Weakly Relational Domains for Floating-Point Computation Analysis

Eric Goubault, Sylvie Putot

CEA Saclay, F91191 Gif-sur-Yvette Cedex, France
{eric.goubault,sylvie.putot}@cea.fr

1 Introduction

We present new numerical abstract domains for static analysis of the errors introduced by the approximation by floating-point arithmetic of real numbers computation, by abstract interpretation [2]. The analysis follows the floating-point computation, and bounds at each operation, the error committed between the floating-point and the real result. These new domains extend a former domain [3], [5], based for each variable on the abstraction of the floating-point value by an interval, and of the error by a sum of intervals decomposing the provenance of the total error along the control points (or groups of points) in the source code.

The idea of this former domain is to provide some information on the source of errors in the program. The origin of the main losses of precision is most of the time very localized, so identifying the operations responsible for these main losses, while bounding the total error, can be very useful. But it does not allow to improve the accuracy of the total error on one variable, compared to the much less costly floating-point plus total error version - it can prove even worse some times. Indeed, the correlations between variables are not used for a tighter computation values or of errors, and the method suffers from the classical drawbacks of intervals.

Resembling forms, though used in a very different way, were introduced in the interval community, under the name of affine interval arithmetic [1], precisely in order to overcome the problem of loss of correlation between variables in interval arithmetic. But in affine arithmetic, the use of a sum of coefficients is only a means for getting more precision in the dynamic computation of bounds for the real value of a variable, and these coefficients have no meaning.

The new domains we propose combine the interest of providing information on the source of errors, with the possibility to use linear correlations between variables also for improving the approximation. We propose to use these correlations on one hand for the computation of errors, which could be done for a cost very comparable to the existing approximation. And on the other hand, we propose to use a form close to affine interval arithmetic for the computation of floating-point value, except that this form must suit floating-point computations instead of real computations. This approximation is much more expensive than interval arithmetic, but comparable to the approximation used for the errors.

The ideas presented here are still preliminary. In particular, they are not fully experimented yet. We also limit our study to the case where exactly one label $i \in \mathcal{L}$ is associated to each node of the control flow graph of the program.

2 New domain for the errors

The domain introduced in [3], [5], abstracts the real value of a variable x by

$$x = f^x + \sum_{i \in \mathcal{L} \cup \{os\}} \omega_i^x \boldsymbol{\varepsilon_i} \ . \tag{1}$$

where the floating-point number f^x, and the real error terms ω_i^x, are abstracted by intervals, and $\boldsymbol{\varepsilon_i}$ is a formal variable associated to label i. Errors of order higher than one are grouped in one term associated to point os. In this domain, the error terms ω_i^x express both the rounding error committed at point i, and its propagation during further computations on variable x. Thus the fact that a same rounding error can be propagated on different variables, is lost.

The idea here is to distinguish the rounding error from its propagation. Variable x is represented by :

$$x = f^x + \sum_{i \in \mathcal{L}} t_i^x . \gamma_i \boldsymbol{\zeta_i} + \omega_{os}^x \boldsymbol{\varepsilon}_{os} \ ,$$

where $f^x \in \mathbb{F}$ is the floating-point value, abstracted either by an interval or as presented in section 3; $\gamma_i \in \mathbb{R}$ is a symbolic variable associated with the rounding error committed at point i; $t_i^x \in \mathbb{R}$ expresses the propagation of error γ_i on current variable x; $\boldsymbol{\zeta_i}$ is a formal variable associated to label i; and $\omega_{os}^x \boldsymbol{\varepsilon}_{os}$ is the higher order error term. An abstract environment thus consists of such a formal sum for each variable, together with a set of intervals giving the range of the rounding errors γ_i.

Let $\downarrow_\circ (x)$ denote the error committed by rounding x to a floating-point number, the addition for example writes:

$$z = x +^l y = f^z + \sum_{i \in \mathcal{L}} (t_i^x + t_i^y) . \gamma_i \boldsymbol{\zeta_i} + (\omega_{os}^x + \omega_{os}^y) \boldsymbol{\varepsilon}_{os} + \underbrace{\downarrow_\circ (f^x + f^y)}_{\gamma_l} \boldsymbol{\zeta_l},$$

The error due to the rounding of $f^x + f^y$ to a floating-point number can be bounded by an interval, it defines the range of γ_l. The corresponding coefficient t_l^x is initialized to 1. In further computations, the t_l^x express the linear correlations between variables on which error γ_l is propagated. They are abstracted by intervals, of width usually small, due to rounding errors in the analysis or unions.

3 New domain for the floating-point value

Linear correlations between variables can be used directly on the errors or on the real values of variables, but not on floating-point values. We thus propose to decompose the floating-point value f of a variable resulting from a set of operations, in the real value of this set of operations $r(f)$, plus the sum of errors $\delta(f)$ accumulated along the computation, $f = r(f) + \delta(f)$. Other proposals have been made to overcome this problem, most notably [6]. Our approach is different, in that [6] linearizes the expressions[1], whereas we do not need to.

- We use affine interval arithmetic [1] to compute accurately the real part $r(f)$, keeping track of linear correlations between variables:

$$r(f) = \{\alpha_0^f + \sum_{i=1}^{n} \alpha_i^f \theta_i \boldsymbol{\varphi_i}, \theta_i \in [-1,1]\} \subset [r_-(f), r_+(f)],$$

where $\alpha_i^f,\ 0 \leq i \leq n$ are real numbers, $\theta_i,\ 0 \leq i \leq n$ are symbolic variables, with value in $[-1,1]$. The assignment of a constant interval at label l, $f =^l [a,b]$, is written

$$r(f) = (a+b)/2 + (b-a)/2\, \theta_l \boldsymbol{\varphi_l}$$

to express the fact that $r(f)$ takes one value in this interval. For example, supposing $(a+b)/2$ and $(b-a)/2$ are represented exactly, $f - f$ will be found equal to 0. Indeed, the addition of two affine forms is computed componentwise:

$$r(f+g) = (\alpha_0^f + \alpha_0^g) + \sum_{i \in \mathcal{L}} (\alpha_i^f + \alpha_i^g) \theta_i \boldsymbol{\varphi_i}\,.$$

In the multiplication, the non linear part creates a new term $\alpha_l \theta_l \boldsymbol{\varphi_l}$:

$$r(f \times^l g) = \alpha_0^f \alpha_0^g + \sum_{i \in \mathcal{L},\, i \neq l} (\alpha_i^f \alpha_0^g + \alpha_i^g \alpha_0^f) \theta_i \boldsymbol{\varphi_i} + (|\alpha_l^f \alpha_0^g + \alpha_l^g \alpha_0^f| + \sum_{i \in \mathcal{L},\, j \in \mathcal{L}} |\alpha_i^f \alpha_j^g|) \theta_l \boldsymbol{\varphi_l}\,.$$

We do not discuss here the case of division, or union or intersection operators, that can not systematically be done componentwise. In these cases, using the centered form with only one symbol θ_l can be an acceptable compromise.

- For the error, we do not use the sum of errors as computed in section 2. Indeed, we do not just bound the error due to the successive roundings on the whole interval. We also want to compute the errors on the bounds, in order to get much tighter result in some cases (see example at the end of the section). We note δ_-^f and δ_+^f the errors due to the successive roundings committed on the bounds $r_-(f)$ and $r_+(f)$. The set of floating-point numbers represented by f is the interval

$$\gamma(f) = [r_-(f) + \delta_-^f, r_+(f) + \delta_+^f].$$

However, with affine interval arithmetic, the bounds of the set resulting from an arithmetic operation $f.g$ are not necessarily got from the bounds of f and g as in classical interval arithmetic. Thus, we also need the maximum error δ_M^f committed on the interval $[r_-(f), r_+(f)]$.

[1] A floating-point expression is transformed into a linear expression in the real field with interval coefficients.

We describe very briefly the principle of the propagation of errors: for an operation $h = f.g$, for example if the maximum of $r_+(h)$ is got from bounds of f and g, then the error δ^h_+ is computed using the errors on these bounds, otherwise using the maximum error on the intervals. To this must be added the new rounding error due to the conversion to a floating-point of the sum of this real result and previous errors. This error can be computed quite accurately for the bounds.

Note that an other way to use affine arithmetic for the computation of floating-point values, without the burden of an extra error term $\delta(f)$, would be to use directly the affine form $r(f)$, but with floating-point instead of real coefficients α^f_i. Instead of bounding them using higher precision numbers, we would bound them using floating-point number with outward rounding (it is not possible to use the current rounding mode for each coefficient, because the sum of rounded intermediate results is not equal to the rounded sum). The result would be similar to the formulation proposed here, but in which we would use only the maximum error on intervals. We show this on the example that follows.

Consider the computation $y = x - a * x$, for $a < 1$ such that $a/2$ is not a floating-point number, and starting with $x \in [0, 1]$. With interval arithmetic, we get $y \in [-a, 1]$. If this computation, followed by $x = y$, is in a loop, the loop will be found unstable and the fixpoint over-approximation unacceptable.

Consider now the affine interval arithmetic with floating-point coefficient, supposing the initial value of x can be associated to label 1. Here, x can be represented without over-approximation by $x = \frac{1}{2} + \frac{1}{2}\theta_1\boldsymbol{\varphi_1}$. It is not the case for ax: we note f^- the nearest floating-point value smaller than $a/2$, and f^+ the nearest floating-point value greater than $a/2$. Then $a * x = [f^-, f^+] + [f^-, f^+]\theta_1\boldsymbol{\varphi_1}$, and $y = [\frac{1}{2} - f^+, \frac{1}{2} - f^-] + [\frac{1}{2} - f^+, \frac{1}{2} - f^-]\theta_1\boldsymbol{\varphi_1}$. Then the values of y are found in $[f^- - f^+, 1 - 2f^-]$, which is better than with interval arithmetic, but the property that y is greater than 0, crucial to the study of the loop, is still lost.

Finally, we consider affine interval arithmetic with real coefficients, plus the rounding error. Variable x is again represented by $r(x) = \frac{1}{2} + \frac{1}{2}\theta_1\boldsymbol{\varphi_1}$, with no error. The real result of the multiplication is $r(ax) = \frac{a}{2} + \frac{a}{2}\theta_1\boldsymbol{\varphi_1}$, and supposing that we use higher precision numbers to abstract the real numbers, $\frac{a}{2}$ is known exactly. The error when rounding the lower bound of the result (0) is 0, and the error on the upper bound (a), is also 0. The maximum rounding error is of the order of the unit in the last place of a. The real value of y is $r(y) = \frac{1-a}{2} + \frac{1-a}{2}\theta_1\boldsymbol{\varphi_1}$. The lower bound of $r(y)$ is reached for $\theta_1 = -1$, that is from the lower bound of the set containing $r(x)$ and the upper bound of $r(ax)$, thus $\delta^y_- = 0$. We get in the same way $\delta^y_+ = 0$, the maximum error on ax is not used. And the floating-point value of y is in $[0, 1 - a]$.

4 Conclusion

We have presented in this paper preliminary ideas about two new weakly relational abstract domains, one for the safe computation of invariants describing the range of floating-point computations, and the other, on the estimates of the contributions of floating-point operations to imprecision errors with respect to an ideal real number semantics. Other variations using correlations between values and errors, for example by means of relative error, could also be used, and should be developped elsewhere.

The computations using these two domains are independent, except by the fact that the rounding error at one point is computed using the bounds for the floating-point result at this point. Nevertheless, the computation rules of the domain of errors and floating-point values have noticeable similarities. There seems to be a formal link, yet to be fully understood, between some of the propagation coefficients appearing in the two domains. These domains are probably part of a broader picture, namely that they might be linked to a more general semantics, through abstraction and concretization pairs. This should be investigated in the near future.

References

1. J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In SIBGRAPI'93, Recife, PE (Brazil), October 20-22, 1993.
2. P. Cousot and R. Cousot. Abstract interpretation frameworks. Journal of Logic and Symbolic Computation, 2(4):511–547, 1992.
3. E. Goubault. Static analyses of the precision of floating-point operations. In Static Analysis Symposium, SAS'01, number 2126 in LNCS, Springer-Verlag, 2001.
4. E. Goubault, M. Martel, and S. Putot. Asserting the precision of floating-point computations : a simple abstract interpreter. In ESOP'02, LNCS, Springer 2002.
5. M. Martel. Propagation of roundoff errors in finite precision computations : a semantics approach. In ESOP'02, number 2305 in LNCS, Springer-Verlag, 2002.
6. A. Miné. Relational Abstract Domains for the Detection of Floating-Point Run-Time Errors. In ESOP'04, number 2986 in LNCS, Springer-Verlag, 2004.
7. S. Putot, E. Goubault and M. Martel. Static Analysis-Based Validation of Floating-Point Computations. In LNCS 2991, Springer-Verlag, 2004.