Dedicated Search Strategies For Finding Critical Counterexamples In Programs With Floating Point Computations*

Claude Michel
Université Côte d'Azur, CNRS, I3S, France
Email: claude.michel@i3s.unice.fr

Michel Rueher
Université Côte d'Azur, CNRS, I3S, France
Email: michel.rueher@i3s.unice.fr

***Abstract*—Constraint Programming (CP) is an efficient technique for searching counter-examples that violate a property of the program to verify. However, the search process can become very long and costly when the program to check contains Floating Point (FP) computations. In this paper we discuss the capabilities of a set of variable choice strategies and subdomain selection strategies that take advantages of the very specific properties of the floats. We also outline the capabilities of a CP techniques for computing ranges of error values as well as input values that exercise a given error; a critical issue to take advantage of cancellation phenomena when searching counterexamples.**

***Index Terms*—: counterexamples, floating point computations, constraint programming, search strategies, absorption, cancellation**

I. Introduction

Programs with FP computations control complex and critical systems in numerous domains, including cars and other transportation systems, nuclear energy plants, or medical devices. FP computations are derived from mathematical models on real numbers [10], but computations on FP numbers are different from computations on real numbers. For instance, some real numbers like 0.1 do not have an exact representation. FP arithmetic operators are neither associative nor distributive, and may be subject to phenomena such as absorption[1] and cancellation[2] [13].
As a consequence, the flow of a very simple program over the floats ($\mathbb{F}$) can differ from the expected flow over the reals ($\mathbb{R}$). Such a flow discrepancy might have critical consequences if, for instance, the condition it relates to is braking in an ABS system.

II. Dedicated search strategies

CP [4], [5] has been used to deal with such problems, but the search for a counter-example remains long and costly when the program to check contains floats. This stems from the fact that existing search strategies have been designed for discrete domains and, to a lesser extent, continuous domains. However, integer and real strategies are poorly adapted to floats because the set of floats is finite but its cardinality is very high and the distribution of floats is not at all uniform (half of the floats are in the range[-1,1]).

A. Variable selection strategies

In [15] we have defined dedicated variable selection strategies that take advantage of the absorption phenomena and the non-uniformity of the density.
Intuitively, density captures the proximity of floating point values within a given domain. It helps to identify domains that have a small number of values on a big domain or a big number of values on a small domain (with respect to the width). Likewise, we have also defined a property that helps to identify potential absorption phenomena. Consider a floating point addition constraint $z_{\mathbb{F}} = x_{\mathbb{F}} \oplus y_{\mathbb{F}}$ in which the rounding more is set to "round to nearest even". If the domain $x_{\mathbb{F}}$ has a significantly larger magnitude than $y_{\mathbb{F}}$, some values in $y_{\mathbb{F}}$ may simply be absorbed when carrying out the addition. Measuring which fraction of $y_{\mathbb{F}}$ is obliterated in this way is the purpose of the absorption property we have defined.
Combining wisely two properties can also be helpful to select useful solutions. For instance, `densWabs` combination improves the *density* property while selecting variable the domain of which are likely to be involved in an absorption phenomena, whereas `absWdens` selects the variable that maximizes absorption among the subset of variables that satisfies some density.
On a small set of benchmarks from test and verification of floating point software, `MaxAbs` and `Maxdens` outperformed the standard strategy by a factor of more than 110. These performances are even better for problems with solution. On the other hand, the improvement for benchmarks without solution is only 4 times. `densWabs` and `absWdens` were only very efficient on problems with solutions.

* This is joined work with Claude Michel, Heytem Zitoun, Remy Garcia and Laurent Michel. This work was also partially supported by ANR COVERIF (ANR-15-CE25-0002).

[1]Absorption occurs when adding two floating point numbers with different order of magnitude. The result of such an addition is the furthest from zero.

[2]Cancellation occurs when most of the most significant bits are lost by subtracting the close results of two operations.

B. Subdomain selection strategies

We also defined a new sub-domain selection strategy that takes advantage of absorption[16]. The goal of this strategy is to concentrate on the most relevant absorption phenomena, in other words, giving priority to the subdomains of x and y most likely to lead to an absorption. After selection of variable `x`, strategy $MaxAbs$ examines addition and subtraction constraints to select a variable `y`, the values of which are most absorbed by the values of `x`. Coordinated branching on these variables is performed to exploit the latent absorption.
On a set 49 benchmarks from test and program verification (SMTLib [1], FPBench [6], and CBMC [3] (but also [4], [7]), the combination of $MaxAbs$ and the classical $MaxOccurences$ strategy was significantly better than standard strategies.

III. capabilities of a CP solver for computing ranges of error

None of the above strategy could take advantage of cancellation phenomena because we couldn't compute an accurate error.

Floating point computation errors have been the subject of many works based on an overestimation of actual errors, e.g., Fluctuat [9] that combines affine arithmetic and zonotopes, PRECiSA [14] that relies on static analysis. The point is that all these approaches compute an error estimation that can hardly be used to identify catastrophic cancellation problems or to compute input values that exercise a given error [11]. In order to overcome this lack of reasoning capabilities and to enhance the analysis of errors, we propose to incorporate in a constraint solver over the floats [16], [12], [2], the domain of errors which is dual to the domain of values. Both domains are associated with each variable of the problem. Computation errors form a new dimension to consider. They require a specific domain in view of the distinct nature of elements to represent, but also, due to the possible values of errors which belong to the set of reals. Therefore, we introduce a domain of errors, which is associated with each variable of a problem. Since all arithmetic constraints processed here are reduced to the four basic operations, and since those four operations are applied on floats, i.e. a finite subset of rationals, this domain can be defined as an interval of rationals with bounds in $\mathbb{Q}$. Another domain of errors is required for the smooth running of our system: it is the domain of errors on operations that appears in the computation of the deviations. Contrary to previous domains, it is not attached to each variable of a problem but to each *instance* of an arithmetical operation of a problem.

Like the domain of errors attached to a variable, it takes values in the set of rational numbers. Projection functions and constraints over errors are being evaluated in a prototype based on Objective-CP. Preliminary experiments [8] are promising and will naturally be reinforced with more benchmarks.

IV. Discussion

Experiments with variable selection strategies and subdomain selection strategies taking advantage of some specificities of floats like the density and the absorption phenomena were quite successful on a significant set of hard benchmarks. The role of the CP solver for computing ranges of error is to provide the information required to develop strategies that also take advantage of the cancellation phenomena. The goal of ongoing work is to extend and combine all of these techniques to be able to handle real applications.

References

[1] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The Satisfiability Modulo Theories Library. `www.SMT-LIB.org`, 2016.

[2] Bernard Botella, Arnaud Gotlieb, and Claude Michel. Symbolic execution of floating-point computations. *Software Testing, Verification and Reliability*, 16(2):97–121, 2006.

[3] Edmund Clarke, Daniel Kroening, and Flavio Lerda. A tool for checking ansi-c programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 168–176, 2004.

[4] Hélène Collavizza, Michel Rueher, and Pascal Van Hentenryck. Cpbpv: A constraint-programming framework for bounded program verification. *Constraints*, 15(2):238–264, 2010.

[5] Hélène Collavizza, Nguyen Le Vinh, Michel Rueher, Samuel Devulder, and Thierry Gueguen. A dynamic constraint-based BMC strategy for generating counterexamples. *SAC '11, 26th ACM Symposium On Applied Computing*, 2011.

[6] Nasrine Damouche, Matthieu Martel, Pavel Panchekha, Chen Qiu, Alexander Sanchez-Stern, and Zachary Tatlock. Toward a standard benchmark format and suite for floating-point analysis. In *Numerical Software Verification*, pages 63–77, Cham, 2017. Springer Int. Pub.

[7] Vijay D'Silva, Leopold Haller, Daniel Kroening, and Michael Tautschnig. Numeric bounds analysis with conflict-driven learning. In *TACAS*, pages 48–63, 2012.

[8] Rémy Garcia, Claude Michel, Marie Pelleau, and Michel Rueher. Towards a constraint system for round-off error analysis of floating-point computation. In *Doctoral Program – CP2018*, https://hal.archives-ouvertes.fr/hal-01956046, 2018.

[9] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. A logical product approach to zonotope intersection. In *CAV 2010, Edinburgh, UK, July 15-19,*, volume 6174 of *LNCS*, pages 212–226, 2010.

[10] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, 1991.

[11] Luiz Alberto Vieira Dias Marcus Kimura Lopes, Ricardo Bedin França and Adilson Marques da Cunha. Enhancing Range Analysis in Software Design Models by Detecting Floating-Point Absorption and Cancellation. In *Information Technology - New Generations pp 453-458*, 2018.

[12] Bruno Marre and Claude Michel. Improving the floating point addition and subtraction constraints. In *Proc. of CP 2010)*, LNCS 6308, pages 360–367, 2010.

[13] P.H. Sterbenz. *Floating-point computation*. Prentice-Hall series in automatic computation. Prentice-Hall, 1973.

[14] Laura Titolo, Marco A. Feliú, Mariano M. Moscato, and César A. Muñoz. An abstract interpretation framework for the round-off error analysis of floating-point programs. In *VMCAI 2018*, pages 516–537, 2018.

[15] Heytem Zitoun, Claude Michel, Michel Rueher, and Laurent Michel. Search strategies for floating point constraint systems. In *CP 2017*, pages 707–722, 2017.

[16] Heytem Zitoun, Claude Michel, Michel Rueher, and Laurent Michel. Sub-domain Selection Strategies For Floating Point Constraint Systems. In *Doctoral Program – CP2018*, 2018.