Under-approximations of computations in real numbers based on generalized affine arithmetic

Eric Goubault and Sylvie Putot

CEA-LIST
Laboratory for ModEling and Analysis of Systems in Interaction,
91191 Gif-sur-Yvette Cedex, France
{eric.goubault,sylvie.putot}@cea.fr

Abstract. We build a new implicitly relational abstract domain which gives accurate under-approximations of the set of real values that program variables can take. This statement is demonstrated both on a theoretical basis and on non-trivial numerical examples. It is, we believe, the first non-trivial under-approximating numerical domain in the static analysis literature.

1 Introduction

Most abstract interpretation *numerical domains* construct over-approximations of the range of program variables. This is the case for intervals [3], zones [12], polyhedra [5] etc. Of course, such static analyzes are essentially Galois connection based, formalism which precisely expresses the over-approximation process. One needs dual Galois connections, or dual concretization based frameworks, as developed in e.g. [15], to express under-approximations.

In this paper, we develop an abstract interpretation domain for under-approximating the range of real values of program variables. It is based on a variant of the affine form domain developed by the authors for over-approximations [9], and on ideas from generalized interval arithmetic [7, 8].

Such under-approximations, when combined with over-approximations, give an estimate of the quality of the result of a static analysis. But of course, our work can also be applied to statically find run-time errors that are bound to occur, from some given set of possible initial states. It can also be applied to the analysis of temporal properties of reactive systems. The latter point was studied in [6], and formalized through Galois and dual Galois connections in [14]. It is also studied in abstract model-checking, see for instance [11, 13]. We believe that these analyzes can also benefit from our approach.

Contents In section 2.1, we recall the main definitions and properties of generalized interval arithmetic, and its potential interpretations as under-approximations. We then extend these ideas to affine forms. In section 2.3, we use a generalized mean-value theorem [8] to define an under-approximating semantics of arithmetic expressions. We develop the order-theoretic apparatus needed for an actual static analysis in section 2.4, and apply this analysis in section 3.

Main contributions We describe a new numerical abstract domain that gives accurate under-approximations of the values of program variables. The time and space complexities of the primitive operations are small, as was the case with the approach for over-approximations of [9], which bear interesting relationships with the present work. Indeed, the interest of combining the two analyzes is exemplified. Using the prototype we implemented, we demonstrate very good precision of the analysis on non trivial numerical programs. On linear recursive filters of any order (pervasive in all control programs, for which safety proofs are important), we demonstrate how the analysis results for the output variable can be made as close as we want to the real range, see lemma 1. We also demonstrate in the case of linear recursive filters, how the abstract invariant discovered by our method allows us to find a sequence of inputs over time that lead to a value as close as we want to the maximal or minimal output value, allowing us to produce witnesses of potentially bad behaviors. An even more general result holds for arbitrary reactive programs, see lemma 2, and is exemplified on a perturbed filter.

2 An under-approximating domain based on generalized affine arithmetic

2.1 Generalized affine forms

We first introduce the principles of generalized interval arithmetic (following [7, 8]) and their interpretation using quantifiers. We refer the reader to these recent papers, that revisit the ideas of modal intervals, for more references on generalized intervals, modal intervals and Kaucher arithmetic.

We then extend these ideas to generalized affine forms that can be interpreted either as over or under-approximating forms for real values of variables.

Generalized interval arithmetic and notations Generalized intervals are intervals whose bounds are not ordered. The set of classical intervals is denoted by $\mathbb{IR} = \{[a, b],\ a \in \mathbb{R}, b \in \mathbb{R}, a \leq b\}$. The set of generalized intervals is denoted by $\mathbb{IK} = \{[a, b],\ a \in \mathbb{R}, b \in \mathbb{R}\}$. Intervals (classical or generalized) will be noted with bold letters.

Related to a set of real numbers $\{x \in \mathbb{R},\ a \leq x \leq b\}$, one can consider two generalized intervals, $[a, b]$, which is called proper, and $[b, a]$, which is called improper. We define the operations dual $[a, b] = [b, a]$ and pro $[a, b] = [\min(a, b), \max(a, b)]$.

The generalized intervals are partially ordered by inclusion which extends inclusion of classical intervals. Intervals (classical or generalized) will be noted with bold letters. Given two generalized intervals $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{y} = [\underline{y}, \overline{y}]$, the inclusion is defined by

$$\boldsymbol{x} \sqsubseteq \boldsymbol{y} \Leftrightarrow \underline{y} \leq \underline{x} \wedge \overline{x} \leq \overline{y}.$$

The inclusion is then related to the dual operation by $\boldsymbol{x} \sqsubseteq \boldsymbol{y} \Leftrightarrow \text{dual } \boldsymbol{x} \sqsupseteq \text{dual } \boldsymbol{y}$. Kaucher addition extends addition on classical intervals :

$$\begin{aligned}
\boldsymbol{x} + \boldsymbol{y} &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
\boldsymbol{x} - \boldsymbol{y} &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}] = \boldsymbol{x} + (-\boldsymbol{y}) \text{ where } -\boldsymbol{y} = [-\overline{y}, -\underline{y}].
\end{aligned}$$

We let $\mathcal{P} = \{\boldsymbol{x} = [\underline{x}, \overline{x}],\ \underline{x} \geq 0 \wedge \overline{x} \geq 0\}$, $-\mathcal{P} = \{\boldsymbol{x} = [\underline{x}, \overline{x}],\ \underline{x} \leq 0 \wedge \overline{x} \leq 0\}$, $\mathcal{Z} = \{\boldsymbol{x} = [\underline{x}, \overline{x}],\ \underline{x} \leq 0 \leq \overline{x}\}$, and dual $\mathcal{Z} = \{\boldsymbol{x} = [\underline{x}, \overline{x}],\ \underline{x} \geq 0 \geq \overline{x}\}$. Kaucher multiplication $\boldsymbol{x} \times \boldsymbol{y}$ is described in Table 1. Kaucher division is defined for all y such that $0 \notin \text{pro } \boldsymbol{y}$ by $\boldsymbol{x}/\boldsymbol{y} = \boldsymbol{x} \times [1/\overline{y}, 1/\underline{y}]$. When restricted to proper intervals,

$\boldsymbol{x} \times y$	$\boldsymbol{y} \in \mathcal{P}$	$\boldsymbol{y} \in \mathcal{Z}$	$\boldsymbol{y} \in -\mathcal{P}$	$y \in \text{dual}\mathcal{Z}$
$\boldsymbol{x} \in \mathcal{P}$	$[\underline{x}\underline{y}, \overline{x}\overline{y}]$	$[\overline{x}\underline{y}, \overline{x}\overline{y}]$	$[\overline{x}\underline{y}, \underline{x}\overline{y}]$	$[\underline{x}\underline{y}, \underline{x}\overline{y}]$
$\boldsymbol{x} \in \mathcal{Z}$	$[\underline{x}\overline{y}, \overline{x}\overline{y}]$	$[\min(\underline{x}\overline{y}, \overline{x}\underline{y}), \max(\underline{x}\underline{y}, \overline{x}\overline{y})]$	$[\overline{x}\underline{y}, \underline{x}\underline{y}]$	0
$\boldsymbol{x} \in -\mathcal{P}$	$[\underline{x}\overline{y}, \overline{x}\underline{y}]$	$[\underline{x}\overline{y}, \underline{x}\underline{y}]$	$[\overline{x}\overline{y}, \underline{x}\underline{y}]$	$[\overline{x}\overline{y}, \overline{x}\underline{y}]$
$\boldsymbol{x} \in \text{dual}\mathcal{Z}$	$[\underline{x}\underline{y}, \overline{x}\underline{y}]$	0	$[\overline{x}\overline{y}, \underline{x}\overline{y}]$	$[\max(\underline{x}\underline{y}, \overline{x}\overline{y}), \min(\underline{x}\overline{y}, \overline{x}\underline{y})]$

Table 1. Kaucher multiplication

these operations coincide with the classical interval operations. Kaucher arithmetic has better algebraic properties than classical interval arithmetic: Kaucher addition turns $\mathbb{IK}$ into a group, as $\boldsymbol{x} + (-\text{dual } \boldsymbol{x}) = 0$. Kaucher multiplication turns $\mathbb{IK}$ restricted to generalized intervals whose products of bounds are strictly positive into a group, as $\boldsymbol{x} \times (1/\text{dual } \boldsymbol{x}) = 1$.

Interpretation of interval computations using quantifiers Classical interval computations can be interpreted as quantified propositions. As an example, take f to be the function defined by $f(x) = x^2 - x$. Extended to interval arithmetic, its value on $x = [2, 3]$ is $f([2, 3]) = [2, 3]^2 - [2, 3] = [1, 7]$, which can be interpreted as the proposition

$$(\forall x \in [2, 3])\,(\exists z \in [1, 7])\,(f(x) = z).$$

Modal intervals extend classical intervals by coupling a quantifier to them. Extensions of modal intervals were proposed (see [7]) in the framework of generalized intervals, and called AE extensions because universal quantifiers (All) always precede existential ones (Exist) in the interpretations. They give rise to a generalized interval arithmetic which coincides with Kaucher arithmetic. Let $f : \mathbb{R}^n \to \mathbb{R}$ a function in which each variable appears only once. Let $\boldsymbol{x} \in \mathbb{IK}^n$, which we can decompose in $\boldsymbol{x}_{\mathcal{A}} \in \mathbb{IR}^p$ and $\boldsymbol{x}_{\mathcal{E}} \in (\text{dual } \mathbb{IR})^q$ with $p + q = n$. We consider the problem of computing a quantifier Q_z and an interval $\boldsymbol{z} \in \mathbb{IK}$ such that

$$(\forall x_{\mathcal{A}} \in \boldsymbol{x}_{\mathcal{A}})\,(Q_z z \in \text{pro } \boldsymbol{z})\,(\exists x_{\mathcal{E}} \in \text{pro } \boldsymbol{x}_{\mathcal{E}})(f(x) = z). \qquad (1)$$

In these expressions, if $\boldsymbol{z}$ is proper then $Q_z = \exists$, else $Q_z = \forall$. When all intervals are proper, we retrieve the interpretation of classical interval computation, which gives an over-approximation of the range of $f(x)$:

$$(\forall x \in \boldsymbol{x})\,(\exists z \in \boldsymbol{z})\,(f(x) = z).$$

And when all intervals are improper, we get an under-approximation :

$$(\forall z \in \text{pro } \boldsymbol{z})\,(\exists x \in \text{pro } \boldsymbol{x})\,(f(x) = z).$$

Affine forms for over and under-approximation An affine form ([16]) is a polynomial of degree one in a set of symbols ε_i called noise symbols :

$$\hat{x} = \alpha_0^x + \alpha_1^x \varepsilon_1 + \ldots + \alpha_n^x \varepsilon_n, \;\; \text{with } \alpha_i^x \in \mathbb{R}. \tag{2}$$

Each noise symbol ε_i is a formal variable representing an independent component of the total uncertainty on the quantity x, its value unknown but bounded in $[-1, 1]$; the corresponding coefficient α_i^x, called partial deviation, is a known real value. Coefficient α_0^x is the center of the affine form. The idea is that the same noise symbol can be shared by several quantities, expressing correlations between them.

In [9], we defined a domain for *over-approximation* of real values based on these forms with real coefficients. The concretization $\hat{\Gamma}(\hat{x})$ of $\hat{x}$ is a proper interval obtained by the evaluation of expression (2) with proper intervals $\boldsymbol{\varepsilon_i} = [-1, 1]$ and classical interval arithmetic :

$$\hat{\Gamma}(\hat{x}) = \alpha_0^x + \alpha_1^x \boldsymbol{\varepsilon_1} + \ldots + \alpha_n^x \boldsymbol{\varepsilon_n}.$$

We define here a domain for *under-approximation* based on generalized affine forms, where the $\boldsymbol{\alpha_i^x}$ coefficients are no longer real numbers but proper intervals :

$$\check{x} = \boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_1^x} \varepsilon_1 + \ldots + \boldsymbol{\alpha_n^x} \varepsilon_n, \;\; \text{with } \boldsymbol{\alpha_i^x} \in \mathbb{IR}. \tag{3}$$

We define the concretization $\check{\Gamma}(\check{x})$ of $\check{x}$ obtained by the evaluation of expression (3) with improper intervals $\boldsymbol{\varepsilon_i^*} = [1, -1]$ and Kaucher interval arithmetic :

$$\check{\Gamma}(\check{x}) = \boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_1^x} \boldsymbol{\varepsilon_1^*} + \ldots + \boldsymbol{\alpha_n^x} \boldsymbol{\varepsilon_n^*}.$$

We will construct semantics of arithmetic operations on these forms such that if $\check{\Gamma}(\check{x})$ is an improper interval, then it gives an under-approximation of the range of the real values taken by x. Otherwise, it cannot be interpreted as an under-approximation. Note that if $\boldsymbol{\alpha_0^x}$ is an interval with zero width (i.e. a real number), then $\check{\Gamma}(\check{x})$ is always an improper interval (strictly improper or with zero width). The extension $\hat{\Gamma}(\check{x})$ of $\hat{\Gamma}$ on $\check{x}$ will give an over-approximation of the values of x, but most of the time less precise than $\hat{\Gamma}(\hat{x})$.

2.2 Semantics of affine operations

The result of linear operations on (generalized) affine forms can be exactly interpreted as an affine form, without additional under or over-approximation. For two variables x and y defined by affine forms (3), and a real number r, we get:

$$\begin{aligned} x+y &= (\boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_0^y}) + (\boldsymbol{\alpha_1^x} + \boldsymbol{\alpha_1^y})\varepsilon_1 + \ldots + (\boldsymbol{\alpha_n^x} + \boldsymbol{\alpha_n^y})\varepsilon_n \\ x+r &= (\boldsymbol{\alpha_0^x} + r) + \boldsymbol{\alpha_1^x}\varepsilon_1 + \ldots + \boldsymbol{\alpha_n^x}\varepsilon_n \\ r.x &= r\boldsymbol{\alpha_0^x} + r\boldsymbol{\alpha_1^x}\varepsilon_1 + \ldots + r\boldsymbol{\alpha_n^x}\varepsilon_n \end{aligned}$$

Thus, if for example the range of x and y are known exactly, i.e. x and y are such that $\hat{\Gamma}(x) = \text{pro}\,\check{\Gamma}(x)$ and $\hat{\Gamma}(y) = \text{pro}\,\check{\Gamma}(y)$, then we also have $\hat{\Gamma}(x+y) = \text{pro}\,\check{\Gamma}(x+y)$, i.e. the range of the real result $x+y$ is known exactly (under the assumption that we compute these forms in real numbers, see 3.1 for implementation details).

2.3 Semantics of non affine arithmetic operations

We use for the under-approximation of the result of non affine arithmetic operations, an extension of the mean-value theorem to generalized intervals (see [7, 8]), which we extend to our generalized affine forms. We then derive two possible semantics for the under-approximation of the multiplication.

Mean-value theorem for generalized affine forms Suppose we have an affine model of variables $x_1, \ldots, x_k$, described as affine combinations such as (3) of noise symbols $\varepsilon_1, \ldots, \varepsilon_n$. For a differentiable function $f : \mathbb{R}^k \rightarrow \mathbb{R}$, we write $f^\varepsilon : \mathbb{R}^n \rightarrow \mathbb{R}$ the function induced by f on ε_1 to ε_n. Suppose we have an over-approximation $\boldsymbol{\Delta_i}$ of the partial derivatives

$$\left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon),\ \varepsilon \in [-1,1]^n \right\} \subseteq \boldsymbol{\Delta_i}. \tag{4}$$

Then

$$\tilde{f^\varepsilon}(\varepsilon_1, \ldots, \varepsilon_n) = f^\varepsilon(t_1, \ldots, t_n) + \sum_{i=1}^{n} \boldsymbol{\Delta_i}(\varepsilon_i - t_i), \tag{5}$$

where $(t_1, \ldots, t_n)$ is any point in $[-1,1]^n$, is interpretable in particular in the following sense :

- if $\tilde{f^\varepsilon}(\boldsymbol{\varepsilon_1^*}, \ldots, \boldsymbol{\varepsilon_n^*})$, computed with Kaucher arithmetic, is an improper interval, then $\text{pro}\,\tilde{f^\varepsilon}(\boldsymbol{\varepsilon_1^*}, \ldots, \boldsymbol{\varepsilon_n^*})$ is an under-approximation of $f^\varepsilon(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$.
- if $\tilde{f^\varepsilon}(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$ is a proper interval, then it is an over-approximation of $f^\varepsilon(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$.

Note that a tighter estimation of $\boldsymbol{\Delta_i}$ can also be used (see [7]) :

$$\left\{ \frac{\partial f^\varepsilon}{\partial \varepsilon_i}(\varepsilon_1, \ldots, \varepsilon_i, t_{i+1}, \ldots, t_n),\ (\varepsilon_1, \ldots, \varepsilon_i) \in [-1,1]^i \right\} \subseteq \boldsymbol{\Delta_i}. \tag{6}$$

Also, this theorem can be of course used when we take the ε_i in sub-ranges of $[-1,1]$, it will in fact be used in examples to improve the accuracy of the results.

Application to the multiplication We derive two affine under-approximating models for the multiplication. Model 1 is obtained using the Mean-Value Theorem on the real function f^ε defined by the multiplication of the two real variables x and y, which can be defined as real functions of the ε_i. Model 2 is obtained using it on the approximate model g^ε, in which the approximation is due to previous under-approximation of variables x and y. As both models have advantages and drawbacks, we use a combination.

1. *Model 1, using (5) on the real function, with estimation (4) for* $\boldsymbol{\Delta_i}$. We can easily prove by recurrence that, for all variables z whose real value is a linear function of noise symbols $\varepsilon_1, \ldots, \varepsilon_n$, the coefficient $\boldsymbol{\alpha_i^z}$ of the affine form obtained from our semantics is an over-approximation of $\frac{\partial z}{\partial \varepsilon_i}$. Then, for $f(x, y) = xy$, we can over-approximate
$$\frac{\partial f^\varepsilon}{\partial \varepsilon_i}(x, y) = \frac{\partial x}{\partial \varepsilon_i} y + \frac{\partial y}{\partial \varepsilon_i} x$$
by
$$\boldsymbol{\Delta_i} = \boldsymbol{\alpha_i^x y} + \boldsymbol{\alpha_i^y x}, \tag{7}$$
for any over-approximation $\boldsymbol{x}$ and $\boldsymbol{y}$ of the values taken by x and y. However, the real value $f^\varepsilon(t_1, \ldots, t_n)$ has to be computed inductively, forbidding in practice a dynamic choice of the t_i. But the advantage is that it is computable exactly for any values chosen *a priori* of the t_i. Under the assumption that we compute in real numbers, the center $\alpha_0^{\check{z}}$ of the generalized affine forms used with this model is a real coefficient and not an interval. The concretization $\check{\Gamma}(\check{z})$ is thus always interpretable as an under-approximation.
2. *Model 2, using (5) on the approximate function, with improved estimation (4) for* $\boldsymbol{\Delta_i}$. We consider affine forms $\check{x}$ and $\check{y}$ giving an under-approximation when computed with improper ε_i^*. The approximate function g^ε is given by
$$g^\varepsilon(\varepsilon) = \check{x} \times \check{y} = (\boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_1^x}\varepsilon_1 \ldots + \boldsymbol{\alpha_n^x}\varepsilon_n)(\boldsymbol{\alpha_0^y} + \boldsymbol{\alpha_1^y}\varepsilon_1 \ldots + \boldsymbol{\alpha_n^y}\varepsilon_n).$$
This allows a dynamic choice of $t_1, \ldots, t_n$, as the evaluation of $g^\varepsilon(t_1, \ldots, t_n)$ for any point $(t_1, \ldots, t_n)$ is straightforward. The affine form for the result of the multiplication $z = x \times y$ is then
$$\check{z} = (\boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_1^x}t_1 \ldots + \boldsymbol{\alpha_n^x}t_n)(\boldsymbol{\alpha_0^y} + \boldsymbol{\alpha_1^y}t_1 \ldots + \boldsymbol{\alpha_n^y}t_n) + \sum_{i=1}^{n} \boldsymbol{\Delta_i}(\varepsilon_i - t_i) \tag{8}$$
In the general case, the center $g^\varepsilon(t_1, \ldots, t_n) = (\boldsymbol{\alpha_0^x} + \boldsymbol{\alpha_1^x}t_1 \ldots + \boldsymbol{\alpha_n^x}t_n)(\boldsymbol{\alpha_0^y} + \boldsymbol{\alpha_1^y}t_1 \ldots + \boldsymbol{\alpha_n^y}t_n)$ of this form is a proper interval, which may lead to a $\check{z}$ which is not interpretable as an under-approximation.
Let us now compute the $\boldsymbol{\Delta_i}$. The partial derivatives over ε_i of this product for any given real values
$(a_0^x, a_1^x, \ldots, a_n^x, a_0^y, a_1^y, \ldots, a_n^y) \in (\boldsymbol{\alpha_0^x}, \boldsymbol{\alpha_1^x}, \ldots, \boldsymbol{\alpha_n^x}, \boldsymbol{\alpha_0^y}, \boldsymbol{\alpha_1^y}, \ldots, \boldsymbol{\alpha_n^y})$, is :
$$\frac{\partial g^\varepsilon}{\partial \varepsilon_i}(\varepsilon_1, \ldots, \varepsilon_i, t_{i+1}, \ldots, t_n) = (a_i^x a_0^y + a_i^y a_0^x) + \sum_{j=1}^{i} (a_i^x a_j^y + a_i^y a_j^x)\varepsilon_j$$

$$+\sum_{j=i+1}^{n}(a_i^x a_j^y + a_i^y a_j^x)t_j$$

We deduce bounds for the partial derivatives on the whole range of under-approximations for x and y by

$$\boldsymbol{\Delta_i} = (\boldsymbol{\alpha_i^x}\boldsymbol{\alpha_0^y}+\boldsymbol{\alpha_i^y}\boldsymbol{\alpha_0^x})+\sum_{j=1}^{i}(\boldsymbol{\alpha_i^x}\boldsymbol{\alpha_j^y}+\boldsymbol{\alpha_i^y}\boldsymbol{\alpha_j^x})\varepsilon_j+\sum_{j=i+1}^{n}(\boldsymbol{\alpha_i^x}\boldsymbol{\alpha_j^y}+\boldsymbol{\alpha_i^y}\boldsymbol{\alpha_j^x})t_j. \quad (9)$$

Practical considerations. In both models, in order to obtain an under-approximation of the multiplication, the result of $\sum_{i=1}^{n}\boldsymbol{\Delta_i}(\boldsymbol{\varepsilon_i^*}-t_i)$ must be an improper interval. Considering that the $(\boldsymbol{\varepsilon_i^*}-t_i)$ are improper intervals containing zero, and therefore are in dual $\mathcal{Z}$, we then can deduce from Table 1 what kind of intervals for $\boldsymbol{\Delta_i}$ lead to an improper interval for $\boldsymbol{\Delta_i}(\boldsymbol{\varepsilon_i^*}-t_i)$. The interval $\boldsymbol{\Delta_i}$ is proper, so it can be in $\mathcal{P}$, $-\mathcal{P}$ or $\mathcal{Z}$. If $\boldsymbol{\Delta_i}\in\mathcal{Z}$, then $\boldsymbol{\Delta_i}(\boldsymbol{\varepsilon_i^*}-t_i)$ is zero. Thus our interesting cases are $\boldsymbol{\Delta_i}\in\mathcal{P}$ or $\boldsymbol{\Delta_i}\in-\mathcal{P}$, which is satisfied when the $\boldsymbol{\Delta_i}$ intervals do not contain zero.

It is thus important to have the most accurate estimation of $\boldsymbol{\Delta_i}$ so that it does not include zero. Otherwise, a solution is to bisect one or several of the $\boldsymbol{\varepsilon_i}$ in such a way that on each bisection, our estimation for $\left\{\frac{\partial f^\varepsilon}{\partial\varepsilon_i}(\varepsilon),\ \varepsilon\in\text{pro}\ \boldsymbol{\varepsilon}\right\}$ does not contain zero. With model 2, we also have to find a trade-off between the width of $g^\varepsilon(t_1,\dots,t_n)$ and the estimation of the corresponding $\boldsymbol{\Delta_i}$.

Example. Consider $x=\frac{5}{2}+\frac{1}{2}\varepsilon_1$ and $y=\frac{9}{2}+\frac{1}{2}\varepsilon_2$. We compute an under-approximation, with model 1 for the multiplication and $(t_1,t_2)=(0,0)$, and an over-approximation, with a semantics given in ([9]), of $z=y(x^2-2y)$, respectively noted as $\check{z}$ and $\hat{z}$:

$$\check{z}=-12.375+[8,15]\varepsilon_1+[-8.125,-3.5]\varepsilon_2$$

$$\hat{z}=-12.0625+11.25\varepsilon_1-5.8125\varepsilon_2+0.5625\varepsilon_3+1.5\varepsilon_4.$$

We obtain as estimates of the range, $[-23.875,-0.875]\sqsubseteq z\sqsubseteq[-31.1875,7.0625]$. Using model 2 for the multiplication gives a slightly better under-approximation

$$\check{z}=-12.375+[9,13.5]\epsilon_1+[-8.375,-3.375]\epsilon_2,$$

which concretizes as $[-24.75,0]\sqsubseteq z$.

One bisection of ε_1 and ε_2, with each t_i taken as the center of ε_i, yields an over-approximation $z\sqsubseteq[-31.1875,5.8125]$, and as under-approximation $[-28.453125,2.765625]\sqsubseteq z$ for model 1 and $[-28.453125,2.90625]$ for model 2. Using two bisections again improve the estimation, with same result for the two models : $[-29.611328125,3.689453125]\sqsubseteq z\sqsubseteq[-30.890625,5.359375]$.

Link between under and over-approximation. We consider two variables x and y, whose values are exactly described by affine forms with real coefficients (i.e.

the under-approximation and over-approximation are equal), and we compute the multiplication $z = x \times y$.

Using model 1 with $(t_1, \ldots, t_n) = (0, \ldots, 0)$, we write

$$\check{z} = \alpha_0^x \alpha_0^y + \sum_{i=1}^{n} (\alpha_i^x \alpha_0^y + \alpha_i^y \alpha_0^x)\varepsilon_i + \left(\sum_{j=1}^{n} (\alpha_i^x \alpha_j^y + \alpha_i^y \alpha_j^x) \boldsymbol{\varepsilon_j} \right) \varepsilon_i \tag{10}$$

Computed with improper intervals for the ε_i, (10) gives an under-approximation of z. We saw that, computed with proper intervals for the ε_i, it gives an over-approximation, but better over-approximations can be obtained, as proposed in [9], as variations of

$$\hat{z} = \alpha_0^x \alpha_0^y + \sum_{i=1}^{n} (\alpha_i^x \alpha_0^y + \alpha_i^y \alpha_0^x)\varepsilon_i + (\sum_{i=1}^{n} |\alpha_i^x| . | \sum_{i=1}^{n} |\alpha_i^y|)\varepsilon_{n+1}. \tag{11}$$

where a new noise symbol ε_{n+1} is introduced to take into account the non affine part of the multiplication. We thus see that the part which is representable as an affine form of the existing symbols is shared between (10) and (11). In (10), the remaining part is expressed using the existing noise symbols ε_i to ε_n, over-approximating existing relations. Whereas in (11), a new noise symbols is created. Thus all relations between the non-linear term and the other terms are lost, resulting in an over-approximation even if the range of this non linear term could be bounded precisely.

2.4 Order-theoretic considerations

Let in what follows $\check{x}$ and $\check{y}$ be two under-approximating affine forms. Formally, we need to lift this domain of generalized affine forms so as to represent the empty set. Arithmetic operations are the lift of arithmetic operations defined in sections 2.2 and 2.3. The operations below are also trivially lifted.

Order. We define the order by $\check{x} \sqsubseteq \check{y}$ if and only if $\forall i \geq 0$, $\boldsymbol{\alpha_i^x} \sqsubseteq \boldsymbol{\alpha_i^y}$. If $\check{x} \sqsubseteq \check{y}$, then we have on the concretization $\check{\Gamma}(\check{x}) \sqsubseteq \check{\Gamma}(\check{y})$.

In the case $\check{\Gamma}(\check{x})$ and $\check{\Gamma}(\check{y})$ are improper intervals and thus can be interpreted as under-approximations of the ranges of x and y, this is equivalent to pro $\check{\Gamma}(\check{y}) \sqsubseteq$ pro $\check{\Gamma}(\check{x})$. Inclusion $\check{x} \sqsubseteq \check{y}$ thus expresses that $\check{x}$ is a better under-approximation that $\check{y}$, as it concretizes to an interval whose proper range is larger than the one of $\check{y}$.

Example. Let $\check{x} = 1 + [2, 4]\varepsilon_1$ and $\check{y} = 1 + [1, 5]\varepsilon_1$, so that $\check{x} \sqsubseteq \check{y}$. Using Kaucher arithmetic with improper $\boldsymbol{\varepsilon_1^*}$ and $\boldsymbol{\varepsilon_2^*}$, we compute $\check{\Gamma}(\check{x}) = 1 + [2, -2] = [3, -1]$ and $\check{\Gamma}(\check{y}) = 1 + [1, -1] = [2, 0]$. We indeed have $[3, -1] \sqsubseteq [2, 0]$, i.e. $[0, 2] \sqsubseteq [-1, 3]$.

This ensures even more: let $\check{C}$ be any mapping from program variables to affine forms (i.e. an abstract context), and let e be any arithmetic expression. We denote by $\check{C}[z \leftarrow \check{x}]$ the context in which we replace the mapping for variable z so as to get $\check{C}(z) = \check{x}$. We let $[\![e]\!]\check{C}$ denote the semantics of the arithmetic expression e in context $\check{C}$ as defined in section 2.3. Then $\check{x} \sqsubseteq \check{y}$ implies, for all variables z,

$$\check{\Gamma}\left([\![e]\!]\check{C}[z \leftarrow \check{x}]\right) \sqsubseteq \check{\Gamma}\left([\![e]\!]\check{C}[z \leftarrow \check{y}]\right) \tag{12}$$

(analogous to the order relation for over-approximations defined in [9]), meaning that all future evaluations e using $\check{x}$ will concretize to a bigger interval than using $\check{y}$. Hence, $\check{x}$, as an under-approximation, is more precise than $\check{y}$.

Join. The order-theoretic union is $\check{z} = \check{x} \cup \check{y}$, defined by

$$\check{z} = \check{x} \cup \check{y} = (\boldsymbol{\alpha_0^x} \cup \boldsymbol{\alpha_0^y}) + (\boldsymbol{\alpha_1^x} \cup \boldsymbol{\alpha_1^y})\varepsilon_1 + \ldots + (\boldsymbol{\alpha_n^x} \cup \boldsymbol{\alpha_n^y})\varepsilon_n.$$

A practical alternative solution, which may be used for example in loops, is to take for $\check{z}$ either $\check{x}$ or $\check{y}$.

Meet. When, for all $i \geq 0$, $\boldsymbol{\alpha_i^x} \cap \boldsymbol{\alpha_i^y} \neq \emptyset$, we can define an under-approximation of the intersection by the order-theoretic intersection

$$\check{z} = \check{x} \cap \check{y} = (\boldsymbol{\alpha_0^x} \cap \boldsymbol{\alpha_0^y}) + (\boldsymbol{\alpha_1^x} \cap \boldsymbol{\alpha_1^y})\varepsilon_1 + \ldots + (\boldsymbol{\alpha_n^x} \cap \boldsymbol{\alpha_n^y})\varepsilon_n.$$

Otherwise, we can take the bottom element or enrich the abstract domain by propagating in further computations the over-approximated[1] constraints introduced on the values of the symbolic variables ε :

$$(\boldsymbol{\alpha_0^x} - \boldsymbol{\alpha_0^y}) + (\boldsymbol{\alpha_1^x} - \boldsymbol{\alpha_1^y})\varepsilon_1 + \ldots + (\boldsymbol{\alpha_n^x} - \boldsymbol{\alpha_n^y})\varepsilon_n = 0.$$

In practice, a set of interval constraints is attached to all affine forms, and a form of (interval) Gaussian elimination can be applied for normalizing the forms.

Example. Consider the following program, with independent inputs $x \in [-1, 3]$ and $b \in [2, 4]$:

```
y = 2x + b;
if (y == x) s = 0;
else s = 1;
```

Interpreting the test `(y == x)` amounts to computing the intersection $x \cap y$. We have here the exact bounds for x and y (neither over-approximation nor under-approximation). With a computation in classical intervals, we have $y \in [0, 11]$, and we would find $s \in [0, 1]$. With affine forms, we have $x = 1 + 2\varepsilon_1$, $b = 3 + \varepsilon_2$, $y = 5 + 4\varepsilon_1 + \varepsilon_2$. For the intersection, we have to find values of ε_1 and ε_2 in $[-1, 1]$ satisfying constraint $5 + 4\varepsilon_1 + \varepsilon_2 = 1 + 2\varepsilon_1$. It simplifies to $\varepsilon_2 = -4 - 2\varepsilon_1$, with no solution. We deduce that the intersection is void, and $s = 1$.

[1] See the remark about the link with $\widetilde{pre}$ in section 2.4.

Link with under-approximating abstractions Kaucher arithmetic provides a sound under-approximating abstract interpretation in the sense of [15], as we show now. Define as usual on intervals:

$$\begin{array}{ll} \alpha^+ : \wp(\mathbb{R}) \rightarrow \mathbb{IR} & \gamma^+ : \quad \mathbb{IR} \rightarrow \wp(\mathbb{R}) \\ \qquad S \rightarrow [\inf\ S, \sup\ S] & \qquad [a,b] \rightarrow \{x \mid a \leq x \leq b\} \end{array}$$

and on improper intervals (using the notation $\wp(\mathbb{R})^{op}$ to denote the set of subsets of real numbers ordered with reverse inclusion $\subseteq^{op} = \supseteq$):

$$\begin{array}{ll} \alpha^- : \wp(\mathbb{R}) \rightarrow \text{dual}\ \ \mathbb{IR} & \gamma^- : \text{dual}\ \ \mathbb{IR} \rightarrow \wp(\mathbb{R})^{op} \\ \qquad S \rightarrow [\sup\ S, \inf\ S] & \qquad [a,b] \rightarrow \{x \mid a \geq x \geq b\} \end{array}$$

Of course, (α^+, γ^+) is the classical Galois connection for the interval abstraction. Now, whenever $\boldsymbol{i} \sqsubseteq \alpha^-(S^{op})$, we can prove that $\gamma^-(\boldsymbol{i}) \subseteq S^{op}$, hence (α^-, γ^-) is a dual Galois connection, hence under-approximating in the sense of [15]. Note that the logical interpretation of the four predicate transformers given in [15] is linked in the case of generalized interval arithmetic, to the logical interpretation of proper and improper intervals given in section 2.1: our forward under-approximating semantics for a functional f is an abstraction of $post_f(S)$ (for S a set of initial states.) By the equality $post_f(S) = \widetilde{pre}_{f^{-1}}(S)$, which in turn is best under-approximated by $\widetilde{pre}_{f^{-1\sharp}}(S)$ ($f^{-1^\sharp}$ is the best over-approximation of the inverse of f, see [15]), we explain why in section 2.4 we needed to over-approximate the constraint to solve (by $\widetilde{pre}$) to get an under-approximation of the intersection with this constraint.

Note that the least fixed point in improper intervals, of a functional F defining the abstract loop invariants, corresponds to the greatest fixed point of pro F, thus demonstrating that these under-approximating invariants are valid for all iterations of loops.

For affine forms, both over-approximating and under-approximating, we unfortunately do not have a best abstraction; hence correctness of our abstract semantics follows the generalized framework of [4]: for all variables x of the program, the under-approximating form computed by our semantics is such that $S_x \subseteq^{op} \gamma^- \circ \check{\Gamma}(\check{x})$ in $\wp(\mathbb{R})^{op}$, where S_x is the set of values that x can take. In fact, letting $[\![e]\!]_c$ stand for the concrete semantics of a term e,

$$\check{\Gamma}\left([\![e]\!]\check{C}\right) \sqsubseteq [\![e]\!]_c \gamma^- \circ \check{\Gamma}(\check{C}).$$

All further evaluations of a set of under-approximating affine forms will give an under-approximation of the real set of results. In particular, the dependencies are well encoded in the semantics. Note also that one can replace, for any variable x, $\check{x}$ by any $\check{x}'$ with $\check{x} \sqsubseteq \check{x}'$ and the same property will hold with the newly defined context thanks to relation (12).

3 Applications and experiments

3.1 Implementation

A C library implementing the different under-approximating semantics of this paper has been implemented. Of course, it does not have access to exact real arithmetic. However, computing the interval coefficients $\boldsymbol{\alpha_i^x}$ using outer rounding (i.e. rounding towards $+\infty$ for the upper bound of the interval and towards $-\infty$ for the lower bound) ensures correctness of the result. Also, using a multiple precision library such as MPFR to compute the bounds of these intervals can improve the accuracy.

3.2 Combination of under and over-approximations

We consider, for some given A, the (non-linear) iteration of the Newton algorithm $x_{i+1} = 2x_i - Ax_i^2$. If we take x_0 not too far away from the inverse of A, this iteration converges to the inverse of A. As an example, we take $A \in [1.99, 2.01]$, i.e. $\check{A} = 2 + .01\epsilon_1$, $x_0 = .5$ and ask to iterate this scheme until $|x_{i+1} - x_i| < 5e^{-6}$. To obtain sharper results, we use what we call a regular subdivision of this affine interval into 32 sub-intervals: we propagate in the analysis simultaneously 32 affine forms, starting with 32 sub-intervals $\check{A}_0, \ldots, \check{A}_{31}$ of $\check{A}$, defined for $\epsilon_1 = \left[\frac{i-16}{16}, \frac{i-15}{16}\right]$, centered at $t_i = \frac{i-\frac{31}{2}}{16}$; hence $\check{A}_i = 2 + 0.001t_i + \frac{0.01}{32}\epsilon_1$ by an obvious change of variable, with ϵ_1 still in $[-1, 1]$.

We get the concretizations of lower and upper forms shown in Figure 1. After 6 iterations, we already get stable concretizations, for both lower and over-approximations, of x_i, and they are very close to the exact range :

$$[0.497589799, 0.502433663] \sqsubseteq x_i \sqsubseteq [0.497509414, 0.502514073].$$

The difference between successive iterates $x_{i+1} - x_i$ is under-approximated by $[1.2206e^{-8}, 1.2476e^{-5}]$ at iterate $i = 0$ (respectively over-approximated by $[-9.773e^{-8}, 1.407e^{-5}]$), by $[0, 2.765e^{-10}]$ (respectively, $[-4.5409e^{-6}, 4.5416e^{-6}]$) at iterate $i = 1$, at iterate $i = 2$ by [0,0] (respectively $[-4.5415e^{-6}, 4.5415^{-6}]$) and stays stable ever after.

If we look at the exit value of the iterative scheme given by the exit criterion $|x_{i+1} - x_i| < 5e^{-6}$, we can conclude automatically, by the simultaneity of our over-approximation and under-approximation, that the iteration scheme exits after iteration $i = 1$, with the following results :

$$[0.497512498, 0.502512476] \sqsubseteq x_i \sqsubseteq [0.497509414, 0.502514073]$$

Of course, this is a general fact: the combination of under and over approximations gives in general a very powerful method to determine the invariants of a program.

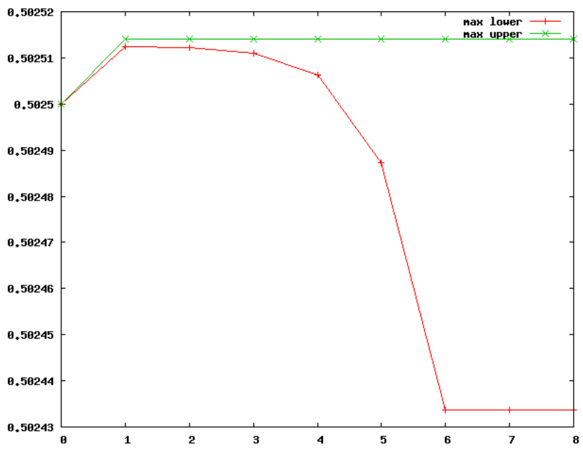

Fig. 1. Estimation of the maximum value of result x_i in the Newton algorithm.

3.3 Filters, perturbations and worst case scenario

Linear scheme. Consider the following filter of order 2:

$$S_i = 0.7E_i - 1.3E_{i-1} + 1.1E_{i-2} + 1.4S_{i-1} - 0.7S_{i-2}$$

where E_i are independent inputs between 0 and 1, so that $\check{E}_i = \hat{E}_i = \frac{1}{2} + \frac{1}{2}\epsilon_{i+1}$, and $S_0 = S_1 = 0$. We first consider the output S_i of this filter for a fixed number of unfoldings, e.g. $i = 99$. Using the over-approximating semantics of [9], our prototype gives us

$$\hat{S}_{99} = \check{S}_{99} = 0.83 + 7.81e^{-9}\varepsilon_1 - 2.1e^{-8}\varepsilon_2 - 1.58e^{-8}\varepsilon_3 + \ldots - 0.16\varepsilon_{99} + 0.35\varepsilon_{100}$$

whose concretization gives (a few more significant digits printed) as an exact enclosure of S_{99} :

$$[-1.0907188500, 2.7573854753] \sqsubseteq S_{99} \sqsubseteq [-1.0907188500, 2.7573854753].$$

Also, the affine form gives the sequence of inputs E_i that maximizes S_{99} : $E_i = 1$ if the corresponding coefficient multiplying ε_{i+1} is positive, $E_i = -1$ otherwise.

Note that the exact enclosure actually converges to

$$S_\infty = [-1.09071884989..., 2.75738551656...],$$

and therefore the signal (sequence of inputs of size 99) leading to the maximal value of S_{99} is a very good estimate of the signal leading to the maximal value of S_i, for any $i \geq 99$. This can be used to find bad-case scenarios of a program : for a specification of the filter forbidding an output greater than 2.5, this sequence of inputs provides a counter-example.

This generalizes to linear recursive filters of any order :

Lemma 1. *Consider the output s_n, $n \geq N$ of a linear recursive filter of order N*

$$s_n = \sum_{k=0}^{N-1} a_k s_{n-k-1} + \sum_{k=0}^{N} b_k e_{n-k}$$

where s_k is the output at iterate k, and e_k is the input at iterate k. Then:

(1) When unfolding k times, under and over approximating forms are equal, and their concretization gives the exact range for s_k, up to rounding errors due to the implementation of the abstract domains
(2) The under-approximating form after k unfoldings provides the sequence of inputs that lead to the maximum range of the kth output.
(3) When the filter is stable, we can make the under-approximation of the output arbitrarily close to the real range s_∞, by unfolding k times for large enough k.

Perturbation by a non linear term. We now perturb this linear scheme by adding a non-linear term $0.005E_iE_{i-1}$, obtaining

$$S_i = 0.7E_i - 1.3E_{i-1} + 1.1E_{i-2} + 1.4S_{i-1} - 0.7S_{i-2} + 0.005E_iE_{i-1}.$$

Again, we analyze S_i for a fixed number $i = 99$ of unfoldings. Using the over-approximating semantics of [9], we get

$$\begin{aligned}\hat{S}_{99} = 0.837 + 7.81e^{-9}\varepsilon_1 - 2.09e^{-8}\varepsilon_2 - 1.58e^{-8}\varepsilon_3 + \ldots - 0.157\varepsilon_{99} + 0.351\varepsilon_{100}\\ +1.77e^{-11}\varepsilon_{101} - 2.66e^{-11}\varepsilon_{102} + \ldots + 0.00175\varepsilon_{197} + 0.00125\varepsilon_{198},\end{aligned}$$

in which terms from ε_{101} to ε_{198} account for the over-approximation of non-linear computations, and do not correspond to inputs. A sequence of inputs leading to a bad-case scenario is thus not given directly by the sign of the $\varepsilon_1, \ldots, \varepsilon_{100}$ as in the linear case. Hence one cannot use the same technique as before, to be sure to reach the supremum of the range for S_{99}. One can get a plausible worst-case scenario by choosing the $E_0, \ldots, E_{99}$ that maximize the sub affine form containing only these ϵ_k, but one has no assurance that this might be even close to the real supremum of S_{99}. But we will show that the under-approximating form allows us to choose at least part of the inputs.

Using the under-approximating semantics that we have developed in this paper, and model 2 for the multiplication, we get :

$$\begin{aligned}\check{S}_{99} = 0.837 + 7.81e^{-9}\varepsilon_1 - 2.09e^{-8}\varepsilon_2 + \ldots + [-0.0577, 0.0635]\varepsilon_{93}\\ +[0.0705, 0.138]\varepsilon_{94} + [0.185, 0.223]\varepsilon_{95} + [0.25, 0.271]\varepsilon_{96} + [0.222, 0.234]\varepsilon_{97}\\ +[0.081, 0.0876]\varepsilon_{98} + [-0.158, -0.155]\varepsilon_{99} + [0.35, 0.352]\varepsilon_{100}\end{aligned}$$

This gives the following estimates for the real enclosure of S_{99}:

$$[-0.47655194955570, 2.1515519079] \sqsubseteq S_{99} \sqsubseteq [-1.10177396494, 2.77677392330].$$

Using model 1 for the multiplication, we get the slightly less precise under-approximation $[-0.435, 2.11] \sqsubseteq S_{99}$. But, when the interval coefficient $\boldsymbol{\alpha_k}$ corresponding to ε_k does not contain zero, we know what is the good choice of input E_{k-1} (see lemma 2). This cannot be proved for the form obtained with model 2 of the multiplication. However in the general case it remains a good heuristic. And in the particular case here, the interval coefficients have the same signs for the two forms. We thus know that $E_{93} = 1$ - note that E_i corresponds to ε_{i+1}! - $E_{94} = 1$, $E_{95} = 1$, $E_{96} = 1$, $E_{97} = 1$, $E_{98} = 0$ and $E_{99} = 1$ is the best depth 7 choice of inputs that will maximize S_{99}. In order to get an estimate of the supremum for S_{99}, one can try any inputs for E_0 to E_{92}. Inputs $E_0, \ldots, E_{92} = 0$ give, for instance, $S_{99} = 2.460374$. As a heuristic, one can use the $\varepsilon_0, \ldots, \varepsilon_{92}$ that maximize the over-approximating term, giving $S_{99} = 2.766383$. A one hour simulation on a 2GHz PC for 10^9 random sequences of 100 entries gives as estimate of the supremum 2.211418, trailing our estimate in both time and precision.

We can generalize again :

Lemma 2. *Suppose we have $\breve{x} = \boldsymbol{\alpha_0^x} + \sum_{i=1}^{n} \boldsymbol{\alpha_i^x} \varepsilon_i$ (with model 1 of multiplication), giving an under-approximation for $x = f(\varepsilon_1, \ldots, \varepsilon_n)$ on $[-1,1]^n$ with f continuous, and $I_+ = \{i \mid 1 \leq i \leq n,\ \alpha_i \in \mathcal{P}\}$, $I_- = \{i \mid 1 \leq i \leq n,\ \alpha_i \in -\mathcal{P}\}$, $I_z = \{i \mid 1 \leq i \leq n,\ \alpha_i \in \mathcal{Z}\}$. Then the supremum of f on $[-1,1]^n$ is reached for some set of values $\varepsilon_1, \ldots, \varepsilon_n \in [-1,1]$ with $\epsilon_i = 1$ for $i \in I_+$, $\epsilon_i = -1$ for $i \in I_-$. A similar result holds for the infimum of f.*

And as with the linear scheme, the under and over-approximations converge towards the following estimates, very close to the estimate of S_{99} :

$$[-0.4765519, 2.1515519] \sqsubseteq S_\infty \sqsubseteq [-1.10177396500, 2.77677396500]$$

Hence the previously found signal gives a scenario which is very close to the worst-case scenario. Indeed $S_{99} = 2.766383$ for which we found a scenario with our heuristics cannot be more than half a percent away from the true maximum.

4 Conclusion and related work

We have shown how to give a practical, tractable abstract semantics for under-approximating the values of program variables. Combined with an over-approximating analysis such as the one of [9], it gives a good indication of the quality of the static analysis performed. And the combination can sometimes be used to improve the analysis results, as shown in section 3.2. We can also, as side products of this analysis, give good estimations of worst-case scenarios, that lead to maximal or minimal values of some variable.

This under-approximating abstract semantics is for the time being applied to real-valued variables, and does not address yet floating-point variables. But we believe that we should be able to extend the fully relational method of [9, 10] to under-approximations of floating-point and imprecision errors. Still, we would only deliver under-approximated bounds, but not an accurate distribution of the floating-point numbers, which are discrete by nature. One can think of adding information about the minimal size of the gaps between two floating-point values in the resulting interval. We also hope to be able to have similar results on worst-case scenarios in that context, in particular for producing executions which maximize the imprecision error. This is left for future work.

Another direction we pursued was to extend the method of this paper to higher-order Taylor forms. We can indeed give a semantics based on a Taylor expansion of arbitrary degree to any program. But we do not know yet how to conclude, except in particular cases, on under-approximated bounds, contrarily to the case of over-approximations (see for instance [1] for a similar observation on over-approximations and Taylor forms).

Finally, the order-theoretic join and meet rely directly on intervals, it is hence most probable that policy iteration techniques [2] can be used on this domain.

References

1. A. Chapoutot and M. Martel. Différentiation automatique et formes de Taylor en analyse statique de programmes numériques (in French). In *AFADL'07*, 2007.
2. A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. In *CAV'05*.
3. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximations of fixed points. *Principles of Programming Languages 4*, pages 238–252, 1977.
4. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.
5. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL'78*, pages 84–97.
6. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Prog. Lang. Systems*, 19:253–291, 1997.
7. A. Goldsztejn. Modal intervals revisited. *Submitted to Reliable Computing.*
8. A. Goldsztejn, D. Daney, M. Rueher, and P. Taillibert. Modal intervals revisited: a mean-value extension to generalized intervals. In *QCP'05*.
9. E. Goubault and S. Putot. Static analysis of numerical algorithms. In *SAS'06, Seoul*, volume 4134 of *LNCS*, pages 18–34, 2006.
10. E. Goubault and S. Putot. Automatic analysis of imprecision errors in software, http://www.di.ens.fr/˜goubault/papers/abstract.pdf, 2007.
11. O. Grumberg, F. Lerda, O. Strichman, and M. Theobald. Proof-guided underapproximation-widening for multi-process systems. In *POPL*, 2005.
12. A. Miné. A new numerical abstract domain based on difference-bound matrices. In *PADO II*, volume 2053 of *LNCS*, pages 155–172, 2001.
13. C. S. Pasareanu, R. Pelánek, and W. Visser. Concrete model checking with abstract matching and refinement. In *CAV*, pages 52–66, 2005.
14. D. A. Schmidt. A calculus of logical relations for over- and underapproximating static analyses. *Sci. Comput. Program.*, 64(1):29–53, 2007.
15. D.A. Schmidt. Underapproximating predicate transformers. In K. Yi, editor, *Proc. Static Analysis Symposium*, pages 127–143. Springer LNCS 4134, 2006.
16. J. Stolfi and L.H. de Figueiredo. An introduction to affine arithmetic, TEMA 2003.