arXiv:2602.14573v1 [cs.PL] 16 Feb 2026

Polar：用于（概率）循环的代数分析器

Marcel Moosbrugger ✉、Julian Müllner、Ilio Bartocci 和 Laura Kovács ✉

维也纳工业大学，奥地利维也纳
{firstname.lastname@tuwien.ac.at}

摘要。 我们介绍 Polar 框架，它利用代数推理，完全自动化地分析经典循环和概率循环。Polar 的核心在于处理能够精确刻画循环语义的代数递推关系。为此，我们实现了多种技术，用于计算变量高阶矩递推关系的精确闭式解、推导不变量，以及求出循环对未知参数的敏感性。Polar 能够分析包含 if语句、多项式算术和常见概率分布的概率循环。通过将循环分析转化为线性递推求解，Polar 利用推导出的递推关系闭式解来计算最强多项式不变量或推导参数敏感性。在定义明确的编程模型限制范围内，Polar 既可靠又完备。放宽这些限制中的任何一项，都会显著增加计算难度。为提高效率并减轻计算负担，Polar 还提供了不完备但可靠的技术，用于计算变量组合的矩。

关键词： 概率循环 · 程序分析 · 线性递推 · 多项式不变量 · 敏感性分析

1 引言

概率程序（PPs）通过整合从常见概率分布中抽样的能力，扩展了经典程序 [9,42]。这种整合使概率程序能够在标准控制流中嵌入随机量，从而对复杂概率分布进行编码。因此，概率程序构成了一个统一框架，可用于对多个领域中的随机系统建模，例如机器学习 [26], 自然科学 [4], 信息物理系统 [21,56], 安全与隐私 [8,10], 或随机化算法 [50]。在此背景下，概率程序循环能够编码随机过程，对不确定性下的动态进行建模，例如动态贝叶斯网络 [57] 或自动驾驶车辆 [39,40]。

鉴于概率程序的应用范围广泛，对概率程序进行自动化形式分析本质上十分困难。正如 Joost-PieterKatoen 及其合著者在 [27], 中指出的：“概率程序的形式化验证严格来说比

非概率程序更难。”本文探讨 *Joost-Pieter Katoen* 在概率编程领域研究的一些验证挑战，并介绍 Polar——一种用于自动化分析非概率循环和概率循环的全自动工具。下文中，我们将非概率程序称为经典程序。我们的 Polar 框架补充了众多算法方法，例如 [32,33], Joost-Pieter Katoen 多年来所开发的方法。具体而言，Polar 将代数推理与统计方法相结合；我们相信，这形成了一套严谨的方法，Joost-Pieter Katoen 会乐于使用和试验。Polar 为开源软件，现已公开提供:

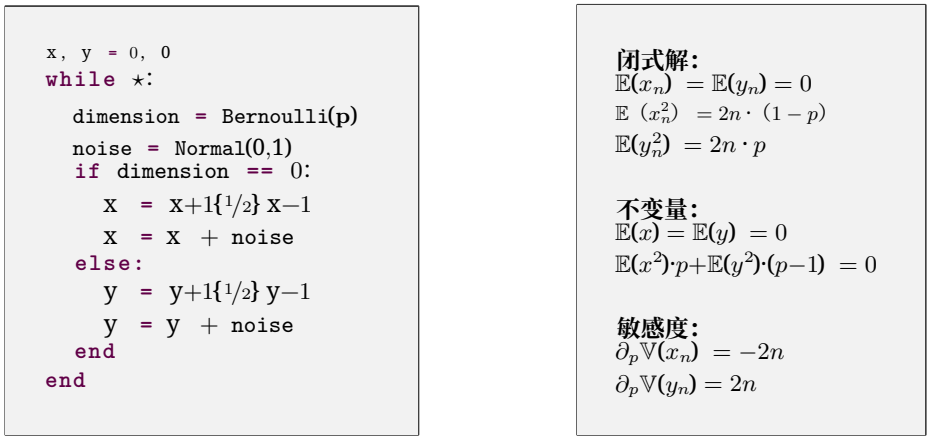
`https://github.com/probing-lab/polar`[1]

The POLAR 方法论。 由于随机性与确定性控制流相结合所带来的复杂性，形式化验证和分析概率程序一直备受关注。演绎程序演算 [42,43],，例如最弱前期望演算，为推理概率程序提供了有效方法。近年来，人们提出了各种概率程序演算，以处理不同的程序性质 [15,3,38,30,31]。自动化这些演算的主要挑战在于处理程序循环。与经典程序验证类似，验证概率循环需要不变式性质，而这些性质通常由人工提供。与这些工作不同，Polar 依靠代数递推求解，无需用户指导即可自动分析经典循环和概率循环。在其编程模型的某些限制条件下，Polar 可计算循环变量的矩所对应循环递推关系的闭式解。随后，这些闭式解可用于推导（最强）循环不变式，以及变量矩相对于未知参数的循环敏感度。

通过*POLAR*进行代数循环分析。Polar 的核心功能是提取代数递推关系，精确描述循环变量的统计矩，例如期望值乃至更高阶矩 [49]。具体而言，这些递推关系是常系数线性递推关系，即所谓的*C*-有限递推 [36]。在我们的设定中，Polar 会计算这类递推关系的闭式解，即以循环迭代次数为参数、表示循环变量各阶矩的函数 n。图 1a 展示了一个概率循环，它模拟了带有高斯噪声和未知参数的二维随机游走 `p`。Polar 自动计算出的闭式解如图 1b 所示，可给出变量 `x` 和 `y` 在任意循环迭代次数 n 及任意取值 `p` 下的期望值和二阶矩。

概率循环分析中的两个核心概念是循环不变量和敏感度。对于程序变量的矩，不变量是在循环执行全过程中始终成立的矩的性质。敏感度量化未知参数的变化如何影响变量矩。Polar利用

[1] 在全文中，Polar 均使用版本 *1 .0.*



```
x, y = 0, 0
while ⋆:
  dimension = Bernoulli(p)
  noise = Normal(0,1)
  if dimension == 0:
    x = x+1{1/2} x−1
    x = x + noise
  else:
    y = y+1{1/2} y−1
    y = y + noise
  end
end
```

闭式解：
$\mathbb{E}(x_n) = \mathbb{E}(y_n) = 0$
$\mathbb{E}(x_n^2) = 2n \cdot (1-p)$
$\mathbb{E}(y_n^2) = 2n \cdot p$

不变量：
$\mathbb{E}(x) = \mathbb{E}(y) = 0$
$\mathbb{E}(x^2)\cdot p + \mathbb{E}(y^2)\cdot(p-1) = 0$

敏感度：
$\partial_p \mathbb{V}(x_n) = -2n$
$\partial_p \mathbb{V}(y_n) = 2n$

(a) (b)

图 1：一个用于建模带有高斯噪声和参数 p 的二维随机游走的概率循环。图 1b 展示了 Polar 计算出的循环相关信息。

所计算出的循环递推关系的闭式解，以自动推断（最强）循环不变量，并量化变量矩对未知参数的敏感度。图1b展示了图1a所示循环中变量 x 和 y 的前两阶矩的最强多项式不变量，以及它们的方差相对于参数 p的敏感度。对于经典循环，Polar计算该循环的最强多项式不变量。

Polar 可分析的循环支持 if语句、符号常量、多项式算术、常量概率选择，以及从常见概率分布中抽样。为确保可计算性，Polar 对其输入程序施加了某些限制（见第 2.3 节）：除了要求概率为常量外，if 条件中的所有变量都只能取有限个值，且所有循环变量依赖都必须是线性的。有趣的是，若不遇到显著的难度边界，就无法放宽其中任何限制 [49,51]。尽管存在这些难度结果，Polar 仍实现了一种不完备但可靠的方法，可为程序变量的组合计算闭式解，即使对于某些违反可计算性限制的循环也如此 [5]。

贡献。 本文是首篇全面介绍 Polar 的教程，阐述 Polar 的功能及其使用方法。我们介绍 Polar 中实现的主要功能和技术，这些功能和技术旨在对经典循环和概率循环进行自动化分析，具体如下。

– 我们介绍如何使用 C-有限递推来计算循环变量 [49] 的（矩的）闭式解。我们还详细说明了 Polar 对输入程序施加的限制，以保证可计算性（第 2 节）。

lop ∈ {与、或}、 *cop* ∈ {=、≠、<、>、≥、≤}、 *Dist* ∈ {伯努利分布、贝塔分布、分类分布、离散均匀分布、指数分布、伽马分布、拉普拉斯分布、正态分布、截断正态分布、均匀分布}

hsymi ::= `a` | `b` | . . . *hvari* ::= `x` | `y` | . . . *hconsti* ::= $r \in \mathbb{R}$ | *hsymi* | *hconsti* `(` `+` | `*` | `/` `)` *hconsti hpolyi* ::= *hconsti* | *hvari* | *hpolyi* `(+` | `-` | `*)` *hpolyi* | *hpolyi*`**`*n* *hassigni* ::= *hvari* `=` *hassign_righti* | *hvari* `,` *hassigni* `,` *hassign_righti* *hcategoricali* ::= *hpolyi* `({`*hconsti*`}` *hpolyi*`)*` [`{`*hconsti*`}`]*hassign_righti* ::= *hcategoricali* | *Dist*`(`*hpolyi**`)` *hbexpri* ::= `true` `(*)` | `false` | *hpolyi hcopi hpolyi* | `not` *hbexpri* | *hbexpri h*循环*i hbexpri hifstmti* ::= `if` *hbexpri*`:` *hstatemsi* `(else if` *hbexpri*`:` *hstatemsi*`)*` [`else:` *hstatemsi*] `end` *hstatemi* ::= *hassigni* | *hifstmti hstatemsi* ::= *hstatemi*$^+$ *h*循环*i* ::= *hstatemi*`*` `while` *hbexpri* `:` *hstatemsi* `end`

图 2：描述 Polar [49] 输入程序语法的语法。

– 我们全面介绍 Polar 如何使用指数多项式闭式解，为经典循环和概率循环计算最强多项式不变量（第 3 节）。– 我们说明如何使用 Polar 计算变量矩相对于未知模型参数的敏感度 [48] （第 4 节）。– 我们展示 Polar 用于分析违反其可计算性限制的循环的不完备但可靠的技术 [5] （第 5 节）。

2 计算闭形式公式

程序循环中的赋值刻画了先前循环迭代中的变量值与当前迭代中的变量值之间的关系。同样，代数递推关系用序列的先前值来描述其各项的值。这种相似性使代数递推求解成为分析循环的自然工具。常系数线性递推（也称为*C*-有限递推）尤其适合自动化：C-有限递推所描述的序列总能表示为可计算的闭式公式 [36]。

Polar 的一项核心功能是通过递推方程对循环进行建模，这使 Polar 能够计算以下递推关系的闭式解：（i）经典非概率循环中循环变量值的闭式解；（ii）概率循环中循环变量的期望值和高阶矩的闭式解。这些闭式解还可用于推导循环不变量，或自动对循环参数进行敏感性分析（见第 3–4 节）。

此 *POLAR* 编程模型。 Polar 的输入程序是不嵌套的 while 循环，包含标准程序结构。图 2 展示了定义输入程序的语法。Polar 支持符号常数、不同表达式之间的概率选择，以及从具有

```
a,b,c = 0,1,1
x,z = 0,1
while ⋆:
  a,b,c = b, c, b+c
  z = -z
  x = x + a**2
end
```

(a) 经典循环。

```
x, y = 1, 0
while ⋆:
  y = y+1{1/2} y-2{1/3} y
  g = Normal(y,1)
  x = x + g**2
end
```

(b) 概率循环

图 3：Polar 可计算循环变量（矩）的闭式解的两个循环。此处，** 表示幂运算符。

已有矩的概率分布中抽样。对于图 3b 中的概率循环，赋值y 包含三个表达式之间的概率选择，其结果为$y+1$ 的概率是 $1/2$，为 $y-2$ 的概率是 $1/3$，剩余概率 $1/6$对应的结果为 y。条件化是概率编程语言中的常见特性，Polar 目前尚不支持。

下面的示例说明了 C-有限递推及其在经典循环自动化分析中的应用。在下文中，经典循环指非概率循环。

示例 *1*（斐波那契循环）。图 3a 中的经典循环使用程序变量 a、 b 和 c，对斐波那契数进行迭代。此外，程序变量 z 在 1 和 -1 之间切换。经过 n 次循环迭代后， x 的值对应于前 n 个斐波那契数的平方和。程序变量 a、 b、 c 和 z 的值序列可由以下 C-有限递推关系系统以及循环变量的初始值完全描述；这里，循环变量的初始值已经代入递推关系中。

$$\begin{aligned} a_{n+1} &= b_n & b_{n+1} &= c_n \\ c_{n+1} &= b_n + c_n & z_{n+1} &= -z_n \end{aligned} \tag{1}$$

其中， a_n 表示第 n 次循环迭代时 a 的值；其他变量/序列也同理。递推关系系统 (1) 可以根据循环体中的赋值自动构造，并简洁地用矩阵和向量表示：

$$\begin{pmatrix} a \\ b \\ c \\ z \end{pmatrix}_{n+1} = \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}}_{M} \cdot \begin{pmatrix} a \\ b \\ c \\ z \end{pmatrix}_{n} \tag{2}$$

矩阵系统 (2) 各分量的闭式解表示为指数多项式，即多项式与指数的和。更准确地说，对于系统中的每个变量 α，我们有

$$\alpha_n = \sum_{\lambda} p_\lambda(n)\lambda^n,$$

其中诸 λ 是矩阵 M 的特征值，且每个 $p_\lambda(n)$ 都是依赖于 λ 以及 α 的初始值的多项式。这种闭式表示并非仅适用于本例，而是对任意 C-有限递推关系系统 [36] 普遍成立。对于图 3a 中的循环，我们得到如下闭式解：

$$\begin{aligned} F_n &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n \\ a_n &= F_n & b_n &= F_{n+1} \\ c_n &= F_{n+2} & z_n &= (-1)^n \end{aligned}$$

其中 F_n 是斐波那契数的 比内公式 。

到目前为止，本例省略了图 3a 中的程序变量 x。尽管循环体中对 x 的赋值包含一个非线性项，其取值仍可用 C-有限递推加以描述。核心思想是针对程序变量中的单项式构造线性递推。为完整刻画 x 的取值，可将系统 (1) 用如下递推关系加以扩展：

$$\begin{aligned} x_{n+1} &= x_n + b_n^2 & b_{n+1}^2 &= c_n^2 \\ c_{n+1}^2 &= b_n^2 + 2cb_n + c_n^2 & cb_{n+1} &= cb_n + c_n^2 \end{aligned} \tag{3}$$

所得系统 (3) 由程序变量中单项式上的 C-有限递推构成，并给出程序变量 x 的闭式。

2.1 概率循环的闭式解

我们的工具 Polar 实现了经典循环闭式解的计算，也能分析概率循环。对于概率循环，程序变量的精确值本质上具有随机性，因此不遵循经典递推关系。不过，程序变量的期望值和高阶矩可以遵循 C-有限递推，称为矩递推 [11,45]。

示例 *2*（矩的闭式解）。图 3b 中的概率循环包含带有概率选择的赋值，以及从概率分布中抽取的操作。因此，循环变量的值不遵循经典代数递推。不过，借助期望值算子 $\mathbb{E}$，我们可以为程序变量的矩推导递推关系，如下所示。

由于图 3b 中的 x 赋值，我们有 $\mathbb{E}(x_{n+1}) = \mathbb{E}(x_n) + \mathbb{E}(g_{n+1}^2)$。$g$ 的赋值包含一个带有状态相关参数的分布，

可利用恒等式 $\texttt{Normal}(y,\ 1) = y + \texttt{Normal}(0,\ 1)$ 求解。因此，我们为 g的前两个矩构造如下递推关系：

$$\begin{aligned}\mathbb{E}(g_{n+1}) &= \mathbb{E}(y_{n+1}) + \mathbb{E}(\texttt{Normal}(0,1)) = \mathbb{E}(y_{n+1}) \\ \mathbb{E}(g_{n+1}^2) &= \mathbb{E}(y_{n+1}^2) + \mathbb{E}(2y_{n+1}\texttt{N}(0,1)) + \mathbb{E}(\texttt{N}(0,1)^2) \\ &= \mathbb{E}(y_{n+1}^2) + 1\end{aligned}$$

类似地，循环变量 y 的前两个矩的递推关系为：

$$\begin{aligned}\mathbb{E}(y_{n+1}) &= \frac{\mathbb{E}(y_n+1)}{2} + \frac{\mathbb{E}(y_n-1)}{3} + \frac{\mathbb{E}(y_n)}{6} = \mathbb{E}(y_n) - \frac{1}{6} \\ \mathbb{E}(y_{n+1}^2) &= \frac{\mathbb{E}((y_n+1)^2)}{2} + \frac{\mathbb{E}((y_n-1)^2)}{3} + \frac{\mathbb{E}(y_n^2)}{6} \\ &= \mathbb{E}(y_n^2) - \frac{1}{3}\mathbb{E}(y_n) + \frac{11}{6}\end{aligned}$$

结合 y 和 g 的矩递推关系以及 x 的矩递推关系，我们得到一个关于图 3b 中程序变量的单项式期望值的 C-有限递推关系系统。求解该递推关系系统后，Polar 推导出以下循环变量矩的闭式解：

$$\mathbb{E}(y_n) = -\frac{n}{6} \qquad \mathbb{E}(y_n^2) = \frac{n^2+65n}{36}$$
$$\mathbb{E}(x_n) = \frac{n^3}{256} + \frac{133n^2}{256} + \frac{205n}{128} + 1$$

正如示例1和示例2所示，通常可以通过对赋值应用期望值算子，并结合自底向上替换（用程序变量在循环体中出现时对应的赋值替换这些变量），来构造递推关系系统。在整个过程中，会利用期望的线性性等性质简化所得递推关系。构造递推关系时，对 if语句的处理并非易事。关于如何构造矩递推关系的更多详情，尤其是有关 if语句支撑集的内容，请参阅 [49]。

2.2 使用 Polar 计算闭式解

Polar 利用矩递推自动计算原始矩的闭式解，例如固定 $k \in \mathbb{N}$ 时的 $\mathbb{E}(x_n^k)$。这些闭式解由循环迭代次数 $n \in \mathbb{N}$ 参数化，也可以计算混合矩，例如 $\mathbb{E}(xy_n)$。此外，还可以利用原始矩计算中心矩。例如，程序变量 x 的方差由其前两个原始矩决定：$\mathbb{V}(x) = \mathbb{E}(x^2) - \mathbb{E}(x)^2$。Polar 提供以下用于闭式计算的命令行界面，其中 BENCHMARK 是待分析输入程序的路径。

```
python polar.py BENCHMARK --goals "GOAL1", ..., "GOALk"
```

Polar 通过 `goals` 参数，为从 `GOAL1` 到 `GOALk` 的每个目标推导闭式。每个从 `GOAL1` 到 `GOALk` 的目标表示程序变量中某个单项式 M 的期望值、中心矩或累积量，并具有以下形式之一：

– `E(M)` 期望值为 M；– `cd(M)` 阶数为 `d` 的 M的中心矩，其中 `d` $\in \mathbb{N}$；– `kd(M)` 阶数为 `d` 的 M的累积量，其中 `d` $\in \mathbb{N}$。

对于带守卫的循环，Polar 还可以通过额外传入标志 `after_loop`，计算循环终止后程序变量的矩。

2.3 可计算性的限制

图 2 中语法所定义的输入程序类是图灵完备的。因此，闭式解一般不可计算。尽管如此，对于下文所列输入程序施加一些进一步的限制（R）后，Polar 保证其可计算性 [45]。

R1 – 常数概率。 第一项限制是，概率选择结构中的概率以及分布参数都必须是常数，也就是说，它们在循环迭代之间不得发生变化。与示例 2 类似，Polar 通过利用特定恒等式消除参数对程序状态的依赖，从而支持某些非常数分布参数。

R2 – 有限守卫。 在循环执行过程中，出现在 if语句中和/或循环守卫中的程序变量只能取有限个值。直观来说，这一限制可防止输入程序达到图灵完备。

R3 – 无环非线性。 Polar 的程序模型允许多项式运算。然而，如果程序变量 x 对 y 存在非线性依赖，那么 y 就不得依赖 x。

在这三项限制 *R1*–*R3* 下，Polar 可以计算程序变量任意矩的闭式。限制 *R2* 不可或缺，因为若没有该限制，程序模型将是图灵完备的。然而，若要解除限制*R1* 和*R3*，也会面临计算能力方面的严重困难性边界（参见第 5 节）。

3 计算不变量

循环不变式是在循环执行过程中始终成立的性质，对于验证包含

循环的程序的安全性和正确性至关重要。一般而言，足以验证给定安全性质的不变式是不可计算的。因此，现有研究要么采用不完备的方法，要么对所支持的循环和不变式类别施加限制。多项式不变量由关于程序变量的多项式方程给出。每个多项式方程 $P = Q$ 都可以直接转换为 $P - Q = 0$。因此，只需考虑形式为 $P = 0$ 的多项式不变量，其中 P 是关于程序变量的多项式。多项式不变量构成了一类相对易于处理且富有表达力的不变式。

例 *3*。对于图 3a 中的循环，变量 a、b和 c 是三个连续的斐波那契数，z 在 1 和 -1之间交替，而变量 x 则对 a的平方求和。以下等式是该循环的多项式不变量，并对应于斐波那契数中广为人知的恒等式：

$$\begin{aligned} a + b - c &= 0 && \text{(Fibonacci definition)} \\ z - b^2 - bc - c^2 &= 0 && \text{(Cassini's identity)} \\ b^4 + 2b^3c - b^2c^2 - 2bc^3 + c^4 - 1 &= 0 && \text{(Quartic identity)} \\ b^2 - bc + x &= 0 && \text{（平方和恒等式）} \end{aligned}$$

循环的最强多项式不变量是程序变量中满足 $P = 0$ 为不变量的所有多项式的集合 I。I 通常包含无限多个元素，但其良好的结构缓解了这一问题：最强多项式不变量构成一个理想。这意味着 I 是一个群，并且对于任意 $P \in \mathrm{I}$，以及程序变量中的任意多项式 Q，都有 $P{\cdot}Q \in \mathrm{I}$。对于给定的循环，I 也称为该循环的不变理想。重要的是，希尔伯特基定理指出，每个多项式理想都可由有限多个元素生成 [22]。因此，只需列出有限个多项式，就能完整刻画 I。事实上，例 3 中的四个多项式不变量构成一个基，并生成图 3a 中斐波那契循环的最强多项式不变量。

考虑概率循环中的多项式不变量时，我们注意到，一般而言 $\mathbb{E}(x^k) \neq \mathbb{E}(x)^k$。直观地说，这就是所有程序变量的各个矩之间的多项式不变量集合不是多项式理想且不是有限生成的主要原因。要推断概率循环变量的矩之间的多项式不变量，一种解决方案是固定一组有限的矩，并且只考虑这些矩上的多项式不变量 [51]。例如，对于图 3b 中的概率循环，以下单个不变量构成了 $\mathbb{E}(x)$ 与 $\mathbb{E}(y)$ 之间所有多项式不变量的基：

$$\mathbb{E}(x) + 2\mathbb{E}(y)^3 - 33\mathbb{E}(y)^2 + \frac{103}{9}\mathbb{E}(y) - 1 = 0$$

对于给定的循环以及程序变量的一组有限矩 M，最强多项式不变量是 $\mathbb{I}$ 中所有多项式 P 的集合，其中这些多项式属于 M，且满足 $P = 0$ 是不变式。例如，若 $M = \{\mathbb{E}(x)\ \mathbb{E}(y)\}$，则 $\mathbb{I}$ 包含程序变量期望值 x和 y中的所有多项式不变量。集合 $\mathbb{I}$ 也称为 矩不变式理想，相对于

M [51]而言。重要的是，矩不变式理想是对经典多项式不变量的恰当推广：对于任意经典循环中的每个程序变量 x ，都有 $\mathbb{E}(x) = x$。因此，对于经典循环，相对于循环变量一阶矩的矩不变理想对应于经典不变理想。

3.1 从闭式解到不变理想

我们的工具 Polar 可以计算任意经典循环的不变理想的基，以及满足第 2.3 节可计算性限制的概率循环的任意矩不变理想的基。具体而言，它首先将所有程序变量（对于概率循环，则是固定的一组矩）的闭式解表示为关于循环迭代次数 n 的指数多项式（参见第 2 节）。其次，Polar 使用现有技术计算所有在任意 $n \in \mathbb{N}$ 下均成立的闭式解之间的多项式关系。在本节余下部分，我们将概述 Polar 如何计算指数多项式闭式解之间的所有多项式关系。

多项式闭式解。Polar 使用 C-有限递推来计算循环变量（的矩）的闭式解；一般而言，这些闭式解是指数多项式。不过，如果所有闭式解都是标准多项式（如示例 2 所示），那么计算（矩）不变理想的基就会大大简化。下面通过一个示例说明这种特殊情况下的基计算。

例*4*。 假设有一个含变量 x、 y 的经典循环，其多项式闭式为 $x_n = n^2 - 1$ 和 $y_n = n^3 + n$。由 $x - n^2 - 1$ 和 $y - n^3 - n$ 生成的理想 $\hat{}I$ 是多项式环 $\mathbb{Q}[x, y, n]$ 中的一个理想，并且包含该 $\hat{}$ 循环的不变理想 I。然而， I 引用了 n ，而它并不是该 $\hat{}$ 循环的变量。直观而言，为了计算 I，需要从 I 中消去 n。更准确地说，$\mathrm{I} = \hat{}I \cap \mathbb{Q}[x, y]$。通过使用 Buchberger 算法 [16] 计算格罗布纳基，可以推导出 $\hat{}I \cap \mathbb{Q}[x, y]$ 的一组基，进而得到 I 的一组基。就此而言，第一步 $\hat{}$ 是针对 I ，按照满足 $n > x,\ y$ 的字典序单项式序计算格罗布纳基；由此，我们得到以下格罗布纳基：

$$n^2 - x - 1 \qquad nx + 2n - y$$
$$ny - x^2 - 3x - 2 \qquad x^3 + 5x^2 + 8x - y^2 + 4$$

然后，$\hat{}\, I \cap \mathbb{Q}[x, y]$ 的基（因此也是 I 的基）由上述所有不含 n 的基元素构成。因此，I 的不变理想由单个不变式 $x^3 + 5x^2 + 8x - y^2 + 4 = 0$ 生成。这种从多项式理想中消去变量的技术普遍适用：(i) 按照字典序计算格罗布纳基，并使待消去的变量具有更高的优先级；(ii) 在所得的基中，仅保留不含“非必要”变量的元素，也就是执行变量消去 [22]。

一般闭式解作为指数多项式。通常，C-有限递推的闭式解包含多项式和指数项。格罗布纳基和 Buchberger 算法本质上依赖多项式算术，因此不足以计算 Polar 所支持的所有循环的不变理想。处理指数项的主要思路是将其替换为新变量，从而把闭式解转换为多项式。对于任意 $n \in \mathbb{N}$，指数项 b^n 与多项式 $p(n)$ 之间不存在恒成立的代数关系。另一方面，多个指数项之间可能存在代数关系。然而，将指数项替换为新变量时，关于这些指数项的任何信息都会丢失。因此，需要单独计算指数之间的乘法关系，并将其加入包含不变理想的基中。然后，如例4所示，消去循环计数器 n 以及为指数项引入的变量，从而得到不变理想的一个基。

例*5.* 假设有一个包含变量 x, y 和闭式解 $x_n = n2^n$ 和 $y_n = n^2 4^n$ 的经典循环。由于闭式解中包含指数，我们分别将 2^n 和 4^n 与新变量 a 和 b 关联；也就是说，$a = 2^n$ 和 $b = 4^n$。方程 $a^2 b - 1 = 0$ 刻画了 2^n 和 4^n 之间的乘法关系。由 $x - na$、$y - n^2 b$ 和 $a^2 b - 1$ 生成的理想 $\hat{}I$ 是多项式环 $\mathbb{Q}[x, y, n, a, b]$ 中的一个理想，并且包含该循环的不变理想。与例4不同，我们现在不仅消去 n，还消去 a 和 b，从而得到由单个多项式 $x^2 - y$ 生成的不变理想。

通常，指数之间的乘法关系不像例5中那样显而易见，需要通过自动计算得出。C-有限递推的闭式解中出现的指数底数是代数数。现有算法可用于计算一般代数数之间的乘法关系 [34,35]。这些算法并不简单，涉及抽象代数中的概念。如果所有指数底数都是有理数，Polar 会采用简单得多的方法。这种较简单的方法包括因式分解，以及求解一个线性丢番图方程组；当指数底数的分子和分母较小时，这种方法尤其适用。下面通过一个例子说明这种方法。

例*6.* 有理数2、 $1/4$和$1/6$ 之间的乘法关系由以下方程的自然解给出：

$$2^x \cdot \left(\frac{1}{4}\right)^y \cdot \left(\frac{1}{6}\right)^z = 1 \iff \left(\frac{2^x}{1}\right) \cdot \left(\frac{1}{2^y 2^y}\right) \left(\frac{1}{2^z 3^z}\right) = 1$$

对于任意解，分子中每个质因子的重数都必须等于分母中该质因子的重数。更准确地说，三元组 $(x,\ y,\ z)$ 是一个解，当且仅当它是以下线性系统的解：

$$\begin{aligned} x - 2y - z &= 0 \\ -z &= 0 \end{aligned}$$

```
x, y = 0, 0
while ⋆:
  x = x+2{1/2} x−1
  y = y+1{1/2} y−2
end
```

(a)

```
stop = 0
count = 0
x = 1
while stop == 0:
  stop = Bernoulli(1/2)
  count = count + 1
  x= 2x
end
```

(b)

图 4：两个具有可计算矩不变理想的概率循环。

该线性系统的所有自然解构成的集合，可由有限个极小解刻画，并且可以通过标准方法计算得到。对于这个例子，一个基由唯一的三元组 (2，1，0) 给出。这个例子中的方法只需增加一个步骤，就可以推广到任意有理数。因子 -1 可能出现的情况通过向线性系统中添加一个方程单独处理，以确保 -1 的重数为偶数。

3.2 使用 Polar 计算不变量

Polar 实现了 (i) [35] 中用于计算一般代数数底数的指数之间乘法关系的算法，以及 (ii) 一种适用于有理数底数的更简单方法。结合第 2 节介绍的闭式计算和格罗布纳基操作，Polar 可全自动推导（矩）不变理想。对于任何符合第 2 节限制条件的（概率）循环，（矩）不变理想基的可计算性均有保证。3 [45,51]。

示例 7. 图 4 展示了两个概率循环的例子。对于这些循环，Polar 可以针对任意一组程序变量矩计算矩不变理想。图 4a 中的循环描述了两个非对称随机游走，它们都从原点出发。所有一阶和二阶矩的矩不变理想可由 Polar 计算出的以下不变式刻画：

$$\begin{aligned} \mathbb{E}(x^2) - \mathbb{E}(y^2) &= 0 \\ \mathbb{E}(xy)^2 + 2\mathbb{E}(xy)\mathbb{E}(y^2) + \frac{81}{4}\mathbb{E}(xy) + \mathbb{E}(y^2)^2 &= 0 \\ \frac{2}{9}\mathbb{E}(xy) + \mathbb{E}(y) + \frac{2}{9}\mathbb{E}(y^2) &= 0 \\ \mathbb{E}(x) - \frac{2}{9}\mathbb{E}(xy) - \frac{2}{9}\mathbb{E}(y^2) &= 0 \end{aligned} \tag{4}$$

矩不变理想包含矩 $\mathbb{E}(x_n)$、 $\mathbb{E}(y_n)$、 $\mathbb{E}(x_n^2)$ 和 $\mathbb{E}(xy_n)$ 之间的所有多项式关系，这些关系在任意次数的循环迭代 $n \in \mathbb{N}$ 后均成立。为强调这些不变式对所有 $n \in \mathbb{N}$ 都成立，我们省略了不变式中矩的索引 n 。可以使用基 (4) 查询该理想中的不变式。例如，可以自动验证 $\mathbb{E}(xy) = \mathbb{E}(x)\mathbb{E}(y)$ 是一个不变式，这表明图 4a 中的两个随机游走互不相关。

图 4b 中的概率循环包含一个循环守卫。由于循环守卫中出现的唯一变量 `stop` 只能取值 0 或 1，因此 Polar 可以处理该循环。以下单个不变式构成了关于程序变量期望值的矩不变式理想的一组基：

$$-\mathbb{E}(\texttt{count}) + 2\mathbb{E}(\texttt{stop}) = 0$$

循环终止后，我们有 $\texttt{stop} = 1$ ，因此 $\mathbb{E}(\texttt{stop}) = 1$。将从循环守卫得到的这一事实与基中的不变式结合起来，即可得到终止后 `count` 的期望值，也就是 $\mathbb{E}(\texttt{count}) = 2$。有许多演算，例如最弱前期望演算 [43,9]，可以用于推理终止后的期望值。然而，就自动化而言，这些演算都需要手动提供循环不变式。研究这些演算与矩不变式理想之间的联系，是未来一个有趣的研究方向。

总之，Polar 通过其命令行界面以及参数 `invariants` 提供计算（矩）不变理想的功能：

```
python polar.py BENCHMARK --goals GOALS --invariants
```

使用此命令，Polar 首先计算通过 `goals` 参数传入的所有目标的闭式解，然后使用本节所述的方法计算这些闭式解之间的所有代数关系。如果未传入任何目标，Polar 默认使用（概率）循环 `BENCHMARK` 中所有程序变量的（期望值）。

4 参数与敏感性分析

使用概率循环对随机过程进行建模时，一个重大挑战是，某些过程参数的取值分布通常是未知的。Polar 通过支持表示任意实数的符号常数来应对这一问题，使得即使参数未知，也能够对这类过程进行建模。在此背景下，敏感性分析 的目标是描述/量化这些符号参数的微小变化如何影响循环变量的矩。因此，当模型参数仅部分已知时，敏感性分析能够为概率循环提供额外信息。由此得到的敏感性信息还可应用于各种任务，例如根据给定目标确定未知参数的最优值。

```
x, y = 0, 0
while ⋆:
  v = 正态分布(p, 2)
  x = x+v {7/10} x−v
  y = y+2x {1/3} y−x**2
end
```

(a)

```
u、w、x、y、z = 0、1、2、3、4
while true:
  z = z+p**2 {1/2} z+p
  y = y − 5pz
  w = 5w + x**2
  x = 5 +w + x
  u = x +py
end
```

(b) [48]

图 5：两个具有可计算的敏感度的概率循环。

例 *8*。 图 5a 描绘了一个参数为 p 、表示任意实数的概率循环。利用第 2 节中的技术并使用符号常数，Polar 可以计算循环变量任意矩的闭式解。例如， y 的期望值的闭式解为：

$$\mathbb{E}(y_n) = \frac{-8n^3\mathbf{p}^2 - 75n^2\mathbf{p}^2 + 30n^2\mathbf{p} - 150n^2 - 67n\mathbf{p}^2 + 30n\mathbf{p} - 150n}{225}$$

回想一下，Polar 计算出的闭式解是关于循环迭代次数 n的指数多项式。这些闭式解还包含多项式和参数化循环的指数底数中的符号常数。因此，可以完全自动地对变量矩的闭式解关于参数求导，从而得到变量矩的敏感度。对于图 5a 中的循环，我们用 $\partial_p\mathbb{E}(y_n)$ 表示 y 的期望值关于 p 的敏感度，并使用 Polar 推导出以下闭式解：

$$\partial_p\mathbb{E}(y_n) = \frac{-16n^3\mathbf{p} - 150n^2\mathbf{p} + 30n^2 - 134n\mathbf{p} + 30n}{225}$$

4.1 不可解循环的敏感性分析

利用变量矩的闭式解分析敏感度，要求循环满足第 2.3 节中的限制。通过提供一种替代方法，Polar 即使对于违反限制$R3$（无环非线性）的循环，也能够计算敏感度。对于图 5b 中的循环，程序变量 w 对 x 存在非线性 依赖。此外，变量 x 依赖于 w，从而形成非线性循环依赖。这违反了限制$R3$，因此 Polar 无法计算这些变量的矩的闭式解。一般而言，这类循环不存在指数多项式闭式解。因此，无法通过对变量矩的闭式解求导来计算敏感度。

借助 [48], 技术，可以通过构造敏感度的递推关系，而非关于矩的递推关系，并利用某些变量与所研究参数之间可能存在的独立性，来避开矩的计算。下面通过一个示例说明这种方法。

示例 *9.* 变量 `w` 和 `x` 导致图 5b 中的循环违反限制$R3$；然而，变量 `w` 和 `x` 与参数 `p` 无关。因此，改变 `p` 不会影响仅涉及 `w` 和 `x` 的矩。更确切地说，对于 w_n 和 x_n 中的任意单项式 M_n，$\partial_p \mathbb{E}(M_n) = 0$。这一事实可用于构造 $\partial_p \mathbb{E}(u_n)$ 的递推关系：

$$\begin{aligned}\partial_p \mathbb{E}(u_n) &= \partial_p \mathbb{E}(x_{n+1}) + \mathbb{E}(y_{n+1}) + \mathbf{p}\partial_p \mathbb{E}(y_{n+1}) \\ &= \mathbb{E}(y_{n+1}) + \mathbf{p}\partial_p \mathbb{E}(y_{n+1})\end{aligned}$$

利用 $\partial_p \mathbb{E}(x_n) = 0$，所有导致违反限制$R3$的变量都会从 $\partial_p \mathbb{E}(u_n)$ 的敏感度递推关系中消失。通过添加 $\mathbb{E}(y_n)$、$\partial_p E(y_n)$、$\mathbb{E}(z_n)$ 和 $\partial_p \mathbb{E}(z_n)$ 的递推关系，即可将 $\partial_p \mathbb{E}(u_n)$ 的递推关系补全为一个 C-有限递推关系系统。因此，敏感度 $\partial_p \mathbb{E}(u_n)$ 具有指数多项式闭式，尽管矩 $\mathbb{E}(u_n)$ 并没有：

$$\partial_p \mathbb{E}(u_n) = -5n^2\mathbf{p}^3 - \frac{15}{4}n^2\mathbf{p}^2 - 5n\mathbf{p}^3 - \frac{15}{4}n\mathbf{p}^2 - 40n\mathbf{p} + 3$$

4.2 使用 Polar 计算参数敏感度

如例 8 所述，通过对矩的闭式解求导来计算敏感度，可通过 Polar 的命令行界面实现：

```
python polar.py BENCHMARK --goals GOALS -sens_diff PARAM
```

使用此命令，Polar 会计算通过 `goals`参数传入的所有目标的闭式解，并对参数 `PARAM`求导；例如，对例 8 中的参数 `p` 求导。

此外，Polar 还会推导敏感度的递推关系，如第 4.1 节所述。为此，Polar 使用简单的静态分析技术来确定与参数无关的程序变量。将 `sens_diff`参数替换为 `sens`即可使用此功能：

```
python polar.py BENCHMARK --goals GOALS -sens PARAM
```

5 计算复杂性与不可解循环

正如第 2 节所述，Polar 的核心功能在于利用线性递推计算循环变量（的矩）的闭式解。为确保可计算性，Polar 对其编程模型施加了第 2.3 节中的三项限制。解除其中任何一项限制都会带来难以逾越的计算边界，下文将详细说明。

```
z = 0
while ⋆:
    z = 1 − z
    x= 2x + y**2 + z
    y= 2y −y**2 + 2z
end
```

图 6：一个不可解循环 [5]。

第 2.3 节的限制*R2* 不允许在 if 条件中使用任意变量。解除该限制将使 Polar 的编程模型图灵完备，并使闭式解不可计算 [49]。此外， [51]的最新结果表明，解除限制*R3*（该限制不允许循环体内存在任意多项式依赖）会使计算闭式解和最强多项式不变量成为 Skolem 难问题。Skolem 问题是数论中一个著名问题，其可判定性近一个世纪以来一直未得到解决 [23,58]。此外，限制*R1* 要求所有概率均为常数。允许状态依赖概率通常会导致矩递推中出现循环非线性依赖，从而带来与解除限制*R3*时类似的计算困难。

尽管受到*R1*–*R3*的限制，Polar 仍实现了其他技术，用于分析无法通过第 2–3 节所述方法求解的循环。我们将此类循环称为不可解循环 [4]。

5.1 不可解循环的闭式解

直观地说，不可解 循环是指循环体内存在任意多项式依赖的循环。因此，不可解循环不满足限制*R3*。尽管存在困难性结果，Polar 仍使用不完备但可靠的方法分析不可解循环。

例 *10.* 在图 6 的循环中，程序变量 y 对自身存在非线性依赖，因此该循环不可解。尽管变量 x 和 y 通常不存在指数多项式闭式解，但这些变量的组合可能存在闭式解。例如， x 与 y 之和遵循线性递推关系，并具有如下闭式解：

$$x_n + y_n = 2^n(x_0 + y_0 + 2) - \frac{(-1)^n}{2} - \frac{3}{2} \tag{5}$$

直观地说，通过将变量 x 和 y相加， y 的棘手非线性自依赖便会消失。一般而言，棘手的非线性依赖可能仍存在于变量的线性组合中，但使用更高次数的多项式组合后便会消失。

5.2 从不可解循环合成可解循环

给定一个不可求解的循环，可以使用第 5.1 节中所述的程序变量多项式组合，综合出一个可求解的循环，以过近似原始的不可求解循环 [5]。

例 *11.* 对于变量组合 $x + y$，图 6 中的不可解循环可由以下可解循环进行过近似。

```
s = x_0 + y_0
while ⋆:
    t = 1 − z
    s = 2s + 3 − 3z
    z = t
end
```

新程序变量 s 模拟了变量组合$x + y$的值。因此， s 和 $x + y$ 具有公式 (5) 中所述的相同闭式解。该可解循环对图 6 中的不可解循环进行了过近似，具体而言，相对于替换 $s \to x + y$，可解循环的每个不变式也是不可解循环的不变式。

5.3 使用 Polar 分析不可解循环

为自动分析不可解循环（通过计算第 5.1 节中讨论的（不可解）闭式解），Polar 实现了 [4,5] 的技术，以合成满足线性递推关系的程序变量多项式组合；当针对单个变量的循环更新无法产生线性递推关系时，此方法尤为适用。为不可解循环合成此类程序变量的多项式组合时，Polar 提供了以下命令行界面：

```
python polar.py BENCHMARK --synth_unsolv_inv --inv_deg DEGREE
```

使用参数 `synth_unsolv_inv`，Polar 会合成遵循良态递推 [4] 的“有问题”程序变量的多项式组合。为规避因放宽 *R1–R3* 的（部分）限制而产生的困难性结果，Polar 会合成所有此类多项式组合，其中多项式的次数由经 `inv_deg` 参数传入的自然数 `DEGREE` 所界定。对于概率输入循环，闭式解针对的是所合成组合的期望值，而非其精确值。

此外，Polar 还实现了 [5] 中所述的方法，从不可解循环合成可解循环，详见第 5.2 节。本质上，Polar 的合成过程会将“导致不可解的”程序变量替换为一个新变量，该变量模拟程序变量的良态多项式组合

，如第 5.1 节所述。Polar 合成的可解循环始终是确定性的，即使原始不可解循环是概率性的。对于概率输入循环，合成循环中的变量表示原始变量及程序变量组合的期望值。为了从给定的不可解循环合成可解循环，Polar 提供了以下命令行界面：

```
python polar.py BENCHMARK --synth_solv_loop --inv_deg DEGREE
```

调用此命令后，Polar 首先会计算次数不超过固定值 `DEGREE` 的所有良态多项式组合。对于找到的每种组合，Polar 都会综合出一个可解循环，对 `BENCHMARK` 中原本不可解的循环进行过近似。

6 相关工作

尽管（概率）循环的验证和不变式生成是受到广泛研究且面临诸多理论挑战的问题，但 Polar 的重点仍是借助计算机高效自动化循环分析。

对于经典程序，递推关系是程序分析中常用的工具 [41,29,24,37]。利用线性递推关系和格罗布纳基计算来推断多项式不变量的方法，最早由 [53,54] 提出，随后扩展到更一般的递推关系 [41,29]。工具 Aligator [28] 能够自动计算一类 经典循环的最强多项式不变量。然而，Aligator 支持的循环严格属于 Polar 支持的循环子集，因为 Aligator 无法处理任何形式的随机性或分支语句。在线性递推关系中使用线性递推来处理概率循环，最早由 [11], 引入，并由此促成了工具 Mora [12] 的开发。虽然 Mora 能为 Polar 支持的严格子集中的概率循环计算闭式解，但它缺少敏感性分析和不变式综合等额外功能。

由于一般概率程序的不变量不可计算，许多技术不得不采用不完备的方法。基于约束的方法使用模板表示候选不变式，并尝试求解模板的实例化，从而得到不变式 [25,32]。在 [7] 中，作者使用神经网络表示候选不变式，并通过反例对其进行训练。另一种利用反例引导概率不变式搜索的技术在 [13]中提出。在某些情况下，无需综合不变式，借助归纳技术即可自动验证概率程序 [14]。与不变式综合不同，也可以在演绎验证框架中使用用户提供的注释 [55]。

循环终止性是 Polar 尚未处理的一项性质。大多数概率终止分析工具都采用基于鞅的证明规则，并通过约束求解实现自动化 [17,2,20,19,18]。另一种方法是，概率终止工具 Amber 中实现的自动化技术 [46,47] 是

基于递推关系 [45]。与不变式综合类似，神经网络和学习技术也已用于概率终止分析 [1]。此外，期望成本分析工具也可用于验证有限期望运行时间 [52,44,6]。通过自动化方法扩展 Polar，以（反）证循环终止性，是未来一个值得探索的方向。

7 结论

我们提出 Polar 框架，利用代数递推对经典循环与概率循环进行自动化分析。Polar 支持分析包含 if语句、多项式算术以及从常见概率分布中抽样的（概率）循环。Polar 的核心功能是提取描述循环变量之矩的 C-有限递推。这些递推关系可得到指数多项式闭式，Polar 利用它们全自动地计算循环的最强多项式不变量。

为确保所支持的输入程序类的可计算性，Polar 对分支语句中的变量施加某些限制，并且仅允许无环多项式依赖。除闭式解与不变量外，Polar 还可用于计算程序变量（的矩）相对于未知模型参数的敏感度。对于违反无环多项式依赖限制的循环，Polar 提供了一种不完备但可靠的方法，用于推断变量组合的闭式解。因此，Polar 为（概率）循环的精确自动化分析提供了多种代数功能。

致谢。 感谢 Daneshvar Amrollahi、George Kenison 和 Miroslav Stankovic 就 Polar 相关主题展开的宝贵讨论与合作研究。我们感谢 ERC Consolidator Grant ARTIST 101002685、WWTF 10.47379/ICT19018 项目 ProbInG、TU Wien 博士学院 SecInt，以及 Amazon Research Award 2023 QuAT 提供的慷慨资助。

参考文献

1. Abate, A., Giacobbe, M., Roy, D.：学习概率终止证明。载于：CAV 论文集（2021 https://doi.org/10.1007/978-3-030-81688-9_1 ）。. Agrawal, S., Chatterjee, K., Novotný, P.：字典序排序超鞅：一种高效的概率程序终止方法。《ACM 程序语言会议论文集》（POPL）（2018 https://doi.org/10.1145/3158122 ）。. Aguirre, A., Barthe, G., Hsu, J., Kaminski, B.L., Katoen, J., Matheja, C.：用于概率敏感性的前期望演算。《ACM 程序语言会议论文集》（POPL）（2021 https://doi.org/10.1145/3434333 ）。. Amrollahi, D., Bartocci, E., Kenison, G., Kovács, L., Moosbrugger, M., Stankovic, M.：为不可解循环求解不变式生成问题。载于：SAS 论文集（2022）。https://doi.org/10.1007/978-3-031-22308-2_3 . Amrollahi, D., Bartocci, E., Kenison, G., Kovács, L., Moosbrugger, M., Stankovic, M.：（可解）循环分析。《形式化方法与系统设计》（2024），即将发表。6. Avanzini, M., Moser, G., Schaper, M.：概率程序的模块化成本分析。《ACM 程序语言会议论文集》（OOPSLA）（2020）。https://doi.org/10.1145/3428240 . Bao, J., Trivedi, N., Pathak, D., Hsu, J., Roy, S.：概率程序的数据驱动不变式学习。载于：CAV 论文集（2022）。https://doi.org/10.1007/978-3-031-13185-1_3 . Barthe, G., Grégoire, B., Béguelin, S.Z.：用于计算机辅助安全证明的概率关系霍尔逻辑。载于：MPC 论文集（2012）。https://doi.org/10.1007/978-3-642-31113-0 . Barthe, G., Katoen, J.P., Silva, A.：《概率编程基础》。剑桥大学出版社（2020 https://doi.org/10.1017/9781108770750 ）。. Barthe, G., Köpf, B., Olmedo, F., Béguelin, S.Z.：用于差分隐私的概率关系推理。载于：POPL 论文集（2012）。https://doi.org/10.1145/2103656.2103670 . Bartocci, E., Kovács, L., Stankovic, M.：为概率可解循环自动生成基于矩的不变量。载于：ATVA 论文集（2019）。https://doi.org/10.1007/978-3-030-31784-3_15 . Bartocci, E., Kovács, L., Stankovic, M.：Mora——基于矩的不变量自动生成。载于：TACAS 论文集（2020）。https://doi.org/10.1007/978-3-030-45190-5_28 . Batz, K., Chen, M., Junges, S., Kaminski, B.L., Katoen, J., Matheja, C.：通过归纳综合归纳不变式验证概率程序。载于：TACAS 论文集（2023 https://doi.org/10.1007/978-3-031-30820-8_25 ）。. Batz, K., Chen, M., Kaminski, B.L., Katoen, J.P., Matheja, C., Schröer, P.：格上的 k-归纳法及其在概率程序中的应用。载于：CAV 论文集（2021 https://doi.org/10.1007/978-3-030-81688-9_25 ）。. Batz, K., Kaminski, B.L., Katoen, J., Matheja, C., Verscht, L.：摊销期望运行时间演算。《 ACM 程序语言会议论文集》（POPL）（2023）。https://doi.org/10.1145/3571260 . Buchberger, B.：Bruno Buchberger 的 1965年博士论文：求解零维多项式理想的剩余类环基元素的算法。《符号计算杂志》（2006 https://doi.org/10.1016/j.jsc.2005.09.007 ）。. Chakarov, A., Sankaranarayanan, S.：使用鞅进行概率程序分析。载于：CAV 论文集（2013）。https://doi.org/10.1007/978-3-642-39799-8_34

18. Chatterjee, K., Fu, H., Goharshady, A.K.：通过正定性证书定理分析概率程序的终止性。载于：CAV会议论文集（2016）。19 https://doi.org/10.1007/978-3-319-41528-4_1 。Chatterjee, K., Novotný, P., Zikelic, D.：用于概率终止的随机不变式。载于：POPL会议论文集（2017 https://doi.org/10.1145/3009837.3009873 ）。20 Chen, J., He, F.：通过ω-正则分解证明几乎必然终止。载于：PLDI会议论文集（2020 https://doi.org/10.1145/3385412.3386002 ）。Chou, Y., 21 Yoon, H., Sankaranarayanan, S.：使用贝叶斯估计和可达性分析对车辆模型进行预测性运行时监控。载于：IROS会议论文集（2020 https://doi.org/10.1109/IROS45743.2020.9340755 ）。Cox, D.A., Little, J., O'Shea, D.：《理想、簇与算法——计算代数几何与交换代数导论》（第2版）。施普林格（1997 https://doi.org/10.1137/1035171 ）。Everest, G., van der Poorten, A.J., Shparlinski, I.E., Ward, T.：《递推数列》。数学调查与专著，美国数学学会，普罗维登斯（2003），ISBN 978-0-8218-3387-2。24. Farzan, A., Kincaid, Z.：组合式递推分析。载于：FMCAD会议论文集（2015 https://doi.org/10.1109/FMCAD.2015.7542253 ）。Feng, Y., Zhang, L., Jansen, D.N., Zhan, N., Xia, B.：为概率程序寻找多项式循环不变量。载于：ATVA会议论文集（2017）。26 https://doi.org/10.1007/978-3-319-68167-2_26 。Ghahramani, Z.：概率机器学习与人工智能。《自然》（2015 https://doi.org/10.1038/nature14541 ）。27 Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.：目标越低，难度越大：概率程序验证中下界的归纳法。ACM程序设计语言汇刊（POPL）（2020 https://doi.org/10.1145/3371105 ）。Humenberger, A., Jaroschek, M., Kovács, L.：Aligator.jl——用于循环不变式生成的Julia软件包。载于：CICM会议论文集（2018）。29 https://doi.org/10.1007/978-3-319-96812-4_10 。Humenberger, A., Jaroschek, M., Kovács, L.：具有多项式赋值的多路径循环的不变式生成。载于：VMCAI会议论文集（2018）。30 https://doi.org/10.1007/978-3-319-73721-8_11 。Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.：用于推理概率程序期望运行时间的最弱前置条件推理。载于：ESOP会议论文集（2016）。31 https://doi.org/10.1007/978-3-662-49498-1 。Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.：用于推理随机化算法期望运行时间的最弱前置条件推理。《ACM学报》（2018）。32 https://doi.org/10.1145/3208102 。Katoen, J., McIver, A., Meinicke, L., Morgan, C.C.：概率程序的线性不变式生成：基于证明的方法的自动化支持。载于：SAS会议论文集（2010 https://doi.org/10.1007/978-3-642-15769-1_24 ）。Katoen, J., 33 Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.：概率模型检查器MRMC的内幕。《性能评估》（2011）。34 https://doi.org/10.1016/j.peva.2010.04.001 。Kauers, M.：《非线性高阶差分方程算法》。博士论文，RISC，约翰内斯·开普勒大学，林茨（2005）。35. Kauers, M., Nuspl, P., Pillwein, V.：c^2-有限数列的阶数界。载于：ISSAC会议论文集（2023 https://doi.org/10.1145/3597066.3597070 ）。Kauers, M., Paule, P.：36 《具体的四面体——符号和、递推方程、生成函数与渐近估计》。施普林格（2011）。https://doi.org/10.1007/978-3-7091-0445-3

37. Kincaid, Z., Cyphert, J., Breck, J., Reps, T.W.：用于不变式综合的非线性推理。ACM程序设计语言汇刊（POPL）（2018）。https://doi.org/10.1145/3158142 38。Klinkenberg, L., Batz, K., Kaminski, B.L., Katoen, J., Moerman, J., Winkler, T.：LOPSTR会议论文集（2020 https://doi.org/10.1007/978-3-030-68446-4_12）。39 Kofnov, A., Moosbrugger, M., Stankovic, M., Bartocci, E., Bura, E.：具有非多项式赋值的概率循环的矩不变式。载于：QEST会议论文集（2022 https://doi.org/10.1007/978-3-031-16336-4_1）。Kofnov, A., 40 Moosbrugger, M., Stankovič, M., Bartocci, E., Bura, E.：具有非多项式赋值的概率循环的精确与近似矩推导。《ACM建模与仿真汇刊》（2024）。https://doi.org/10.1145/3641545，已录用。41. Kovács, L.：关于p可解循环的代数推理。载于：TACAS会议论文集（2008 https://doi.org/10.1007/978-3-540-78800-3_18）。Kozen, D.：概率PDL。42 《计算机与系统科学杂志》（1985） https://doi.org/10.1016/0022-0000(85)90012-1 43。McIver, A., Morgan, C.：《概率系统的抽象、精化与证明》。施普林格（2005 https://doi.org/10.1007/b138392）。Meyer, F., Hark, M., 44 Giesl, J.：使用期望规模推断概率整数程序的期望运行时间。载于：TACAS会议论文集（2021）。https://doi.org/10.1007/978-3-030-72016-2_14 45。Moosbrugger, M., Bartocci, E., Katoen, J.P., Kovács, L.：多项式概率程序的自动终止性分析。载于：ESOP会议论文集（2021）。https://doi.org/10.1007/978-3-030-72019-3_18 46。Moosbrugger, M., Bartocci, E., Katoen, J.P., Kovács, L.：概率终止工具Amber。载于：FM会议论文集（2021）。https://doi.org/10.1007/978-3-030-90870-6_36 47。Moosbrugger, M., Bartocci, E., Katoen, J., Kovács, L.：概率终止工具Amber。《形式化方法与系统设计》（2022）。https://doi.org/10.1007/s10703-023-00424-2 48。Moosbrugger, M., Müllner, J., Kovács, L.：概率循环的自动录敏序分析。载于：iFM会议论文集（2023）。https://doi.org/10.1007/978-3-031-47705-8_2 49。Moosbrugger, M., Stankovic, M., Bartocci, E., Kovács, L.：此刻正是讨论概率循环的时候。ACM程序设计语言汇刊（OOPSLA2）（2022）。https://doi.org/10.1145/3563341 50。Motwani, R., Raghavan, P.：《随机化算法》。剑桥大学出版社（1995 https://doi.org/10.1017/cbo9780511814075）。51 Müllner, J., Moosbrugger, M., Kovács, L.：强不变量很难：关于（概率）程序最强多项式不变量的难度。ACM程序设计语言汇刊（POPL）（2024 https://doi.org/10.1145/3632872）。Ngo, 52 V.C., Carbonneaux, Q., Hoffmann, J.：有界期望：概率程序的资源分析。载于：PLDI会议论文集（2018）。https://doi.org/10.1145/3192366.3192394 53。Rodríguez-carbonell, E., Kapur, D.：自动生成多项式循环不变量：代数基础。载于：ISSAC会议论文集（2004）。https://doi.org/10.1145/1005285.1005324 54。Rodríguez-Carbonell, E., Kapur, D.：生成简单循环中的所有多项式不变量。《符号计算杂志》（2007）。https://doi.org/10.1016/j.jsc.2007.01.002

55. Schröer, P., Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.：概率程序的演绎验证基础设施。ACM 程序语言论文集（OOP-SLA 2 2023 https://doi.org/10.1145/3622870 ）（）。.
56
Selyunin, K., Ratasich, D., Bartocci, E., Islam, M.A., Smolka, S.A., Grosu, R.：神经编程：迈向信息物理系统中的自适应控制。载于：CDC 论文集（ 2015 https://doi.org/10.1109/CDC.2015.7403319 ）。. Stankovic, M.,
57
Bartocci, E., Kovács, L.：基于矩的贝叶斯网络性质分析。《理论计算机科学》（2022）。 https://doi.org/10.1016/j.tcs.2021.12.021
58
. Tao, T.：《结构与随机性》。美国数学会（2008），ISBN 0-8218-4695-7