Accepted for NSAD 2019 informal proceedings (https://staticanalysis.org/nsad2019/)

A Dividing Method Minimizing the Linearization Term in Affine Arithmetic

Maxime Jacquemin[1] and Franck Védrine[1]

CEA, List, Software Reliability and Security Laboratory, PC 174
91191 Gif-Sur-Yvette France
firstname.lastname@cea.fr

Abstract. Affine arithmetic is a well known tool to derive first order guaranteed approximations of general formulas over real numbers. It is also a useful tool in static analysis to keep track of linear correlations between program variables. However non-linear operations, like multiplications or divisions, introduce compensation terms to over-approximate the residue of the linearization process. Tight estimations of those terms are of utmost importance to obtain precise abstractions and thus a useful analysis. In this paper, we propose a new and simple technique to compute precise compensation terms for divisions.

1 Introduction

Affine arithmetic [1, 2], like the well-known interval arithmetic, is a powerful tool to compute guaranteed enclosures of general formulas. Each input or computed quantity x is represented by an *affine form* $x = \alpha_0^x + \alpha_1^x\epsilon_1 + \cdots + \alpha_n^x\epsilon_n$ where the α_i^x are real numbers and the ϵ_i are symbolic variables called *noise symbols* and bounded by $[-1; 1]$. This representation allows to keep track of linear correlations between the quantities of interest. Indeed, two quantities sharing a noise symbol are partially dependent. Linear operations over affine forms are easily computed as follows: $x + \lambda y = (\alpha_0^x + \lambda\alpha_0^y) + (\alpha_1^x + \lambda\alpha_1^y)\epsilon_1 + \cdots (\alpha_n^x + \lambda\alpha_n^y)\epsilon_n$.

However, non-linear operations like multiplications [4] and divisions [3] are problematic. Indeed, to obtain a result as an affine form, we have to linearize those operations. Those mandatory transformations imply losing relations with the operands and the introduction of a compensation term over-approximating the non-linear part. This new term ensures the correctness of the result but can lead to an affine form with an interval projection that is a huge over-approximation of the real bounds of the computation. In the following, we propose a new approach for computing compensation terms for divisions that can give drastically more precise projections when the operands share some relations. The algorithm operates in two steps. First, we find upper and lower bounds of the results of the division by using relations between the operands to obtain a precise estimation. Then, we compute the compensation term using those bounds to ensure a tight projection of the resulting affine form.

2 Technique presentation

2.1 Finding the bounds

We present here how to find an upper bound of the expression $x \div y$ where x and y are affine forms and $0 \notin y$. We do not present how to find a lower bound, but it can be achieve in a similar way.

We want to find an upper bound of $x \div y$, i.e we are looking for a real λ such as $x \div y \leq \lambda$ for all possible assignments of the noise symbols. This condition is equivalent to look for λ such as $x - \lambda y \leq 0$. This formulation of our condition is linear in x and y and so can be computed exactly using affine arithmetic. This means that we can define a decision procedure that, given a λ, can decide if it is an upper bound of $x \div y$. All we have to do is to compute the affine form $x - \lambda y$, then find its upper bound and compare it to zero. We recall that the tightest upper bound of this affine form is computed as $(\alpha_0^x - \lambda\alpha_0^y) + \sum_{i=1}^{n} |\alpha_i^x - \lambda\alpha_i^y|$.

Using this decision procedure, finding a tight upper bound of $x \div y$ comes down to generating a sequence of candidates converging to the optimal bound. An easy way to create this sequence is to use a dichotomy, starting with the range $[\underline{r}, \overline{r}]$ obtained with interval arithmetic. At each step, we compute the upper bound σ of $x - \lambda y$ with $\lambda = (\underline{r} + \overline{r})/2$. If σ is stricly less than zero, then λ is an upper bound of $x \div y$ but not the tightest one, so we call the dichotomy recursively with the range $[\underline{r}, \lambda]$. In a similar way, if σ is strictly greater than zero, then λ is lower than the upper bound of $x \div y$ and we call the dichotomy recursively with the range $[\lambda, \overline{r}]$. Finally if $\sigma = 0$ then λ is the tightest possible upper bound and we are done. For now, we ensure a fast termination by limiting the number of iterations. We will eventually change this and stop the dichotomy when the range $[\underline{r}, \overline{r}]$ is tight enough, with a notion of enough set by the user.

2.2 Computing the compensation term

The linearization of the division is classically computed using Taylor expansion and leads to the following classical expression where Δ is the compensation term:

$$\frac{x}{y} = \frac{1}{\alpha_0^{y\,2}} \left(\alpha_0^x \alpha_0^y + \sum_{i=1}^{n} (\alpha_i^x \alpha_0^y - \alpha_i^y \alpha_0^x)\epsilon_i + \Delta \right) \tag{1}$$

Expressions for Δ are straightforward to obtain based on equation (1). However, all those expressions have the same problem, we can not compute them without introducing an over-approximation. Thus we need to find an expression that will minimize this over-approximation. We propose to express the compensation term as follows:

$$\Delta = \left(\sum_{i=1}^{n} \alpha_i^y \epsilon_i \right) \left(\alpha_0^x - \alpha_0^y \frac{x}{y} \right) \tag{2}$$

Equation (2) is an interesting way of computing Δ because it contains the expression $x \div y$, for which a tight interval approximation can be computed using

our dichotomy technique. Thus, computing equation (2) using interval arithmetic will lead to a tight over-approximation of the compensation term. The resulting interval can then be represented using an affine form introducing a new noise symbol and added to the affine form representing $x \div y$.

2.3 Example

Let us illustrate our technique on the simple example $r = t/(t+1)$ with the input $t = 499.5 + 499.5\epsilon_1 \in [0; 999]$. Using the equation (1), the computed affine form for r is as follows:

$$r = \frac{1}{500.5^2}\left(249999.75 + 499.5\epsilon_1 + \Delta\right) \tag{3}$$

Without our dichotomy technique, the computation of the compensation term using equation (2) gives us $\Delta = 249500250\epsilon_{new}$. The projection of r using this result is $[-995, 997]$, which is a huge over-approximation. Indeed, using our dichotomy technique, we easily find that $r \in [0, 0.999]$, which are its optimal bounds. Injecting this new range in equation (2) gives us $\Delta = 249500.25\epsilon_{new}$. Finally, the projection of r using this compensation term is $[0, 2]$. Our resulting affine form is still an over-approximation, but a much tighter one.

3 Conclusion

Even if our technique seems promising, we still need to run benchmarks against the standard technique described in [3]. We are confident that the dichotomy technique will drastically improve the computation of divisions bounds. However, raising the number of authorized iterations can step the computation time up quite rapidly. A future improvement of this technique will be to implement the idea of stopping the dichotomy when the range is tight enough.

Acknowledgment

We gratefully acknowledge the help of Sylvie Putot.

References

1. Luiz Henrique de Figueiredo and Jorge Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.
2. Eric Goubault and Sylvie Putot. Perturbed affine arithmetic for invariant computation in numerical program analysis. *CoRR*, abs/0807.2961, 2008.
3. Shinya Miyajima and Masahide Kashiwagi. A dividing method utilizing the best multiplication in affine arithmetic. *IEICE Electronic Express*, 1(7):176–181, 2004.
4. Iwona Skalna and Milan Hladík. A new algorithm for chebyshev minimum-error multiplication of reduced affine forms. *Numerical Algorithms*, 76(4):1131–1152, 2017.