# Termination Analysis of Probabilistic Counter Programs

Sergei Novozhilov[1]
supervised by: Prof. Dr. Mingshuai Chen[2]
[1]snovozhilov@connect.ust.hk
[2]m.chen@zju.edu.cn

## 1 Introduction

The halting problem is undoubtedly the most famous unsolvable problem in the theory of computation. Roughly speaking, the problem is to determine, given the code of a program and an input variables, whether the program will run indefinitely or halt after a finite number of steps. The halting problem was first proven to be undecidable by Alan Turing in 1936. Despite the fact that the problem is undecidable in general, there are many techniques for solving it in specific cases, such as invariant generation or the ranking function method.

The introduction of *probabilistic choice* into programs changes the situation, as now the program execution is nondeterministic, and the time and the result of execution may differ from run to run which significantly compicate reasoning about the programs and finding the bugs.

**Example**

In the example, the **Coin**() function reruns 0 or 1 with equal probability. The code will work indefinitely with probability $1/2$ and halt with probability $1/2$.

```
1: x = Coin();
2: while (x > 0) {
3:     x = x + 1;
4: }
```

The discreptancy of the program behavior from run to run leads to a modified version of the halting problem and called ***A**lmost **S**ure **T**ermination* or **AST** for short. Furthermore, if the runtime of the program has finite expectation, then the program is said belong to the ***P**ositive **A**lmost **S**urely **T**eminating* commonly abbreviated as **PAST**.

**Definition (AST and PAST)**

For a class of probabilistic programs **PP** let $T$ be the runtime function which maps a program $p \in$ **PP** to a distribution over the extended set of natural numbers $\mathbb{N} \cup \{\infty\}$:

$$T : \textbf{PP} \to \mathcal{D}(\mathbb{N} \cup \{\infty\})$$

A program $p \in$ **PP** is said to terminate almost surely iff

$$\mathbb{P}[T(p) < \infty] = 1.$$

A program $p \in$ **PP** is said to terminate positively almost surely iff

$$\mathbb{E}[T(p)] < \infty.$$

It is evident from the definition that **PAST** $\subseteq$ **AST**, moreover it is known than **PAST** is the *proper subset* of **AST**, i.e. there are programs that are almost surely terminating but not positively almost surely terminating. The next example illustrates this fact.

**Example (PAST $\subset$ AST)**

This example simulates a one dimensional symmetric random walk.

```
1: fun RandomWalk1D() {
2:     x = 10;
3:     while (x > 0) {
4:         if (Coin() == 1) x = x + 1;
4:         else x = x - 1;
5:     }
6: }
```

From the theory of random walks, it is knownt that $\mathbb{P}[T(\text{RandomWalk1D}) < \infty] = 1$ but $\mathbb{E}[T(\text{RandomWalk1D})] = \infty$, which intuitively means that the program will halt eventually but we might need to wait for it for a long time.

As the probabilistic programs are the superclass of deterministic programs and thereofre the **AST**/**PAST** properties are also undecidable in general. There are two ways approach this problem: the first approach is to develop a technique which will be sound but incomplete (i.e. it will not be able to prove the termination for some programs but whenever it proves the termination, the answer is correct), and the second approach is to develop a simpler programming language which admit a decision procedure for the temination problem. We followed the second approach and developed a resrticted class of programs called ***P**robabilistic **C**ounter **P**rograms **(PCP)*** bulting on the idea of [2] of the constant probability programs.

We can summarize the main contributions of this work as follows:

- We introduce the new class of probabilistic programs called **PCP** toegether with the hierarchy of the classes of programs:

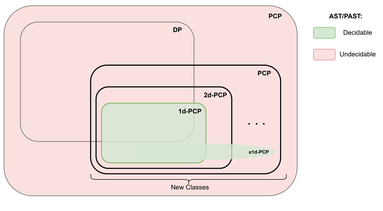
$$1d\textbf{-PCP} \subset 2d\textbf{-PCP} \subset 3d\textbf{-PCP} \subset \ldots \subset kd\textbf{-PCP} \subset \ldots \subset \textbf{PCP}$$

- For $1d$-**PCP** we constructed an efficient algorithm for deciding the **AST** and **PAST** property.

- We proved that **AST** and **PAST** properties are undecidable already for $2d$-**PCP**.

- However we showed that in significant nuber of cases the $kd$-**PCP** programs essentially behave as $1d$-**PCP** programs and we call this class $e1d$-**PCP** programs. For such programs the **AST** and **PAST** properties are decidable by the reduction to the $1d$-**PCP** case.

The following diagram shows the classes and the relations between them:




## 2 Background

The subject of our research is the probabilistic programs which can modify its variables in a very restricted way: namely, the programs are allowed only to change the variables values only by adding a fixed value to a variable. Moreover, we organize all such programs into a graded hierarchy of the classes, whenre the gradation is done based on the number of the variables used.

**Definition (PCP, $kd$-PCP )**

Probabilistic counter programs are the programs which can be written using the syntax:

| | |
|---|---|
| `init x = n;` | (counter initialization) |
| `x = x + n;` | (counter update) |
| `if (G) {B1} else {B2}` | (if conditions) |
| `while (G) {Body}` | (while loops) |
| `{B1} [p1] {B2}... [pk] {Bk+1}` | (probabilistic branching) |

Where `Bi` are blocks, the guards `G` are boolean expressions over the counters and the probabilistic branching construction chooses either of the branches with probability $p_i$.
If the program uses no more than $k$ counters, then it is called $k$-dimensional **PCP** or $kd$-**PCP** for short.

Notice that instead of using the **Coin**() function, we use the probabilistic branching construction. The programs using a coin and the probabilistic branching are equivalent and can be easily transformed to each other.

**Example**

Here we show how the previous example can be transformed into the $1d$-**PCP**.

```
1: fun RandomWalk1D() {
2:     x = 10;
3:     while (x > 0) {
4:         {x = x + 1} [1/2] {x = x - 1}
5:     }
6: }
```

The following program is an example of the $4d$-**PCP** which simulates a process of collecting of $4$ coupons, where each step we are getting a new coupon one of the $4$ types with equal probability. The program will halt after we collect all $4$ coupons:

```
1: init x1 = 0, x2 = 0, x3 = 0, x4 = 0;
2: while (x1 + x2 + x3 + x4 < 4) {
3:     {x1 = 1} [1/4] {x2 = 1} [1/4]
4:     {x3 = 1} [1/4] {x4 = 1}
5: }
```

In the line of our approach, we compile the programs into the *counter probabilistic transition system* which are play the same role for the probabilistic programs as the control flow graphs for the deterministic programs.

**Definition (CPTS)**

Counter Transition System is a tuple $(L, x, T)$ where $L$ is the set of locations, $x \in \mathbb{Z}$ is the variables vector, $T \subset S \times \mathcal{P}(x) \times \mathbb{Z} \times [0,1] \times S$ is a set of transitions, where for $t = (l_i, g, \text{upd}, p, l_j)$:

- $l_i, l_j \in L$ are the source and target locations
- $g(x) \in \mathcal{P}(x)$ is a predicate over the variables $x$, each consists of boolean connectives $\{\vee, \wedge, \neg\}$, algebraic predicates $\{\geq, \leq, >, <, =, \neq\}$, arithmetical functions $\{+, -, \cdot, (-\bmod m)\}$;
- upd $\in \mathbb{Z}^k$ is an update vector, which is applied to the variables $x$ after the transition;

Also we need a classical notion of the Markov chain to define the semantics of the probabilistic programs.

**Definition (MC)**

Countable state Markov chain is a tuple $(S, P)$ where $S$ is the set of states, $P : S \times S \to [0,1]$ is the transition matrix.

## 3 Decision Procedure for $1d$-PCP

The decision procedure for the one dimensional case is based on the sequence of reductions, first we reduce a $1d$-**PCP** to the $1d$-**CPTS**, then we reduce the $1d$-**CPTS** to the **MC**. Then we prove that the Markov chain can be decomposed into a regular infinite Markov chain and a finite irregular Markov chain:

$$M = M_{\text{reg}} \sqcup M_{\text{irr}},$$

where the regualr part can be analyzed using classical one dimentional random walks theory and the irregular part is finite and can be handled by the existing probabilistic model checking techniques.

**Theorem ($1d$-PCP Termination)**

The **AST** and **PAST** properties are decidable for the $1d$-**PCP** programs.

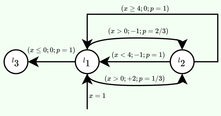
We demonstrate the decision procedure on a simple example.

**Example**

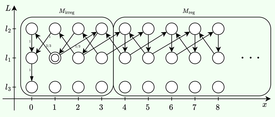
Consider a program

```
1: init x = 1;
2: while (x > 0) {
3:     {x = x - 1;} [2/3] {x = x + 2;}
4:     if (x < 4) {x = x - 1;}
5: }
```

The program is equivalent to the following **CPTS**:




Which is then converted to a countable state Markov chain which further is decomposed into finite state irregular part and infinite state regular part:




The regular part is analyzed using the random walks theory and the irregular part is analyzed using the finite state probabilistic model checking techniques, after what we can conclude that the program is **AST** but not **PAST**.

## 4 Undecidability of $2d$-PCP

Apparanetly the decision procedure for the $1d$-**PCP** cannot be extended to the higher dimensional counter programs.

**Theorem ($\geq 2d$-PCP Termination)**

The **AST** and **PAST** properties are undecidable for the $kd$-**PCP** programs whenever $k \geq 2$.

The proof is based on the reduction from a turing complete deteministic model of computation based on two register and two types of instructions: inc and dec. The undecidability of such a minimalistic model was proved in [1].

## 5 Conclusion

In this work, we explored the termination analysis of probabilistic counter programs (**PCP**). By examining the class of **PCP** programs, we identified a hierarchy based on the number of counters. We proved the decidability results of the **AST** and **PAST** properties of $1d$-**PCP** class and undecidability results for $kd$-**PCP** class for $k \geq 0$.

The future work consisits of automatizing the proposed approach and devising a procedure for automatic variables reduction for the briefly described $e1d$-**PCP** class.

## References

[1] A. Dudenhefner, "Certified Decision Procedures for Two-Counter Machines," in *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, 2022, doi: 10.4230/LIPIcs.FSCD.2022.16.

[2] J. Giesl, P. Giesl, and M. Hark, "Computing Expected Runtimes for Constant Probability Programs," in *Automated Deduction – CADE 27*, Springer International Publishing, Cham, 2019, isbn: 978-3-030-29436-6.

**Acknowledgements:**

This project was supported by the GripS program. The author expresses his gratitude to Prof. Dr. Mingshuai Chen for his guidance and invaluable discussions throughout the project.