Static Analysis of Programs with Imprecise Probabilistic Inputs

Assale Adje\textsuperscript{2}, Olivier Bouissou\textsuperscript{1}, Jean Goubault-Larrecq\textsuperscript{2}, Eric Goubault\textsuperscript{1}, and Sylvie Putot\textsuperscript{1}

\textsuperscript{1} CEA LIST
CEA Saclay, 91191 Gif-sur-Yvette CEDEX, France
firstname.lastname@cea.fr
\textsuperscript{2} LSV, ENS Cachan
61 avenue du président Wilson, 94230 Cachan, France
lastname@lsv.ens-cachan.fr

Abstract. Having a precise yet sound abstraction of the inputs of numerical programs is important to analyze their behavior. For many programs, these inputs are probabilistic, but the actual distribution used is only partially known. We present a static analysis framework for reasoning about programs with inputs given as imprecise probabilities: we define a collecting semantics based on the notion of previsions and an abstract semantics based on an extension of Dempster-Shafer structures. We prove the correctness of our approach and show on some realistic examples the kind of invariants we are able to infer.

1 Introduction

Static analysis of embedded softwares faces the difficulty of correctly and precisely handling the program inputs. These inputs are usually given by sensors that measure a physical value continuously evolving with time. The classical abstraction of such inputs is to assign them with the range of values that the sensor may measure: in this way, we obtain a non-deterministic over-approximation of the values of the inputs which is then propagated through the program.

However, in addition to this non-deterministic abstraction of the values, we often have a probabilistic information on where the inputs lie in the range of possible values. This probabilistic information may be given by some knowledge on the physical environment with which the program interacts, or may be introduced as noise by the sensor. This noise can be very often modeled as a random variable with a Gaussian law; the value of the inputs is then given by $V + \varepsilon$ where $V$ is a non-deterministically chosen value and $\varepsilon$ is the probabilistic noise.

In this article, we present a framework to analyse deterministic programs with both probabilistic and non-deterministic inputs. In Section 2 we motivate our use of previsions and Probability-boxes. In Section 3 we define our concrete semantics based on previsions and in Section 4 we present our abstract semantics based on probabilistic affine forms. We prove its correctness in Section 5 and show in Section 6 the kind of invariants we are able to compute on realistic examples. Let us remark that to ease the understanding of this article, we have omitted

© Springer-Verlag Berlin Heidelberg 2014
various technical details such as the use of floating-point numbers or the impact of run-time errors on the semantics. We discuss these points in the course of the exposition.

**Running Example.** In this article, we use a linear, order 2 filter to illustrate both our concrete and abstract semantics. This filter is given by the loop:

```c
while(1) {
    y = 0.7*x - 1.3*x0 + 1.1*x1 + 1.4*y0 - 0.7*y1;
    x1 = x0; x0 = x; y1 = y0; y0 = y;
    x = input();
}
```

Numerical filters are very important for the analysis of embedded softwares as they are present in (almost) every software that must handle data coming from sensors. Computing the range of values reachable by the output variable $y$ is a challenge addressed by many techniques [15]. However, all these methods assume that the inputs $x$ (given by the function `input()` in the program) are bounded within a certain range and do not assume any distribution of the values within this range. Here, we assume that the input variables follow some probability distribution but we do not know which: we assume that $x$ follows a uniform law on the range $[-A, A]$ for some $A \in [0, 0.2]$. Moreover, we assume that the distribution of the inputs may change during the execution of the filter, i.e. the distribution of input $x$ read at iterate $n$ (represented in the program by $x$) is not the same as the one of $x$ read at iterate $n-1$ (represented in the program by $x1$). We however know that both are uniform distribution on some range $[-A, A]$. We ran 10 simulations of this filter and show below the 10 distributions in cumulative form (CDF) for the output variable $y$. Our goal is to compute guaranteed yet precise bounds on this set of distributions.

![CDF.png](attachment://CDF.png)

**Contribution.** In this paper, we present three main results. First, we define a semantics for imperative programs with inputs defined as imprecise probabilities. We define an operational and denotational semantics based on previsions and show their equivalence. Next, we define a new abstract domain based on probabilistic affine forms and especially new join and meet operators. Finally, we prove the correctness of the abstract semantics w.r.t the concrete ones and give some hints on how to adapt it to the analysis of hybrid systems by showing on one example how we can solve ODEs with our domain.