Static Analysis of the Accuracy and Robustness of Finite Precision Implementations

Eric Goubault and Sylvie Putot
CEA, LIST

**ZONOTOPIQUE ABSTRACT DOMAINS**

We developed a version adapted to static analysis of the affine forms introduced by Comba et Stolfi in 1993. These forms describe the sets of values taken by affine combinations of finitely many independent symbolic variables called “noise symbols” \( \{ \theta_i \} \) taking their values in the interval \([-1, 1]\). The concretisations of such forms as subsets of \( \mathbb{R}^n \) yield a classical class of polyhedra, with a central symmetry called zonotopes.

![Figure 1: Zonotopic concretization in \( \mathbb{R}^n \)](image)

However, the explicit parametrisation by affine forms shows a strong analogy with the Taylor methods used in guaranteed numerical computations or hybrid systems analysis. Abstract transfer functions are time and space efficient, and provide functional abstractions (i.e. abstraction of the input-output relationship), as in e.g. the interpretation of the expression \( y = x^2 - x \) where \( x \in [-1,1] \). Variable \( x \) is abstracted by \( \bar{x} = 5 + \hat{x} \) and function \( x \rightarrow y = x^2 - x \) is abstracted as:

\[
y = \frac{2 \bar{x} + 50 \hat{x} + 12 \hat{x}^2}{2} = 17 \bar{x} + 10 \bar{x} \hat{x} + 12 \hat{x}^2
\]

The red line below is the linearization of the function (blue dashed line) and the green zonotope is the concretization of \((\bar{x}, \hat{x})\). Its width is given by the green term, \( \hat{x} \).

![Figure 2: Functional interpretation](image)

We also designed efficient abstract join and meet operations, useful for interpreting tests, and for computing least fixed points, i.e. invariants.

**INVARIANT SYNTHESIS**

The domain is expressive enough to synthesise invariants for many stable recurrent sequences such as the linear recursive filter of order 2:

\[
S_{n+2} - 0.7 \bar{S}_{n+1} + 1.32 \bar{S}_n + 1.1 \bar{S}_{n+1} + 1.4 \bar{S}_{n+1} + 0.7 \bar{S}_n
\]

for \( \bar{S}_n \) inputs in \([-1,1]\). Here, Fluctuat is able to determine a zonotopic invariant set of the \( S_n \) iterates, as shown on the graphic interface below.

![Figure 3: Invariant set for tuples \( (S_n, S_{n+1}, S_{n+2}) \)](image)

**DISCONTINUITY AND ROBUSTNESS ANALYSIS**

A recurrent difficulty when we want to assess the influence of finite precision is, the possibility for a test to be unstable: when, for a given input, the finite precision control flow can differ from the control flow that would be taken by the same execution in real numbers. Not taking this possibility into account may be unsafe if the difference of paths leads to a discontinuity in the computation. But taking it into account without special care soon leads to large over-approximations.

For instance, we consider a Householder scheme to approximate the square root function. Fluctuat shows a discontinuity error (in purple) on the stepping criterion of the loop, signalling that it may not converge in the same number of iterations in finite precision and in real numbers.

![Figure 4: Rounding error amplified by cancellation](image)

Uncertainties can also be attached to inputs and parameters, with a special language of assertions, and propagated through computations.

**INVARIANT SYNTHESIS**

Our analysis computes unstable tests conditions as the intersection of the constraints on noise symbols induced by the differing path conditions, and deduces tight bounds on the difference between the computation in both branches. We automatically characterize conditional blocks that perform a continuous treatment of inputs, and are thus robust, and those that do not, circumventing the path-space explosion problem that a naive approach would run into.

![Figure 5: Discontinuity error due to the stopping criterion](image)

**FUNCTIONAL PROOF**

We can also prove functional properties, both in the real number semantics (method error) and in the finite precision semantics (implementation error), for instance here on the Householder scheme for square root.

**ACCURACY OF FINITE PRECISION IMPLEMENTATIONS**

Fluctuat considers both real number and finite precision arithmetics, bounds the errors due to the finite precision implementation and traces the source of errors. When a program variable is selected, a graph is displayed, representing the bound for the contribution of each program line on the error on this variable at the end of the program.

For instance below, we see that variable \( x \) is very inaccurately computed in floating-point numbers, and that the main source of error comes from the fact that \( 0.7 \) is not exactly representable. It is propagated through computations (cancellation here).

**HYBRIDFLUCTUAT: extension to hybrid systems analysis**

With a connection to a guaranteed ODE solver, we can also analyse hybrid systems such as control systems. The discrete control software is analyzed together with a physical continuous environment, described by switched ODE systems. This helps for instance proving the stability of a closed loop system, that otherwise, in open loop, would be unstable.

The solutions of the ODE systems are abstracted by a call to a guaranteed integration tool, that computes an over-approximation of the actual solution of the ODE. The interaction between the program and the environment are modelled by assertions in the program. The program can change parameters of modes of the ODE system at time \( t \) (HYB2D_PARAM, modelling actuators). Conversely, the program can request the value of a continuous variable at time \( t \) (HYB2D_VALUE, modelling sensors).

**EXTENSIONS**

We developed some extensions of the zonotopic abstract domains, still relying on the affine forms parametrisation \( x = \sum x_i \hat{x}_i \).

- **Inner-approximation:** with interval or zonotopic coefficients \( x_i \) and intervals in in Kaiser arithmetic, we get generalized affine sets for inner-approximation. We can use them to determine sets of values of the outputs, that are sure to be reached for some inputs in the specified ranges.
- **Probabilistic affine forms:** with noise symbols \( \eta_i \) and coding sets of probability distributions (P-boxes, Dempster-Shafer structures on intervals), we get probabilistic affine forms. They model both non-deterministic and probabilistic uncertainties, and deliver guaranteed sets of distribution of program variables. For instance, on the linear filter example, we can prove that reachable values for the output are deterministically in \([-0.9,2.7]\), but outside \([-0.25,1.75]\) only with very low probability.

**SELECTED PUBLICATIONS**