Approximating floating-point operations to verify numerical programs

Mohammed Said BELAID, Claude MICHEL and Michel RUEHER

I3S-CNRS
2000, route des Lucioles, BP 121
06903 Sophia Antipolis Cedex - France
{MSBelaid, Claude.Michel}@i3s.unice.fr, Michel.Rueher@gmail.com

Abstract

The verification of programs performing floating-point computations is a key issue in the development of critical software. Reasoning on floating-point computations is a tricky task that requires dedicated tools: floating-point arithmetic is not correctly handled by solvers over the reals. Several methods have been developed in order to verify programs working with floating-point numbers. Some of them misinterpret the floating-point numbers as fixed point numbers [2], which is far from being safe. Others try to ensure the absence of runtime errors by over-approximating floating-point computations [3]. The late method is safe, but unfortunately rejects many valid programs.

We rely on constraint programming techniques to generate test cases [5] or to verify the conformity of a program with its specification [6]. Constraint programming offers many benefits like the capability to deduce information from partially instantiated problems or to exhibit (counter)examples. Floating-point constraint solvers are available [4] but have some difficulties to handle large pieces of software. Part of these limitations roots in the poorness of floating-point arithmetic. That is why we introduce here a new method to solve constraints over the floating-point numbers which takes advantage of solvers over the reals. The basic idea is to build safe but tight approximations over the reals of the floating-point operations. To ensure the tightness of the approximations, each floating-point operation is approximated according to its rounding mode. For example, assume that \( x \) and \( y \) are positive normalized floating-point numbers, then the floating-point addition \( x \oplus y \) with a rounding mode set to \( -\infty \), is bounded by

\[
\alpha \times (x + y) < x \oplus y \leq x + y
\]

where \( \alpha = \frac{1}{1 + 2^{-p+1}} \) and \( p \) is the size of the significant. Approximations for special cases have also been refined, e.g., for the addition with a rounding mode set to zero, or for the multiplication by a constant.

Thanks to these approximations, a problem over the floating-point numbers is translated into a set of constraints over the reals. This set of

*This work was partially supported by the ANR-07-SESUR-003 project CAVERN.
constraints can directly be solved by available solvers over reals, and thus, is relieved from the burden of floating-point arithmetic. Note that these approximations can also be used to improve variable domain reductions in other approaches than constraint programming.

References:


**Keywords:** floating-point numbers, program verification, safe approximations.