运行时间, 其高度依据右侧刻度. 其中, 对 rsm4hm 运行时间的统计仅包括合成秩上鞅不包括尾概率的计算部分.

断言分析的结果与文献[7]给出的基本一致, 唯一不同的在于: 我们支持逻辑运算符"&&", 它可以分析同时满足两个或更多条件的情况, 结果如表 3 所示. 其中: assertions 列表示用户输入需要被分析的断言; *c* 列表示收集到路径集 *S* 的覆盖率下界; *bound* 列表示断言正确的概率区间, 精确度同样保留到小数点后两位.

表 3 估计断言正确的概率区间

Ex.	assertions	*c* (%)	*bound*
framingham	*points*⩾10	96.36	[13.73%,17.36%]
	points−*Errpoints*⩾5	96.36	[77.81%,84.93%]
	points−*pointsErr*⩾5	96.36	[17.76%,24.68%]
Sum-three	*x*>5 && *y*>0	98.86	[15.05%,16.36%]
	x>0 && *y*>0 && *z*>0	98.86	[12.56%,13.73%]
	x+*y*>*z*+10	98.86	[44.77%,47.07%]
	x+*y*+*z*>100	98.86	[1.24%,2.59%]

5 结束语

很多软件系统中都存在不确定性, 包括数据分析、随机算法和蒙特卡罗模拟[17]. 我们设计了 PROPER 来更方便地分析概率程序. PROPER 有助于对概率程序的终止进行定性和定量分析, 还可以收集具有高置信覆盖率的路径集, 并为概率程序中的断言正确性计算概率区间.

- 相关工具

用于概率程序分析的软件工具尚未受到太多关注. 据我们所知, 目前存在的工具有 PSI[18], PSense[19]和 ProbFuzz[20]. PSI 是一个用来近似概率密度函数的符号推理工具, 其中, 概率密度函数由概率程序表示. PSense 是一个自动验证工具, 它针对初始输入为概率程序的敏感性生成严格的上界. ProbFuzz 是一个用于测试概率程序的工具. 上述提及的工具都不考虑概率程序的终止分析. 例如, ProbFuzz 侧重于测试而不是验证, PSI 只考虑推理, PSense 则侧重解决敏感性问题. 此外, PSI/PSense 要求输入的程序中 while 循环具有有限的迭代次数, 而 PROPER 可以处理迭代次数可能无限的循环.

此外, 我们还将 PROPER 与其他 3 篇文献[8,15,21]中的工具进行了比较, 分别具有不同的优势: 文献[8]的工具可以证明概率程序的几乎肯定终止, 但不能像我们的工具那样生成预期终止时间的上界; 文献[21]的工具虽然可以计算期望成本消耗的上界, 但由于它不是基于 Farkas 引理, 因此对于线性秩上鞅算法是不完备的, 而我们的算法是完备的; 文献[15]的工具旨在用可靠完备的算法推导期望运行时间, 但只能处理一类特殊的仿射程序, 循环约束只能是一个线性不等式, 循环体只能是简单的增量赋值形式 *x*:=*x*+*c*, 其中, *x* 是程序变量, *c* 是常数, 而我们的算法针对通用仿射概率程序.

- 未来工作

不可否认的是, 我们只解决了概率程序分析和验证这一复杂问题的某些方面, PROPER 仍有很多需要改进的地方: 1) PROPER 只能处理仿射程序, 不能处理变量与变量之间的乘法、除法和指数等更复杂的情形; 2) PROPER 不支持非确定性概率程序, 它要求程序变量符合一定的概率分布, 不包含非确定的转换; 3) 在今后的工作中, 我们希望在保证高效率的前提下, 找到一种更好的方法来计算准确的终止时间和断言正确性.

References:

[1] Tran D, Kucukelbir A, Dieng AB, *et al*. Edward: A library for probabilistic modeling, inference, and criticism. 2016.

[2] Tolpin D, Willem V, Yang H, *et al*. Design and implementation of probabilistic programming language Anglican. In: Proc. of the IFL 2016. ACM, 2016. 6:1–6:12. [doi: 10.1145/3064899.3064910]
[3] Goodman N, Stuhlmuller A. The design and implementation of probabilistic programming languages. 2014. http://dippl.org
[4] Goodman N, Mansinghka V, Roy D, *et al*. Church: A language for generative models. In: Proc. of the UAI. AUAI, 2012. 220–229.
[5] Chatterjee K, Fu H, Goharshady AK. Termination analysis of probabilistic programs through positivstellensatz's. In: Proc. of the CAV 2016. 2016. 3–22. [doi: 10.1007/978-3-319-41528-4_1]
[6] Chatterjee K, Fu H, Novotný P, *et al*. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In: Proc. of the POPL 2016. 2016. 327–342. [doi: 10.1145/2837614.2837639]
[7] Sankaranarayanan S, Chakarov A, Gulwani S. Static analysis for probabilistic programs: Inferring whole program properties from finitely many paths. ACM Sigplan Conf. on Programming Language Design & Implementation, 2013, 48(6): 447–458.
[8] Chakarov A, Sankaranarayanan S. Probabilistic program analysis with martingales. In: Proc. of the CAV. 2013. 511–526. [doi: 10.1007/978-3-642-39799-8_34]
[9] Handelman D. Representing polynomials by positive linear functions on compact convex polyhedras. Pacific Journal, 1988, 132(1): 35–62.
[10] Azuma K. Weighted sums of certain dependent random variables. Tohoku Mathematical Journal, 1967, 19(3): 357–367.
[11] Hoeffding W. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 1963, 58(301): 13–30. [doi: 10.1080/01621459.1963.10500830]
[12] McDiarmid C. Concentration, in probabilistic methods for algorithmic discrete mathematics. In: Proc. of the Algorithms in Combinatorics 16. 1998. 195–248. [doi: 10.1007/978-3-662-12788-9_6]
[13] Bennett G. Probability inequalities for the sum of independent random variables. Journal of the American Statistical Association, 1962, 57(297): 33–45.
[14] Arora S, Lund C, Motwani R, *et al*. Proof verification and the hardness of approximation problems. Journal of the ACM, 1998, 45(3): 501–555. [doi: 10.1145/278298.278306]
[15] Giesl J, Giesl P, Hark M. Computing expected runtimes for constant probability programs. In: Proc. of the Automated Deduction (CADE 27). 2019.
[16] Kura S, Urabe N, Hasuo I. Tail probabilities for randomized program runtimes via martingales for higher moments. Cham: Springer, 2018.
[17] Hastings W. Monte Carlo sampling methods using Markov chains and their applications. Journal of the ACM, 1970, 97–109.
[18] Gehr T, Misailovic S, Vechev M. PSI: Exact symbolic inference for probabilistic programs. In: Proc. of the CAV. 2016. 62–83. [doi: 10.1007/978-3-319-41528-4 4]
[19] Huang Z, Wang Z, Misailovic S. PSense: Automatic sensitivity analysis for probabilistic programs. In: Proc. of the ATVA. 2018. 387–403.
[20] Dutta S, Legunsen O, Huang Z, *et al*. Testing probabilistic programming systems. In: Proc. of the FSE. 2018. 574–586. [doi: 10.1145/3236024.3236057]
[21] Chan N, Carbonneaux Q, Hoffmann J. Bounded expectations: Resource analysis for probabilistic programs. In: Proc. of the Programming Language Design & Implementatio. 2018. 496–512.

赵旭慧(1996－), 女, 硕士, 主要研究领域为形式化方法.

邓玉欣(1978－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为形式化方法.

符鸿飞(1984－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为形式化方法.