Slicing of Probabilistic Programs Based on Specifications

Marcelo Navarro
Computer Science Department (DCC), University of Chile, Santiago, Chile

Federico Olmedo
Computer Science Department (DCC), University of Chile, Santiago, Chile

Abstract

We present the first slicing approach for probabilistic programs *based on specifications*. Concretely, we show that when probabilistic programs are accompanied by their functional specifications in the form of pre- and post-condition, one can exploit this semantic information to produce specification-preserving slices *strictly more precise* than slices yielded by conventional techniques based on data/control dependency.

To illustrate this, assume that Alice and Bob repeatedly flip a fair coin until observing a matching outcome, either both heads or both tails. However, Alice decides to “trick” Bob and switches the outcome of her coin, before comparing it to Bob’s. The game can be encoded by the program below, which is instrumented with a variable n that tracks the required number of rounds until observing the first match. The program terminates after K loop iterations with probability $1/2^K$ provided $K > 0$, and with probability 0 otherwise, satisfying the annotated specification.

```
\\ pre: $\frac{1}{2^K}[K > 0]$
$n := 0$;
$a, b := 0, 1$;
while $(a \neq b)$ do
   $n := n + 1$;
   $\{a := 0\}\ [1/2]\ \{a := 1\}$;
   $a := 1 - a$;
   $\{b := 0\}\ [1/2]\ \{b := 1\}$
\\ post: $[n = K]$
```

Traditional slicing techniques based on data/control dependencies conclude that the only valid slice of the program (w.r.t. output variable n) is the very same program. However, our slicing approach allows removing the assignment $a := 1 - a$ from the loop body, while preserving the program specification.

At the technical level, our slicing technique works by propagating post-conditions backward using the greatest pre-expectation transformer – the probabilistic counterpart of Dijkstra’s weakest pre-condition transformer. This endows programs with an axiomatic semantics, expressed in terms of a verification condition generator (VCGen) that yields quantitative proof obligations.

In particular, we design (and prove sound) VCGens for both the partial (allowing divergence) and the total (requiring termination) correctness of probabilistic programs, making our slicing technique *termination-sensitive*. To handle iteration, we assume that program loops are annotated with invariants. To reason about (probabilistic) termination, we assume that loop annotations also include (probabilistic) variants.

Another appealing property of our slicing technique is its *modularity*: It yields valid slices of a program from valid slices of its subprograms. Most importantly, this involves only *local reasoning*.

Besides developing the theoretical foundations of our slicing approach, we also exhibit an algorithm for computing program slices. Interestingly, the algorithm computes the *least* slice that can be derived from the slicing approach, according to a proper notion of slice size, using, as main ingredient, a shortest-path algorithm.

Finally, we demonstrate the applicability of our approach by means of a few illustrative examples and a case study from the probabilistic modeling field.

2012 ACM Subject Classification Theory of computation → Probabilistic computation; Theory of computation → Program specifications; Software and its engineering → Designing software

Keywords and phrases probabilistic programming, program slicing, expectation transformer semantics, verification condition generator

Digital Object Identifier 10.4230/LIPIcs.ECOOP.2022.34

Category Extended Abstract

Related Version *Full Version*: `https://doi.org/10.1016/j.scico.2022.102822`

Funding This research has been supported by the FONDECYT Grant No. 11181208.