Probabilistic Programs — A Natural Model for Approximate Computations

Nils Jansen, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo

Software Modeling and Verification Group, RWTH Aachen University
{nils.jansen,benjamin.kaminski,katoen,matheja,federico.olmedo}@informatik.rwth-aachen.de

Approximate computing is a computational paradigm in which the requirement of having fully deterministic and accurate arithmetical operations is dropped [4]. For diverse practical problem (e.g. video decoding or image compression) by relaxing this requirement one can obtain considerably more efficient algorithms—be it in terms of execution time, memory usage, energy consumption, etc.

When moving from accurate to approximate computations, a natural question arises: *How accurate* is the result of the approximate computation compared to the result of the accurate one? Any attempt to seriously answer this question raises the need for an appropriate formal model of approximate computations. *Probabilistic programs* extend conventional programs with probabilistic choices or random assignments and indeed provide a well–studied and suitable model for this purpose. Their syntax and semantics [5, 6, 3, 2] are flexible and expressive enough to model many aspects of approximate computing quite naturally.

A recent approach for dealing with quantitative reliability analysis of approximate computations brought up the concept of *unreliable operators* [1]. In this approach instead of performing e.g. the accurate assignment `x = x + y`, one performs the approximate assignment `x = x +. y`, the latter being specified through a *reliability* value $0 \leq \varepsilon \leq 1$. Then with probability $1-\varepsilon$ the approximate assignment yields the expected effect, while with probability ε it produces an *arbitrary*—and thus erroneous—effect. This model, however, is unable to quantify how much the erroneous effect departs from the accurate one.

Our goal is to take this reliability analysis a step further by employing a more expressive language of probabilistic programs: the so-called probabilistic guarded command language (pGCL) [6]—an extension of Dijkstra's guarded command language with probabilistic choices. By employing this language besides representing the probability with wich erroneous computation occurs, we can provide more stochastic information such as the most likely effect or the expected (i.e. average) effect of the erroneous computations. For instance we can model an approximate assignment as a pGCL program

$$\{\mathtt{x = x + y}\}\ [1-\varepsilon]\ \{\mathtt{x = x + y + 1}\}\ .$$

The effect of this approximate assignment is to be interpreted as follows: With probability $1-\varepsilon$ the accurate assigment `x = x + y` is performed. With the remaining probability ε the assignment has an erroneous effect but it is *not arbitrary*: In the erroneous case the assignment `x = x + y + 1` is performed instead.

In this richer model we can reason beyond the mere reliability of the approximate assignment: We can for instance conclude that the expected value of x after the assignment is $\mathtt{x} + \mathtt{y} + \varepsilon$ or that its conditional expected value given that an error occurred during the assignment is $\mathtt{x} + \mathtt{y} + 1$ (in particular not arbitrary). We believe that this more fine–grained modelling has a potential to further developments in the quantitative reliability analysis of approximate computations and provides a promising blend between the approximate computations and probabilistic program verification communities.

References

1. Carbin, M., Misailovic, S., Rinard, M.C.: Verifying quantitative reliability for programs that execute on unreliable hardware. In: Proc. of OOPSLA 2013. pp. 33–52. ACM (2013)
2. Gretz, F., Jansen, N., Kaminski, B.L., Katoen, J., McIver, A., Olmedo, F.: Conditioning in probabilistic programming. CoRR abs/1504.00198 (2015)
3. Gretz, F., Katoen, J.P., McIver, A.: Operational versus Weakest Pre–Expectation Semantics for the Probabilistic Guarded Command Language. Performance Evaluation 73, 110–132 (2014)
4. Han, J., Orshansky, M.: Approximate computing: An emerging paradigm for energy-efficient design. In: Test Symposium (ETS), 2013 18th IEEE European. pp. 1–6 (May 2013)
5. Kozen, D.: Semantics of Probabilistic Programs. J. Comput. Syst. Sci. 22(3), 328–350 (1981)
6. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer (2004)