Final Report

Verifying Output Distribution Equivalence for Rectangular Discrete Probabilistic Programs via the PGF transformer semantics

Cheng Peng (2020533068)

Finished on May 25, 2024

Abstract

Randomized behavior is ubiquitous and inevitable in real-world programs, especially in certain security-critical domains like cryptography and cyber-physics systems. However, the lack of the efficient algorithms and data structures for representing and manipulating distributions makes it difficult to automatically verify the correctness of probabilistic programs. Traditional methods are only capable of computing digests of probability distributions, e.g., the expectation and variance, which are insufficient.

Recently, the MOVES research team headed by Prof. Katoen Joost-Pieter at RWTH Aachen proposed probability generating function transformer semantics of probabilistic programs and devised automated verification techniques based the semantics. This novel approach preserves the entire distribution carried by programs, enabling precise reasoning about the behavior of probabilistic programs.

In this course project, we investigate the theoretical foundation of the aforementioned approach, the PGF transformer semantics, and develop proof-of-concept tools that can verify a simple program generates exactly the desired distribution.

1 Introduction

1.1 Motivation

- Randomness is pervasive, arising naturally and inevitably in certain context. For example, applications interacting with the physical world have to handle noisy input. Models of crowd dynamics often involves non-deterministic motions. Furthermore, randomness is a key to efficient algorithms and data structure.
- Probabilistic programs often function as the key component of security-critical systems, for example, cryptograpy libraries are bedrock of online banking systems. Therefore, it is necessary to verify the correctness of probabilistic programs.
- It is common to require that the output distribution of a program matches exactly with a pre-determined distribution. Unaware of tiny deviation from the desired distribution may lead to side-channel leak or introduce vulnerabilites. Previous approaches focusing on deriving expecation or

establishing bounds are thus insufficient, since they are not sensitive to small perturbations.

2 Problem Statement

3 The ReDiP Programming Language

3.1 The pGCL programming language

3.2 Restrictions on pGCL

3.3 Execution Tree Based Operational Semantics

4 PGF transformer semantics of ReDiP

4.1 Review on Probability Generating Functions (PGFs)

1D formal power series:

$$f = \sum_{i=0}^{\infty} f_i x^i$$

This can be extended to 2D scenario:

$$f = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} f_{i,j} x^i y^j$$

And even $k \in \mathbb{N}$ dimension, for indeterminate $\mathbb{X} = (X_1, X_2, \ldots, X_k)$:

$$f = \sum_{\sigma \in \mathbb{N}^k} f(\sigma)\mathbf{X}^\sigma \quad f : \mathbb{N}^k \to \mathbb{R}, X^\sigma = \prod_{i=1}^{k} X_i^{\sigma_i}$$

Rational PGFs: for $n, m \in \mathbb{N}$,

$$g = \frac{\sum_{i=0}^{n} A_i X^i}{\sum_{j=0}^{m} B_j X^j}$$

If a formal power series is the Taylor series of a rational function, we say that it is in closed form.

4.2 Distribution transformation of ReDiP

Theorem 4.1. *closed form preservation of ReDiP For a loop-free ReDiP program, and a rational PGF f/g, $[\![P]\!](f/g)$ is also a rational PGF.*

Proof. By induction on the program structure. □

5 Verifying Equivalence of linear PGF transformers

6 Evaluation

6.1 Experiment setup

6.2 Results

7 Discussion

7.1 Limitation of the approach

7.2 State of the method

7.3 Future research work

7.4 Related works