Semantics of Roundoff Error Propagation in Finite Precision Calculations

Matthieu Martel (matthieu.martel@cea.fr)
CEA - Recherche Technologique
LIST-DTSI-SOL
CEA F91191 Gif-Sur-Yvette Cedex, France

Abstract. We introduce a concrete semantics for floating-point operations which describes the propagation of roundoff errors throughout a calculation. This semantics is used to assert the correctness of a static analysis which can be straightforwardly derived from it.
In our model, every elementary operation introduces a new first order error term, which is later propagated and combined with other error terms, yielding higher order error terms. The semantics is parameterized by the maximal order of error to be examined and verifies whether higher order errors actually are negligible. We consider also coarser semantics computing the contribution, to the final error, of the errors due to some intermediate computations. As a result, we obtain a family of semantics and we show that the less precise ones are abstractions of the more precise ones.

Keywords: Numerical Precision, Abstract Interpretation, Floating-point Arithmetic, IEEE 754 Standard.

1. Introduction

It is often hard for a programmer to understand how precise the result of a floating-point calculation is and to understand which operations introduce the most significant errors [10, 19]. Recent work has shown that abstract interpretation [6, 7] is a good candidate for the validation and debugging of numerical codes [11, 23, 24] and the first attempts to verify industrial programs with a prototype analyzer, Fluctuat, are promising [13, 30].

In this article, we present the theoretical basis of the Fluctuat abstract interpreter which estimates the accuracy of a numerical result and which detects the elementary operations that introduce the most imprecision. Our approach consists in defining a precise concrete semantics for floating-point operations, based on the IEEE 754 Standard [2, 10], and, next, in practice, in using a more compact abstract semantics, among the many possible ones, to check properties of interest for a program.

Our semantics computes the errors arising during a floating-point calculation and how they are propagated by the next operations. As a result, we have the contribution to the global error of each error introduced during the execution of a program. This new approach differs from the existing ones in that it not only attempts to estimate the accuracy of a result, but also provides indications as to the source of the imprecision. It also differs from much other work in that it models the propagation of errors on initial data (sensitivity analysis) as well as the propagation of roundoff errors due to the intermediate floating-point computations (this can dominate the global error in some cases).

To our knowledge, numerical accuracy problems have almost never been treated with static analysis techniques. Most of the alternative techniques aim at dynamically estimating a better approximation of the real numbers that the program would output if the machine were working in infinite precision (see Section 1.1). In contrast, since we are interested in detecting the errors possibly introduced by floating-point numbers, we always work with the values used by the machine and we compute the errors attached to them. In addition, because they are based on static analysis, our estimations are valid for a large class of executions (and not just one as in testing techniques).

We develop a general concrete semantics $\mathcal{S}^{\mathcal{L}^*}$ for floating-point operations which explains the propagation of roundoff errors during a calculation. Elementary operations introduce new first order error terms which, once combined, yield higher order error terms. $\mathcal{S}^{\mathcal{L}^*}$ models the roundoff error propagation in finite precision computations for error terms of any order and is based on IEEE 754 Standard for floating-point numbers [2]. By modeling the propagation of errors in the general case, $\mathcal{S}^{\mathcal{L}^*}$ contributes to the general understanding of this problem and provides a theoretical basis for many static analyses. In particular, $\mathcal{S}^{\mathcal{L}^*}$ can be straightforwardly adapted to define abstract interpretations generalizing the one of [11].

Next we propose some approximations of $\mathcal{S}^{\mathcal{L}^*}$. We show that for any integer n, $\mathcal{S}^{\mathcal{L}^*}$ can be approximated by another semantics $\mathcal{S}^{\mathcal{L}^n}$ which only computes the contribution to the global error of the error terms of order at most n, as well as the residual error, i.e. the global error due to the error terms of order higher than n. Approximations are proven correct by means of Galois connections. For example, $\mathcal{S}^{\mathcal{L}^1}$ computes the contribution to the global error of the first order errors. In addition, in contrast to [11], $\mathcal{S}^{\mathcal{L}^1}$ does verify that the contribution to the global error of the error terms of order greater than one is negligible. Finally, we introduce coarser semantics which compute the contribution to the global error in the result of a computation, of the errors introduced by some pieces of code in the program. These semantics provide less information than the most general ones but use non-standard values of smaller size. In practice, this allows the user to first detect which functions of a program introduce most errors and, next, to examine in more detail which part of an imprecise function increases the error [13]. We introduce a partial order relation $\dot{\subseteq}$ on the set of the partitions of the program points and we show that, for two partitions $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$, the semantics based on $\mathcal{J}_1$ approximates the semantics based on $\mathcal{J}_2$.

As a result, we have a family of semantics parameterized by an order of error n and a partition $\mathcal{J}$. We study how to move from one semantics $\mathcal{S}^{{\mathcal{J}_1}^{n_1}}$ to another semantics $\mathcal{S}^{{\mathcal{J}_2}^{n_2}}$, during the evaluation of a program, in order to reduce the execution time. This technique, that consists of dynamically changing the partition of the control points which errors are attached to, is called dynamic partitioning. We show that, in the most general case, this dynamic partitioning problem is NP-complete.

1.1. Related work

A first approach to numerical accuracy consists of attempting to get results approximating the real solution of a problem by means of alternative arithmetics. For example, in stochastic arithmetics, each operation is executed a few times with different randomly chosen rounding modes. The mean of the results gives a good approximation of the exact solution in the reals [5, 31]. However, the correctness of the estimation is not guaranteed in all cases. Other approximations of the real numbers can be obtained by using multi-precision arithmetics [15, 29], interval arithmetics [27] or rational arithmetics [25]. Compared to the semantics developed in this article, these methods do not help to understand the origin of the errors in the final result or they neglect some error terms. In addition they are not well suited to static analysis.

A second approach to numerical accuracy consists of estimating the errors made during the execution of a program, by symbolic computations [28] or by automatic differentiation. For example, automatic differentiation consists of numerically computing the derivatives of the functions calculated by a program in order to estimate the sensitivity to the precision of the inputs [3, 14]. Again, the linear approximation made by differentiation may lead to an underestimation of the errors. In particular cases, automatic differentiation may also be used to improve the precision of a floating-point calculation with respect to the real calculation and to bound the residual error [20, 21]. The automatic differentiation approach is similar to ours in the sense that we always keep the floats computed by the machine and we attach to them information on the errors. However, automatic differentiation is difficult to use for static analysis, mainly in the case of loops. An approach to this problem is the use of static convergence criteria, like the static approximation of the Lyapunov exponents [1, 24].

Finally, other related work does not directly concern the propagation of errors throughout a numerical program even if they also contribute to the validation of the numerical accuracy of softwares. This includes formal proof techniques of numerical properties over the floats (e.g. [4, 8, 16]), constraint solvers over the floats which are used for structural test case generation [26], or mathematical studies of the stability of numerical algorithms [18].

1.2. Technical Overview

Section 2 gives an overview of the techniques developed in this article and Section 3 briefly describes some aspects of IEEE 754 Standard. In Sections 4 and 5, we introduce the semantics detailing the contribution, to the global error, of the error terms of any order and of order at most n, respectively. In Section 6, we introduce a coarser semantics which computes the contribution of the error introduced in pieces of code partitioning the program. We show that the semantics based on certain partitions are comparable. In Section 7, we show that the dynamic partitioning problem, which consists of reducing the execution time by optimizing the partition used by the semantics, is NP-complete. Section 8 concludes.

2. Motivating Example

In this section, we illustrate how the propagation of roundoff errors is modeled in our non-standard semantics using as an example a simple calculation described by the program below:

```
|1| a = 621.35;
|2| b = 1.2875;
|3| c = a * b;
```

In this example, we use a simplified set of floating-point numbers composed of a mantissa of four digits written in base 10. Thus, the program involves two values, 621.35 and 1.2875 which are not representable by floats. Our semantics assigns to `a` the non-standard value $a = 621.3\varepsilon + 0.05\varepsilon_1$ to indicate that, because of the roundoff, the float 621.3 is assigned to `a` and that a rounding error of 0.05 arises at line 1. This non-standard value is called an *error series*. ε is a formal variable attached to the float coefficient of a and ε_1 is a formal variable related to the control point 1. Similarly, the error series $b = 1.287\varepsilon + 0.0005\varepsilon_2$

is assigned to `b`. Let us focus on the product of line 3 which, carried out with error series, yields:

$$a \times b = 799.6131\varepsilon\varepsilon + 0.06435\varepsilon\varepsilon_1 + 0.31065\varepsilon\varepsilon_2 + 0.000025\varepsilon_1\varepsilon_2 \quad (1)$$

Using real and float numbers, the results of this operation are $621.35 \times 1.2875 = 799.988125$ and $621.3 \times 1.287 = 799.6131$. The difference between both solutions is 0.375025 and this error stems from the fact that the initial error on `a` (resp. `b`) was multiplied by `b` (resp. `a`) and that a *second order error*, corresponding to the multiplication of both error terms was introduced. So, at the end of the calculation, the contribution to the global error of the initial error on `a` (resp. `b`) is 0.06435 (resp. 0.31065) and corresponds to the coefficient attached to the formal variables $\varepsilon\varepsilon_1$ (resp. $\varepsilon\varepsilon_2$). The contribution of the second order error due to the initial errors on both `a` and `b` is given by the term $0.000025\varepsilon_1\varepsilon_2$.

In our calculus, the products of formal variables produce new variables by the following rewriting rule. $\varepsilon_u\varepsilon_v$ is rewritten ε_{uv} by concatenation of the indexes u and v. We assume that the index of ε is the empty word. As detailed in Section 4, the length of the word used as index of a formal variable indicates the order of the error term attached to it. In our example, we rewrite $\varepsilon_1\varepsilon_2$ as ε_{12} and $\varepsilon\varepsilon$, $\varepsilon\varepsilon_1$ and $\varepsilon\varepsilon_2$ as ε, ε_1 and ε_2, respectively. Equation (1) becomes:

$$a \times b = 799.6131\varepsilon + 0.06435\varepsilon_1 + 0.31065\varepsilon_2 + 0.000025\varepsilon_{12} \quad (2)$$

Finally, in Equation (2), the number 799.6131 has too many digits to be representable in our floating-point number system. Assuming that the elementary operations over floats are correctly rounded towards zero, we may claim that the floating-point number computed by our machine is 799.6 and that a new error term $0.0131\varepsilon_3$ is introduced by the multiplication. In the rest of this article, we assume that the floats conform to IEEE 754 Standard that actually guarantees the correct rounding of elementary operations. To sum up, we have

$$a \times b = 799.6\varepsilon + 0.06435\varepsilon_1 + 0.31065\varepsilon_2 + 0.000025\varepsilon_{12} + 0.0131\varepsilon_3 \quad (3)$$

At first sight, one could believe that the precision of this calculation mainly depends on the initial error on a since it is 100 times larger than the one on b. However, Equation (3) indicates that the final error is mainly due to the initial error on b. Hence, to improve the precision of the final result, one should first try to increase the precision on b (whenever possible). Note that, as illustrated by the above example and in contrast to other existing methods, we do not attempt to compute a better approximation of the real number that the program would output if the computer were using real numbers. Since we are interested in detecting the errors possibly introduced by floating-point calculations, we always work with the floating-point numbers used by the machine and we compute the errors attached to them.

Let us also remark that, in Equation (3), the error terms attached to ε_1, ε_2, ε_{12} and ε_3 are real numbers. In our definitions of concrete semantics (Sections 4 to 6), these errors remain real numbers. In our implementation of a static analyzer based on these semantics [13], they are abstracted by intervals of multi-precision floating-point numbers [15, 29].

3. Preliminary definitions

3.1. IEEE 754 Standard

IEEE 754 Standard was introduced in 1985 to harmonize the representation of floating-point numbers as well as the behavior of the elementary floating-point operations [2, 10]. This standard is now implemented in almost all modern processors and, consequently, it provides a precise semantics, used as a basis in this article, for the elementary operations occurring in high-level programming languages. First of all, a *floating-point number* x in base β is defined by

$$x = s \cdot (d_0.d_1 \dots d_{p-1}) \cdot \beta^e = s \cdot m \cdot \beta^{e-p+1} \tag{4}$$

where

- $s \in \{-1, 1\}$ is the sign,
- $m = d_0 d_1 \dots d_{p-1}$ is the *mantissa* with digits $0 \leq d_i < \beta$, $0 \leq i \leq p-1$,
- p is the *precision*,
- e is the exponent, $e_{min} \leq e \leq e_{max}$.

A floating-point number x is *normalized* whenever $d_0 \neq 0$. Normalization avoids multiple representations of the same number. IEEE 754 Standard specifies a few values for p, e_{min} and e_{max}. For example, single precision numbers are defined by $\beta = 2$, $p = 23$, $e_{min} = -126$ and $e_{max} = +127$; Double precision numbers are defined by $\beta = 2$, $p = 52$, $e_{min} = -1022$ and $e_{max} = +1023$. $\beta = 2$ is the only allowed basis but slight variants also are defined by IEEE 854 Standard which, for instances, allows $\beta = 2$ or $\beta = 10$.

IEEE 754 Standard, also introduces *denormalized numbers* which are floating-point numbers with $d_0 = d_1 = \ldots = d_k = 0$, $k < p - 1$ and $e = e_{min}$. Denormalized numbers make underflow gradual [10]. Finally, the following special values also are defined:

- NaN (Not a Number) resulting from an invalid operation,
- the values $\pm\infty$ corresponding to overflows,
- the values $+0$ and -0 (signed zeros).

We do not consider the extended single and extended double formats, also defined by IEEE 754 Standard, whose implementations are machine-dependent. In the rest of this paper, the notation $\mathbb{F}$ indifferently refers to the set of single or double precision numbers, since our assumptions conform to both types. $\mathbb{R}$ denotes the set of real numbers.

IEEE 754 Standard defines four rounding modes for elementary operations over floating-point numbers. These modes are towards $-\infty$, towards $+\infty$, towards zero and to the nearest. We denote them by $\circ_{-\infty}$, $\circ_{+\infty}$, $\circ_0$ and $\circ_\sim$ respectively.

Let $\mathbb{R}$ denote the set of real numbers and let $\uparrow_\circ \; : \; \mathbb{R} \to \mathbb{F}$ be the function which returns the roundoff of a real number following the rounding mode $\circ \in \{\circ_{-\infty}, \circ_{+\infty}, \circ_0, \circ_\sim\}$. IEEE 754 Standard specifies the behavior of the elementary operations $\diamond \in \{+,\ -,\ \times,\ \div\}$ between floating-point numbers by

$$f_1 \diamond_{\mathbb{F},\circ} f_2 \;=\; \uparrow_\circ(f_1 \diamond_{\mathbb{R}} f_2) \tag{5}$$

IEEE 754 Standard also specifies how the square root function must be rounded in a similar way to Equation (5) but does not specify, for theoretical reasons [22], the roundoff of other functions like sin, log, etc.

In this article, we also use the function $\downarrow_\circ : \mathbb{R} \to \mathbb{R}$ which calculates the exact error arising when a number r is approximated by $\uparrow_\circ(r)$. By definition we have

$$\downarrow_\circ(r) = r - \uparrow_\circ(r)$$

Let us remark that the elementary operations are total functions on $\mathbb{F}$, i.e. that the results of operations involving special values are specified. For instance, $1 \div +\infty = +0$, $+\infty \times +0 =$ NaN, etc. [10, 17]. However, for the sake of simplicity, we do not consider these special values, assuming that all operations have valid operands and do not return overflows or NaNs.

3.2. Standard semantics

To study the propagation of errors along a sequence of computations, we consider arithmetic expressions annotated by static labels ℓ, ℓ_1, ℓ_2, etc. and generated by the grammar of Equation (6).

$$a^\ell ::= r^\ell \mid a_0^{\ell_0} +^\ell a_1^{\ell_1} \mid a_0^{\ell_0} -^\ell a_1^{\ell_1} \mid a_0^{\ell_0} \times^\ell a_1^{\ell_1} \mid a_0^{\ell_0} \div^\ell a_1^{\ell_1} \mid F^\ell(a_0^{\ell_0}) \quad (6)$$

r denotes a value in the domain of error series and F denotes an unary function $\sqrt{\ }$, sin, etc. For any term generated by the above grammar, a unique label is attached to each sub-expression. These labels, which correspond to the nodes of the syntax tree, are used to identify the errors introduced during a calculation by an initial datum or an operation. For instance, in the expression $r_0^{\ell_0} +^\ell r_1^{\ell_1}$ the initial errors corresponding to r_0 and r_1 are attached to the formal variables ε_{ℓ_0} and ε_{ℓ_1} and the new error introduced by the addition is attached to ε_ℓ.

In the rest of this article, the set of all the labels occurring in a program is denoted $\mathcal{L}$. We use the small step operational semantics defined by the reduction rules below, where $\diamond \in \{+,\ -,\ \times,\ \div\}$. These rules correspond to a left to right evaluation strategy of the expressions.

$$\frac{a_0^{\ell_0} \to a_2^{\ell_2}}{a_0^{\ell_0} \diamond^\ell a_1^{\ell_1} \to a_2^{\ell_2} \diamond^\ell a_1^{\ell_1}} \qquad \frac{a_1^{\ell_1} \to a_2^{\ell_2}}{r_0^{\ell_0} \diamond^\ell a_1^{\ell_1} \to r_0^{\ell_0} \diamond^\ell a_2^{\ell_2}}$$

$$\frac{r = r_0 \diamond^\ell r_1}{r_0^{\ell_0} \diamond^\ell r_1^{\ell_1} \to r^\ell} \qquad \frac{a_0^{\ell_0} \to a_1^{\ell_1}}{F^\ell(a_0^{\ell_0}) \to F^\ell(a_1^{\ell_1})} \qquad \frac{r = F^\ell(r_0)}{F^\ell(r_0^{\ell_0}) \to r^\ell}$$

In the following, we introduce various domains for the values r and we specify various implementations of the operators $\diamond$. We only deal with arithmetic expressions because the semantics of the rest of the language (which is detailed in [12]) presents little interest. The only particularity concerns loops and conditionals, when the result of a comparison between floating-point numbers differs from the same comparison in $\mathbb{R}$. In this case, the semantics in $\mathbb{F}$ and $\mathbb{R}$ lead to the execution of different pieces of code. Our semantics mimics what the computer does and follows the execution path resulting from the evaluation of the test in $\mathbb{F}$. However, the errors are no longer computed (yet some improvements could consist of continuing to calculate them for local divergences of the control flow).

Labels are only attached to the arithmetic expressions because no roundoff error is introduced elsewhere. A label ℓ is related to the syntactic occurrence of an operator. If an arithmetic operation $\diamond^\ell$ is executed many times then the coefficient attached to ε_ℓ denotes the sum of the roundoff errors introduced by each instance of $\diamond$. For example, if the assignment $x = r_1^{\ell_1} \diamond^{\ell} r_2^{\ell_2}$ is carried out inside a loop then, in the error series attached to the value of x after n iterations, the coefficient attached to ε_ℓ is the sum of the roundoff errors introduced by $\diamond^\ell$ during the n first iterations.

4. General semantics

In this section, we define our most general semantics for floating-point operations, denoted $\mathcal{S}^{\mathcal{L}^*}$, and we prove its correctness. This semantics, defined in Section 4.1, details the contribution to the global error on the result of a calculation c of all the elementary errors of any order arising in c. The non-standard values in $\mathcal{S}^{\mathcal{L}^*}$ may be too large to be useful in practice and we introduce abstractions in Section 5 and Section 6. Nevertheless, the validity of the abstract semantics relies on the correctness of the concrete one, which is established in Section 4.2.

4.1. Definition of $\mathcal{S}^{\mathcal{L}^*}$

This section introduces the definition of our most general semantics $\mathcal{S}^{\mathcal{L}^*}$ which computes the errors of any order made during a calculation. Intuitively, a number r^ℓ occurring at point ℓ and corresponding to an initial datum r is represented by the error series $f\varepsilon + \omega^\ell \varepsilon_\ell$, where $f = \uparrow_\circ(r)$ is the floating-point number approximating r and $\omega^\ell = \downarrow_\circ(r)$. The functions $\uparrow_\circ$ and $\downarrow_\circ$ are defined in Section 3.1. f and ω^ℓ are written as coefficients of a formal series and ε and ε_ℓ are formal variables related to the value f known by the computer and the error between r and f.

A number r occurring at point ℓ_0 and corresponding to the result of the evaluation of an expression $a_0^{\ell_0}$ is represented by the series

$$r^{\ell_0} = f\varepsilon + \sum_{u \in \overline{\mathcal{L}^+}} \omega^u \varepsilon_u \tag{7}$$

where $\mathcal{L}$ is a set containing all the labels occurring in $a_0^{\ell_0}$ and $\overline{\mathcal{L}^+}$ is a subset of the words on the alphabet $\mathcal{L}$. $\overline{\mathcal{L}^+}$ is formally defined further in this Section. In Equation (7), f is the floating-point number approximating r and is always attached to the formal variable ε. Let ℓ be a word made up of one character. In the formal series $\sum_{u \in \overline{\mathcal{L}^+}} \omega^u \varepsilon_u$, $\omega^\ell \varepsilon_\ell$ denotes the contribution to the global error of the first order error introduced by the computation of the operation labelled ℓ during the evaluation of $a_0^{\ell_0}$. $\omega^\ell \in \mathbb{R}$ is the scalar value of this error term and ε_ℓ is a formal variable.

For a given word $u = \ell_1\ell_2\dots\ell_n$ such that $n \geq 2$, ε_u denotes the contribution to the global error of the n^{th} order error due to the combination of the errors made at points $\ell_1 \dots \ell_n$. For instance, let us consider the multiplication at point ℓ_3 of two initial data $r_1^{\ell_1} = (f_1\varepsilon + \omega_1^{\ell_1}\varepsilon_{\ell_1})$ and $r_2^{\ell_2} = (f_2\varepsilon + \omega_2^{\ell_2}\varepsilon_{\ell_2})$:

$$\begin{gathered} r_1^{\ell_1} \times^{\ell_3} r_2^{\ell_2} = \\ \uparrow_\circ(f_1f_2)\varepsilon + f_2\omega_1^{\ell_1}\varepsilon_{\ell_1} + f_1\omega_2^{\ell_2}\varepsilon_{\ell_2} + \omega_1^{\ell_1}\omega_2^{\ell_2}\varepsilon_{\ell_1\ell_2} + \downarrow_\circ(f_1f_2)\varepsilon_{\ell_3} \end{gathered} \tag{8}$$

As shown in Equation (8), the floating-point number computed by this multiplication is $\uparrow_\circ(f_1f_2)$. This is guaranteed by the IEEE 754 Standard, as discussed in Section 3.1. The initial first order errors $\omega_1^{\ell_1}\varepsilon_{\ell_1}$ and $\omega_2^{\ell_2}\varepsilon_{\ell_2}$ are multiplied by f_2 and f_1 respectively. In addition, the multiplication introduces a new first order error $\downarrow_\circ(f_1f_2)\varepsilon_{\ell_3}$, due to the roundoff of f_1f_2. Finally, this operation also introduces an error whose weight is $\omega_1^{\ell_1}\omega_2^{\ell_2}$ and which is a second order error. We attach this coefficient to the formal variable $\varepsilon_{\ell_1\ell_2}$ denoting the contribution to the global error of the second order error in ℓ_1 and ℓ_2.

Intuitively, we wish to identify the error terms $\varepsilon_{\ell_1\ell_2}$ and $\varepsilon_{\ell_2\ell_1}$ which both correspond to the second order error due to points ℓ_1 and ℓ_2. More generally, given a sequence of n letters $\ell_1, \dots, \ell_n$, we wish to identify all the error terms of order n related to words made of permutations of $\ell_1, \dots, \ell_n$. From a formal point of view, let $\mathcal{L}^*$ denote the set of words of finite length on the alphabet $\mathcal{L}$. The empty word is denoted ϵ, $|u|$ denotes the size of the word u and $u \cdot v$ denotes the concatenation of the words u and v. We introduce the equivalence relation $\sim$ which identifies the words made of the same letters: Let $u, v \in \mathcal{L}^*$, $u \sim v$ if and only if for all letter $\ell \in \mathcal{L}$, the number of occurrences of ℓ in u equals the number of occurrences of ℓ in v.

Let $\overline{\mathcal{L}^*}$ be the quotient set $\mathcal{L}^*/\!\sim$. We take as representative of an equivalence class the smallest element u of the class w.r.t. the lexicographical order. $\overline{\mathcal{L}^+}$ denotes the set $\overline{\mathcal{L}^*} \setminus \{\epsilon\}$. For any word $u \in \overline{\mathcal{L}^+}$, the formal variable ε_u is related to an n^{th} order error whenever $n = |u|$. $\varepsilon_\epsilon = \varepsilon$ is related to the floating-point number f known by the computer instead of the real value. In this article, the symbols f and ω^ϵ are used indifferently to denote the coefficient of the variable ε_ϵ.

Let $\mathcal{F}(\mathcal{D}, \overline{\mathcal{L}^*}) = \{\sum_{u \in \overline{\mathcal{L}^*}} \omega^u \varepsilon_u \;:\; \forall u,\ \omega^u \in \mathcal{D}\}$ be the set of the formal series whose formal variables are annotated by elements of $\overline{\mathcal{L}^*}$ and whose coefficients ω^u belong to $\mathcal{D}$. In this section as well as in Sections 5 and 6, we only consider concrete semantics and no informational order is required between the elements of $\mathcal{F}(\mathcal{D}, \overline{\mathcal{L}^*})$. Such an order is introduced in Section 7, for static analysis.

The semantics $\mathcal{S}^{\mathcal{L}^*}$ uses the domain $\mathbb{R}^{\mathcal{L}^*} = \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^*})$ of error series. The elementary operations on $\mathbb{R}^{\mathcal{L}^*}$ are defined in Figure 1. In this section, $\mathcal{W}$ denotes the free commutative monoid $\mathcal{W} = (\mathcal{L}^*/\sim, \epsilon, \cdot)$, whose elements belong to $\mathcal{L}^*/\sim$, with the empty word ϵ as neutral element and equipped with the associative concatenation operator $\cdot$. $\mathcal{W}^+$ denote the set $\{x \in \mathcal{W} \ : \ x \neq \epsilon\}$. In Sections 5 and 6, we will use other monoids, based on different sets of elements and on other associative operations.

$$r_1 +^\ell r_2 \stackrel{\text{def}}{=} \uparrow_\circ(f_1+f_2)\varepsilon + \sum_{u\in\mathcal{W}^+} (\omega_1^u + \omega_2^u)\varepsilon_u + \downarrow_\circ(f_1+f_2)\varepsilon_\ell \tag{9}$$

$$r_1 -^\ell r_2 \stackrel{\text{def}}{=} \uparrow_\circ(f_1-f_2)\varepsilon + \sum_{u\in\mathcal{W}^+} (\omega_1^u - \omega_2^u)\varepsilon_u + \downarrow_\circ(f_1-f_2)\varepsilon_\ell \tag{10}$$

$$r_1 \times^\ell r_2 \stackrel{\text{def}}{=} \uparrow_\circ(f_1 f_2)\varepsilon + \sum_{\substack{u \in \mathcal{W} \\ v \in \mathcal{W} \\ |u\cdot v| > 0}} \omega_1^u \omega_2^v \varepsilon_{u\cdot v} + \downarrow_\circ(f_1 f_2)\varepsilon_\ell \tag{11}$$

$$(r_1)^{-1^\ell} \stackrel{\text{def}}{=} \uparrow_\circ(f_1^{-1})\varepsilon + \frac{1}{f_1}\sum_{n\geq 1}(-1)^n \left(\sum_{u\in\mathcal{W}^+} \frac{\omega_1^u}{f_1}\varepsilon_u \right)^n + \downarrow_\circ(f_1^{-1})\varepsilon_\ell \tag{12}$$

$$\begin{aligned} r_1 \div^\ell r_2 \stackrel{\text{def}}{=} & \uparrow_\circ\left(\frac{f_1}{f_2}\right)\varepsilon + \downarrow_\circ\left(\frac{f_1}{f_2}\right)\varepsilon_\ell + \\ & \sum_{\substack{n\geq 0,\ u\in\mathcal{W} \\ d_{v_1}+\ldots+d_{v_k} = n \\ d_{v_1},\ldots,d_{v_k} \geq 0 \\ |uv_1^{d_{v_1}}\ldots v_k^{d_{v_k}}| > 0}} (-1)^n \times \frac{\omega_1^u}{f_2} \times \frac{n!}{d_{v_1}!\ldots d_{v_k}!} \times \prod_{v\in\mathcal{W}^+} \left(\frac{\omega_2^v}{f_2}\right)^{d_v} \varepsilon_{uv_1^{d_{v_1}}\ldots v_k^{d_{v_k}}} \end{aligned} \tag{13}$$

$$\begin{aligned} \sqrt{r_1}^\ell \stackrel{\text{def}}{=} & \uparrow_\circ(\sqrt{f_1})\varepsilon + \textstyle\sum_{n\geq 1}\left[\frac{\frac{1}{2}(-\frac{1}{2})\ldots(\frac{3}{2}-n)}{n!} \times \sqrt{f_1} \times \left(\sum_{u\in\mathcal{W}^+} \frac{\omega_1^u}{f_1}\varepsilon_u\right)^n \right] \\ & + \downarrow_\circ(\sqrt{f_1})\varepsilon_\ell \end{aligned} \tag{14}$$

Figure 1. Elementary operations for the semantics $\mathcal{S}^{\mathcal{L}^*}$.

In Figure 1, the formal series $\sum_{u\in\overline{\mathcal{L}^*}} \omega^u \varepsilon_u$ related to the result of an operation $\diamond^\ell$ contains the combination of the errors on the operands as well as a new error term $\downarrow_\circ(f_1 \diamond_\mathbb{R} f_2)\varepsilon_\ell$ corresponding to the error introduced by the operation $\diamond_\mathbb{F}$ occurring at point ℓ. The rules for addition and subtraction are natural. The elementary errors are added or subtracted componentwise in the formal series and the new error due to point ℓ corresponds to the roundoff of the result.

Multiplication requires more care because it introduces higher-order errors due to the multiplication of the elementary errors. Higher-order errors appear when multiplying error terms. For instance, for $\ell_1, \ell_2 \in \mathcal{L}$, and for first order errors $\omega_1^{\ell_1}\varepsilon_{\ell_1}$ and $\omega_2^{\ell_2}\varepsilon_{\ell_2}$, the operation $\omega_1^{\ell_1}\varepsilon_{\ell_1} \times \omega_2^{\ell_2}\varepsilon_{\ell_2}$ introduces a second-order error term written $\omega_1^{\ell_1}\omega_2^{\ell_2}\varepsilon_{\ell_1\ell_2}$.

The formal series resulting from the operation of r^{-1^ℓ} is obtained by means of the power series development $\frac{1}{1+x} = \sum_{n\geq 0}(-1)^n x^n$ for all x such that $-1 < x < 1$. Let $r = f\varepsilon + \sum_{u\in\mathcal{W}^+}\omega^u\varepsilon_u$ and let $e = \sum_{u\in\mathcal{W}^+}\omega^u\varepsilon_u$. Then we have

$$\frac{1}{f+e} = \frac{1}{f} \times \frac{1}{1+\frac{e}{f}} = \frac{1}{f} \times \sum_{n\geq 0}(-1)^n \frac{e^n}{f^n}$$

and we obtain Equation (12) of Figure 1. Note that, since the power series is only defined as long as it is convergent, the above technique cannot be used for any value of r. In the case of the inverse function, the convergence disc of the series $\sum_{n\geq 0}(-1)^n x^n = (1+x)^{-1}$ has radius $\rho = 1$. So, in Equation (12), we require $-1 < \sum_{u\in\mathcal{W}^+}\frac{\omega^u}{f_1} < 1$. This constraint means that Equation (12) is correct as long as the absolute value of the sum of the elementary errors is less than the related floating-point number.

The semantics of division is obtained by combining equations (11) and (12). Basically, $r_1 \div^\ell r_2$ equals $r_1 \times^\ell (r_2)^{-1^\ell}$ but for the roundoff term $\downarrow_\circ(f_1 \div f_2)\varepsilon_\ell$. Following the IEEE 754 Standard, the roundoff error introduced by the division exactly is $\downarrow_\circ(f_1 \div f_2)$ and differs from the error term introduced by $r_1 \times^\ell (r_2)^{-1^\ell}$. As a consequence, the division must be defined in a single step. In Equation (13), $d_1, \ldots, d_k$ are non-negative integers used for the multinomial coefficients introduced by the development:

$$\sum_{n\geq 0}(t_1 + \ldots + t_k)^n = \sum_{n\geq 0} \sum_{\substack{d_1 + \ldots + d_k = n \\ d_1 \geq 0, \ldots, d_k \geq 0}} \frac{n!}{d_1! \ldots d_k!} t_1^{d_1} \ldots t_k^{d_k}$$

Finally, in Equation (14), the semantics of the square root function is obtained as for inverse but other elementary functions (e.g. the trigonometric ones) are more difficult to handle, due to the fact that the IEEE 754 Standard does not specify their roundoff. In this case, system dependent assumptions must be chosen, in order to specify what the $\downarrow_\circ$ function returns.

For an expression $a_0^{\ell_0}$ such that $\mathrm{Lab}(a_0^{\ell_0}) \subseteq \mathcal{L}$, the semantics $\mathcal{S}^{\mathcal{L}^*}$ is defined by the domain $\mathbb{R}^{\mathcal{L}^*}$ for values and by the reduction rules $\rightarrow^{\mathcal{L}}$ obtained by substituting the operators on $\mathbb{R}^{\mathcal{L}^*}$ to the operators $\diamond$ in the reduction rules of Section 3.

4.2. Correctness of $\mathcal{S}^{\mathcal{L}^*}$

Concerning the correctness of the operations defined by equations (9) to (14), it is a simple matter to verify that both sides of these equations denote the same quantity, i.e. for any operator $\diamond$ and any numbers $r_1 = \sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u$, $r_2 = \sum_{u \in \mathcal{W}} \omega_2^u \varepsilon_u$ and $r = r_1 \diamond^\ell r_2 = \sum_{u \in \mathcal{W}} \omega^u \varepsilon_u$, we have

$$\sum_{u \in \mathcal{W}} \omega^u = \left(\sum_{u \in \mathcal{W}} \omega_1^u \right) \diamond \left(\sum_{u \in \mathcal{W}} \omega_2^u \right) \tag{15}$$

However, this is too weak a correctness criterion since it does not examine whether a given error term is correctly propagated during a computation. For example, Equation (16) incorrectly models the propagation of errors since it exchanges the errors attached to ε_{ℓ_1} and ε_{ℓ_2}.

$$\begin{array}{c} (f_1\varepsilon + \omega^{\ell_1}\varepsilon_{\ell_1}) +^\ell (f_2\varepsilon + \omega^{\ell_2}\varepsilon_{\ell_2}) \overset{\text{bad!}}{=} \\ \uparrow_\circ(f_1 + f_2)\varepsilon + \omega^{\ell_1}\varepsilon_{\ell_2} + \omega^{\ell_2}\varepsilon_{\ell_1} + \downarrow_\circ(f_1 + f_2)\varepsilon_\ell \end{array} \tag{16}$$

Defining addition by a generalization of Equation (16) leads to an undesirable formula satisfying the correctness criterion of Equation (15).

We aim to show that no such confusion was made in our definitions of the elementary operations, mainly for multiplication and division. So we compare the sensitivity of the terms occurring in both sides of the equations (9-14) to the values of a finite number of the coefficients ω_1^u and ω_2^u. The sensitivity of r_1 and r_2 is given by $\frac{\partial^n}{\partial \omega_{k_1}^{u_1} \dots \omega_{k_n}^{u_n}}$ for a finite subset $\omega_{k_1}^{u_1}, \dots \omega_{k_n}^{u_n}$ of the coefficients, with $k_i = 1$ or 2, $u_i \in \mathcal{W}^+$, $1 \leq i \leq n$. For example, this enables to detect that Equation (16) is not correct, since both terms do not have the same sensitivity to ω^{ℓ_1} :

$$\frac{\partial}{\partial \omega^{\ell_1}} \Big((f_1\varepsilon + \omega^{\ell_1}\varepsilon_{\ell_1}) +^\ell (f_2\varepsilon + \omega^{\ell_2}\varepsilon_{\ell_2}) \Big) = \varepsilon_{\ell_1} \tag{17}$$

$$\frac{\partial}{\partial \omega^{\ell_1}} \Big(\uparrow_\circ(f_1 + f_2)\varepsilon + \omega^{\ell_1}\varepsilon_{\ell_2} + \omega^{\ell_2}\varepsilon_{\ell_1} + \downarrow_\circ(f_1 + f_2)\varepsilon_\ell \Big) = \varepsilon_{\ell_2} \tag{18}$$

We first introduce Lemma 1 which deals with first order partial derivatives.

LEMMA 1. *Let* $\diamond \in \{+, -, \times, \div\}$ *be an elementary operation and let* $\diamond^\ell \in \{+^\ell, -^\ell, \times^\ell, \div^\ell\}$ *be the corresponding non-standard operation defined in Equations (9) to (14). For any* $r_1 = \sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u$, $r_2 = \sum_{u \in \mathcal{W}} \omega_2^u \varepsilon_u$ *and for any* $u_0 \in \mathcal{W}^+ \setminus \{\ell\}$ *we have*

$$\frac{\partial(r_1 \diamond r_2)}{\partial \omega_1^{u_0}} = \frac{\partial(r_1 \diamond^\ell r_2)}{\partial \omega_1^{u_0}} \quad \textit{and} \quad \frac{\partial(r_1 \diamond r_2)}{\partial \omega_2^{u_0}} = \frac{\partial(r_1 \diamond^\ell r_2)}{\partial \omega_2^{u_0}} \tag{19}$$

Proof. (Multiplication)
On one hand, we have:

$$\frac{\partial(r_1 \times^\ell r_2)}{\partial \omega_1^{u_0}} = \frac{\partial}{\partial \omega_1^{u_0}} \left(\sum_{u,v \in \mathcal{W}} \omega_1^u \omega_2^v \varepsilon_{uv} \right)$$

Using the equality

$$\sum_{u,v \in \mathcal{W}} \omega_1^u \omega_2^v \varepsilon_{uv} = \sum_{v \in \mathcal{W}} \omega_1^{u_0} \omega_2^v \varepsilon_{u_0 v} + \sum_{u \in \mathcal{W} \setminus \{u_0\}, v \in \mathcal{W}} \omega_1^u \omega_2^v \varepsilon_{uv}$$

we obtain

$$\frac{\partial(r_1 \times^\ell r_2)}{\partial \omega_1^{u_0}} = \sum_{v \in \mathcal{W}} \omega_2^v \varepsilon_{u_0 v}$$

On the other hand,

$$\frac{\partial}{\partial \omega_1^{u_0}} \left(\sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u \times \sum_{v \in \mathcal{W}} \omega_2^u \varepsilon_u \right) =$$

$$\sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u \times \frac{\partial}{\partial \omega_1^{u_0}} \left(\sum_{v \in \mathcal{W}} \omega_2^v \varepsilon_v \right) + \sum_{v \in \mathcal{W}} \omega_2^v \varepsilon_v \times \frac{\partial}{\partial \omega_1^{u_0}} \left(\sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u \right) =$$

$$0 + \left(\sum_{v \in \mathcal{W}} \omega_2^v \varepsilon_v \right) \times \varepsilon_{u_0} = \sum_{v \in \mathcal{W}} \omega_2^v \varepsilon_{u_0 v}$$

□

Lemma 1 ensures that the sensitivity to a single error term in the input series is correctly handled in our model. Proposition 1 below generalizes Lemma 1 to the sensitivity to a finite number of coefficients.

PROPOSITION 1. *Let* $\diamond \in \{+, -, \times, \div\}$ *denote one of the usual operators on formal series and let* $\diamond^\ell \in \{+^\ell, -^\ell, \times^\ell, \div^\ell\}$ *denote the operators defined in Equations (9) to (14). For any* $r_1 = \sum_{u \in \mathcal{W}} \omega_1^u \varepsilon_u$, $r_2 = \sum_{u \in \mathcal{W}} \omega_2^u \varepsilon_u$ *and for any* $\omega_{k_1}^{u_1}, \ldots \omega_{k_n}^{u_n}$, $k_i = 1$ *or* 2, $u_i \in \mathcal{W}^+ \setminus \{\ell\}$, $1 \le i \le n$, *we have:*

$$\frac{\partial^n (r_1 \diamond r_2)}{\partial \omega_{k_1}^{u_1} \ldots \omega_{k_n}^{u_n}} = \frac{\partial^n (r_1 \diamond^\ell r_2)}{\partial \omega_{k_1}^{u_1} \ldots \omega_{k_n}^{u_n}} \tag{20}$$

The proof is by recurrence, using Lemma 1. As a conclusion, let us remark that, from Equations (17) and (18), the incorrect definition of addition given in Equation (16) satisfies neither Lemma 1, nor Proposition 1.

5. Restriction to errors of the n^{th} order

The semantics $\mathcal{S}^{\mathcal{L}^*}$, introduced in Section 4, computes the errors of any order arising during a calculation. This is a general model for error propagation but it is commonly assumed that, in practice, errors of order greater than one or (rarely) two are negligible. However, even if, from a practical point of view, we are only interested in detailing the contribution of the first n order errors to the global error, for $n = 1$ or $n = 2$, a safe semantics must check that higher order errors actually are negligible.

We introduce a family $(\mathcal{S}^{\mathcal{L}^n})_{n\in\mathbb{N}}$ of semantics such that the semantics $\mathcal{S}^{\mathcal{L}^n}$ details the contribution to the global error of the errors of order at most n. In addition, $\mathcal{S}^{\mathcal{L}^n}$ collapses into the coefficient of a single formal variable of the series the whole contribution of the errors of order higher than n. A value r is represented by

$$r = f\varepsilon + \sum_{u\in\overline{\mathcal{L}^+},\ |u|\le n} \omega^u\varepsilon_u + \omega^\varsigma\varepsilon_\varsigma \tag{21}$$

The term $\omega^\varsigma\varepsilon_\varsigma$ of the series aggregates the elementary errors of order higher than n. Starting with $n = 1$, one can examine the contribution of the first order errors to the global error in a computation. If the ω^ς coefficient is negligible in the result, then the semantics $\mathcal{S}^{\mathcal{L}^1}$ provides enough information to understand the nature of the error. Otherwise, $\mathcal{S}^{\mathcal{L}^1}$ states that there is a higher order error which is not negligible but does not indicate which are the main operations which make this error grow. This information can be obtained by the semantics $\mathcal{S}^{\mathcal{L}^n}$ for an adequate value of n.

Let $\mathcal{L}^n$ be the set of words of length at most n on the alphabet $\mathcal{L}$ and let $\overline{\mathcal{L}^n} = (\mathcal{L}^n/\!\sim) \cup \{\varsigma\}$. $\varsigma \notin \mathcal{L}^*$ is a special word representing all the words of size greater than n. We define the new concatenation operator

$$u \cdot_n v = \begin{cases} u\cdot v \text{ if } |u\cdot v| \le n \text{ and } u, v \ne \varsigma \\ \varsigma \text{ otherwise} \end{cases} \tag{22}$$

For the sake of simplicity, we write $u \cdot v$ instead of $u \cdot_n v$ whenever it is clear that u and v belong to $\overline{\mathcal{L}^n}$. The domain of error series of order at most n is $\mathbb{R}^{\mathcal{L}^n} = \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})$. The elementary operations on $\mathbb{R}^{\mathcal{L}^n}$

are defined by the equations (9) to (14) of Section 4 in which $\mathcal{W}$ now denotes the new monoid $\mathcal{W} = (\overline{\mathcal{L}^n}, \epsilon, \cdot_n)$.

Let $\mathcal{S}^{\mathcal{L}^n}$ denote the semantics defined by the domain $\mathbb{R}^{\mathcal{L}^n}$ for values and by the reduction rules of Section 2. The semantics $\mathcal{S}^{\mathcal{L}^n}$ indicates the contribution to the global error of the elementary errors of order at most n.

We now focus on the correctness of $(\mathcal{S}^{\mathcal{L}^n})_{n\in\mathbb{N}}$. Recall that the correctness of the most general semantics $\mathcal{S}^{\mathcal{L}^*}$ stems from Lemma 1 and Proposition 1. For any integer n, $\mathcal{S}^{\mathcal{L}^n}$ is another concrete semantics, more useful in practice than $\mathcal{S}^{\mathcal{L}^*}$, but a correctness proof like that for $\mathcal{S}^{\mathcal{L}^*}$ would be too difficult. So we aim at showing that $\mathcal{S}^{\mathcal{L}^n}$ approximates $\mathcal{S}^{\mathcal{L}^*}$ and that, for $m \leq n$, $\mathcal{S}^{\mathcal{L}^m}$ is a conservative approximation of $\mathcal{S}^{\mathcal{L}^n}$. So, the semantics of order m can always be considered as being a safe approximation of the semantics of order n, for any $n > m$.

Let $m \leq n$. Because all our semantics are concrete, we compare the collecting semantics, i.e. we relate a set of executions in $\mathcal{S}^{\mathcal{L}^n}$ to a set of executions in $\mathcal{S}^{\mathcal{L}^m}$ by means of the Galois connections:

$$\langle \wp(\mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})), \subseteq\rangle \quad \underset{\alpha^{n,m}}{\overset{\gamma^{m,n}}{\leftrightarrows}} \quad \langle \wp(\mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^m})), \subseteq\rangle \tag{23}$$

$\alpha^{n,m}$ and $\gamma^{m,n}$ are first defined for single values and, next, for sets of values.

$$\alpha^{n,m}\left(\sum_{u\in\overline{\mathcal{L}^n}} \omega^u \varepsilon_u\right) \stackrel{\text{def}}{=}$$

$$\sum_{u\in\overline{\mathcal{L}^m}\setminus\{\varsigma\}} \omega^u \varepsilon_u + \left(\sum_{u\in(\overline{\mathcal{L}^n}\setminus\overline{\mathcal{L}^m})\cup\{\varsigma\}} \omega^u\right)\varepsilon_\varsigma$$

and, for some set X, $\alpha^{n,m}(X) = \{\alpha^{n,m}(x) \ : \ x \in X\}$. The abstraction of the coefficients attached to ε_u, for any $u \in \overline{\mathcal{L}^m} \setminus \{\varsigma\}$, is natural. $\alpha^{n,m}$ also adds the coefficients of the terms of order higher than m and appends the result to ε_ς. Next, the concretization becomes:

$$\gamma^{m,n}\left(\sum_{u\in\overline{\mathcal{L}^m}} \nu^u \varepsilon_u\right) \stackrel{\text{def}}{=} \left\{ \sum_{u\in\overline{\mathcal{L}^n}} \omega^u \varepsilon_u \ : \ \left| \begin{array}{l} \omega^u = \nu^u \text{ if } u \in \overline{\mathcal{L}^m} \setminus \{\varsigma\} \\ \sum_{u\in(\overline{\mathcal{L}^n}\setminus\overline{\mathcal{L}^m})\cup\{\varsigma\}} \omega^u = \nu^\varsigma \end{array} \right. \right\}$$

and $\gamma^{m,n}(X) = \cup_{x\in X}\gamma^{m,n}(x)$. $\gamma^{m,n}$ maps a series $\sum_{u\in\overline{\mathcal{L}^m}} \nu^u\varepsilon_u \in \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^m})$ to the set of series of the form $\sum_{u\in\overline{\mathcal{L}^n}} \omega^u\varepsilon_u \in \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})$ such that $\omega^u = \nu^u$ for any $u \in \overline{\mathcal{L}^m} \setminus \{\varsigma\}$ and such that ν^ς equals the sum of the remaining terms.

The correctness of the elementary operations in $\mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^m})$, w.r.t. the correctness of the same operations in $\mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})$ stems from Lemma 2.

LEMMA 2. *Let ℓ be a control point, let $r^{\mathcal{L}^n}, s^{\mathcal{L}^n} \in \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})$ be error series and let $r^{\mathcal{L}^m} = \alpha^{n,m}(r^{\mathcal{L}^n})$, $s^{\mathcal{L}^m} = \alpha^{n,m}(s^{\mathcal{L}^n})$, $1 \le m \le n$. For any operator $\diamond \in \{+, -, \times, \div\}$ we have*

$$r^{\mathcal{L}^n} \diamond^\ell s^{\mathcal{L}^n} \in \gamma^{m,n}(r^{\mathcal{L}^m} \diamond^\ell s^{\mathcal{L}^m})$$

Proof. Proof (Multiplication)
Let

$$r^{\mathcal{L}^n} = \sum_{u \in \overline{\mathcal{L}^n}} \omega_r^u \varepsilon_u$$

and

$$s^{\mathcal{L}^n} = \sum_{u \in \overline{\mathcal{L}^n}} \omega_s^u \varepsilon_u$$

First, by Equation (11), we have:

$$\begin{aligned} t^{\mathcal{L}^n} &= r^{\mathcal{L}^n} \times^\ell s^{\mathcal{L}^n} \\ &= \uparrow_\circ(\omega_r^\epsilon \omega_s^\epsilon)\varepsilon_\epsilon + \sum_{\substack{u,v \in \overline{\mathcal{L}^n} \\ |u.v| > 0}} \omega_r^u \omega_s^v \varepsilon_{u.v} + \downarrow_\circ(\omega_r^\epsilon \omega_s^\epsilon)\varepsilon_\ell \end{aligned} \tag{24}$$

Similarly, if $r^{\mathcal{L}^m} = \sum_{u \in \overline{\mathcal{L}^m}} \nu_r^u \varepsilon_u$ and $s^{\mathcal{L}^m} = \sum_{u \in \overline{\mathcal{L}^m}} \nu_s^u \varepsilon_u$ then

$$\begin{aligned} t^{\mathcal{L}^m} &= r^{\mathcal{L}^m} \times^\ell s^{\mathcal{L}^m} \\ &= \uparrow_\circ(\nu_r^\epsilon \nu_s^\epsilon)\varepsilon_\epsilon + \sum_{\substack{u,v \in \overline{\mathcal{L}^m} \\ |u.v| > 0}} \nu_r^u \nu_s^v \varepsilon_{u.v} + \downarrow_\circ(\nu_r^\epsilon \nu_s^\epsilon)\varepsilon_\ell \end{aligned} \tag{25}$$

By definition of $\gamma^{m,n}$,

$$\gamma^{m,n}(t^{\mathcal{L}^m}) =$$

$$\left\{ \sum_{u \in \overline{\mathcal{L}^n}} \omega^u \varepsilon_u \ : \ \left| \begin{array}{l} \omega^u = \nu_t^u \text{ if } u \in \overline{\mathcal{L}^m} \setminus \{\varsigma\} \\ \sum_{u \in (\overline{\mathcal{L}^n} \setminus \overline{\mathcal{L}^m}) \cup \{\varsigma\}} \omega^u = \nu_t^\varsigma \end{array} \right. \right\} \tag{26}$$

where, in (26),

$$\nu_t^u = \sum_{u_1 u_2 = u} \nu_r^{u_1} \nu_s^{u_2}$$

and

$$\nu_t^\varsigma = \sum_{u_1 u_2 = u \in (\overline{\mathcal{L}^n} \setminus \overline{\mathcal{L}^m}) \cup \{\varsigma\}} \nu_r^{u_1} \nu_s^{u_2}$$

We have to show that $\sum_{u \in (\overline{\mathcal{L}^n} \setminus \overline{\mathcal{L}^m}) \cup \{\varsigma\}} \omega_t^u = \nu_t^\varsigma$. We use the notations $M = \overline{\mathcal{L}^m} \setminus \{\varsigma\}$ and $N = (\overline{\mathcal{L}^n} \setminus \overline{\mathcal{L}^m}) \cup \{\varsigma\}$. We have:

$$\sum_{u \in N} \omega_t^u \ = \sum_{\substack{u \in \overline{\mathcal{L}^n}, v \in \overline{\mathcal{L}^n} \\ u.v \in N}} \omega_r^u \omega_s^v$$

$$= \sum_{\substack{u,v \in M \\ u.v \in N}} \omega_r^u \omega_s^v + \sum_{u,v \in N} \omega_r^u \omega_s^v$$
$$+ \sum_{\substack{u \in M, v \in N \\ u.v \in N}} \omega_r^u \omega_s^v + \sum_{\substack{u \in N, v \in M \\ u.v \in N}} \omega_r^u \omega_s^v$$

Since $r^{\mathcal{L}^m} = \alpha^{n,m}(r^{\mathcal{L}^n})$ and $s^{\mathcal{L}^m} = \alpha^{n,m}(s^{\mathcal{L}^n})$, $\sum_{u \in N} \omega_r^u = \nu_r^\varsigma$ and $\sum_{u \in N} \omega_s^u = \nu_s^\varsigma$. So,

$$\sum_{u \in N} \omega_t^u = \sum_{\substack{u,v \in M \\ u.v \in N}} \omega_r^u \omega_s^v + \nu_r^\varsigma \nu_s^\varsigma + \sum_{u \in M} \omega_r^u \nu_s^\varsigma + \sum_{v \in M} \nu_r^\varsigma \omega_s^v$$

Since $r^{\mathcal{L}^m} = \alpha^{n,m}(r^{\mathcal{L}^n})$ and $s^{\mathcal{L}^m} = \alpha^{n,m}(s^{\mathcal{L}^n})$, for any word u such that $|u| \leq m$, we have $\omega_r^u = \nu_r^u$ and $\omega_s^u = \nu_s^u$, so

$$\sum_{u \in N} \omega_t^u = \sum_{\substack{u,v \in M \\ u.v \in N}} \nu_r^u \nu_s^v + \nu_r^\varsigma \nu_s^\varsigma + \sum_{u \in M} \nu_r^u \nu_s^\varsigma + \sum_{v \in M} \nu_r^\varsigma \nu_s^v$$
$$= \sum_{\substack{u,v \in \overline{\mathcal{L}^m} \\ u.v \in N}} \nu_r^u \nu_s^v = \nu_t^\varsigma$$

□

Let us remark that Lemma 2 and its proof include the case $n = \infty$ (where $\mathcal{S}^{\mathcal{L}^*}$ is written instead of $\mathcal{S}^{\mathcal{L}^\infty}$). To extend Lemma 2 to sequences of reduction steps, we introduce the mapping $\mathcal{R}$ defined by Equation (27). Lab$(a_0^{\ell_0})$ is the set of labels occurring in $a_0^{\ell_0}$.

$$\mathcal{R}(a_0^{\ell_0}) \; : \; \left| \begin{array}{lcl} \mathrm{Lab}(a_0^{\ell_0}) & \to & \wp\left(\mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})\right) \\ \ell & \mapsto & \begin{cases} \{r\} \text{ if } a^\ell = r^\ell \\ \emptyset \text{ otherwise} \end{cases} \end{array} \right. \tag{27}$$

$\mathcal{R}(a_0^{\ell_0})(\ell)$ returns a singleton made of a value if the sub-expression a^ℓ in $a_0^{\ell_0}$ is a value and $\mathcal{R}(a_0^{\ell_0})(\ell)$ returns the empty set otherwise.

PROPOSITION 2. *Let $a_0^{\ell_0 \mathcal{L}^m}$ and $a_0^{\ell_0 \mathcal{L}^n}$ be syntactically equivalent expressions such that for any $\ell \in \mathcal{L}$, we have:*

$$\mathcal{R}(a_0^{\ell_0 \mathcal{L}^n})(\ell) \subseteq \gamma^{m,n}(\mathcal{R}(a_0^{\ell_0 \mathcal{L}^m})(\ell))$$

If $a_0^{\ell_0 \mathcal{L}^m} \to a_1^{\ell_1 \mathcal{L}^m}$ *then* $a_0^{\ell_0 \mathcal{L}^n} \to a_1^{\ell_1 \mathcal{L}^n}$ *such that* $a_1^{\ell_1 \mathcal{L}^m}$ *and* $a_1^{\ell_1 \mathcal{L}^n}$ *are syntactically equivalent expressions and for all* $\ell \in \mathcal{L}$, $\mathcal{R}(a_1^{\ell_1 \mathcal{L}^n})(\ell) \subseteq \gamma^{m,n}(\mathcal{R}(a_1^{\ell_1 \mathcal{L}^m})(\ell))$.

Given an arithmetic expression $a_0^{\ell_0}$, Proposition 2 shows how to link $\mathcal{S}^{\mathcal{L}^n}(a_0^{\ell_0})$ to $\mathcal{S}^{\mathcal{L}^{n+1}}(a_0^{\ell_0})$ for any integer $n \geq 0$. The proof is a simple induction on the structure of the expression a_0.

The semantics $\mathcal{S}^{\mathcal{L}^n}$ is based on the domain $\mathbb{R}^{\mathcal{L}^n} = \mathcal{F}(\mathbb{R}, \overline{\mathcal{L}^n})$. The semantics $\mathcal{S}^{\mathcal{L}^*}$ can be viewed as the infinite limit of this model, since the operations on $\mathbb{R}^{\mathcal{L}^*}$, as defined in Section 4, correspond to the ones of equations (9) to (14) with $\mathcal{W} = (\mathcal{L}^*, \epsilon, \cdot)$. Conversely, the semantics $\mathcal{S}^{\mathcal{L}^0}$ uses error series of the form $\omega^\epsilon \varepsilon_\epsilon + \omega^\varsigma \varepsilon_\varsigma$ and computes the global error made during a calculation. In short, there is a chain of Galois connections between the semantics of any order:

$$\mathcal{S}^{\mathcal{L}^*}(a_0^{\ell_0}) \leftrightarrows \ldots \mathcal{S}^{\mathcal{L}^n}(a_0^{\ell_0}) \leftrightarrows \mathcal{S}^{\mathcal{L}^{n-1}}(a_0^{\ell_0}) \ldots \leftrightarrows \mathcal{S}^{\mathcal{L}^0}(a_0^{\ell_0})$$

$\mathcal{S}^{\mathcal{L}^*}(a_0^{\ell_0})$ is the most informative result since it indicates the contribution of the elementary errors of any order. $\mathcal{S}^{\mathcal{L}^0}(a_0^{\ell_0})$ is the least informative result which only indicates the global error made during the computation.

6. Coarse grain errors

In this section, we introduce a new semantics that generalizes the ones of Section 4 and Section 5. Intuitively, we no longer consider elementary errors corresponding to the errors due to individual operations and we instead compute errors due to pieces of code partitioning the program. For instance, one may be interested in the contribution to the global error of the whole error due to an intermediate formula or due to a given line in the program code.

In practice, these new semantics are important to reduce the memory size used to store the values. From a theoretical point of view, it is necessary to prove that they are correct with respect to the general semantics $\mathcal{S}^{\mathcal{L}^*}$ of Section 4.

We show that the different partitions of the control points can be partially ordered in such a way that we can compare the semantics based on comparable partitions. Let $\mathcal{J} = \{J_1, J_2, \ldots, J_p\} \in \mathcal{P}(\mathcal{L})$ be a partition of the control points. We now consider the words on the alphabet $\mathcal{J}$. $\mathcal{J}^n$ denotes the words of maximal length n and $\overline{\mathcal{J}^n} = (\mathcal{J}^n/\sim) \cup \{\varsigma\}$. The concatenation operator $\cdot_n$ related to $\overline{\mathcal{J}^n}$ is defined in Equation (22).

For a maximal order of error $n \in \mathbb{N}$, we consider the family of domains $(\mathcal{F}(\mathbb{R}, \overline{\mathcal{J}^n}))_{\mathcal{J} \in \mathcal{P}(\mathcal{L})}$, equivalently denoted $(\mathbb{R}^{\mathcal{J}^n})_{\mathcal{J} \in \mathcal{P}(\mathcal{L})}$. Basically, a unique label identifies all the operations of the same partition. A value $r^{\mathcal{J}^n} \in \mathbb{R}^{\mathcal{J}^n}$ is written

$$r^{\mathcal{J}^n} = f\varepsilon + \sum_{\substack{u \in \overline{\mathcal{J}^{n+}} \\ u = J_1 \dots J_k}} \Bigg(\sum_{\substack{v = \ell_1 \dots \ell_k \in \overline{\mathcal{L}^n} \\ \forall i,\ 1 \le i \le k,\ \ell \in J_i}} \omega^v \Bigg) \varepsilon_u = f\varepsilon + \sum_{u \in \overline{\mathcal{J}^{n+}}} \omega^u \varepsilon_u$$

If $|u| = 1$, the word $u = J$ is related to the first order error due to the operations whose label belongs to J. The elementary operations on $\mathbb{R}^{\mathcal{J}^n}$ are defined by the equations (9) to (14) of Section 4 in which $\mathcal{W}$ now denotes the new monoid $(\overline{\mathcal{J}^n}, \cdot_n, \epsilon)$.

Note that this semantics generalizes the semantics of Section 5 which is based on a particular partition $\mathcal{J} = \{\{\ell\} \ : \ \ell \in \mathcal{L}\}$. Another interesting partition consists of using singletons for the initial data and collapsing all the other control points. This enables us to determine the contribution, to the global error, of initial errors on the program inputs (sensitivity analysis).

In the rest of this section, we compare the semantics based on different partitions of the labels. Intuitively, a partition $\mathcal{J}_2$ is coarser than a partition $\mathcal{J}_1$ if $\mathcal{J}_2$ collapses some of the elements of $\mathcal{J}_1$. For a maximal order of error n and using this ordering, the partition $\mathcal{J} = \{\{\ell\} : \ell \in \mathcal{L}\}$ corresponds to $\mathcal{S}^{\mathcal{L}^n}$ and is the finest partition of order n. We show that any semantics based on a partition $\mathcal{J}_2$, coarser than a partition $\mathcal{J}_1$, is an approximation of the semantics based on $\mathcal{J}_1$. Consequently, any semantics based on a partition of the control points is an approximation of the general semantics $\mathcal{S}^{\mathcal{L}^n}$ and $\mathcal{S}^{\mathcal{L}^*}$ defined in Section 4 and 5. The partial ordering on the set of partitions is defined below.

DEFINITION 1. *Let $\mathcal{J}_1$ and $\mathcal{J}_2$ be two partitions of the set $\mathcal{L}$. $\mathcal{J}_2$ is a coarser partition of $\mathcal{L}$ than $\mathcal{J}_1$ and we write $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$ iff $\forall J_1 \in \mathcal{J}_1,\ \exists J_2 \in \mathcal{J}_2 \ :\ J_1 \subseteq J_2$.*

If $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$ then some components of the partition $\mathcal{J}_1$ are collapsed in $\mathcal{J}_2$. $\dot{\subseteq}$ is used in the following to order the partitions of the set $\mathcal{L}$ of labels. The *translation* function $\tau_{\mathcal{J}_1^n, \mathcal{J}_2^n}$ maps words of the language $\overline{\mathcal{J}_1^n}$ to words of $\overline{\mathcal{J}_2^n}$ as follows.

$$\tau_{\mathcal{J}_1^n, \mathcal{J}_2^n}(J_1.u) = J_2.\tau_{\mathcal{J}_1^n, \mathcal{J}_2^n}(u) \text{ where } J_1 \in \mathcal{J}_1,\ J_1 \subseteq J_2,\ J_2 \in \mathcal{J}_2 \quad (28)$$

The correctness of any semantics based on a partition $\mathcal{J}_2$ of $\mathcal{L}$ stems from the fact that, for any $\mathcal{J}_2 \in \mathcal{P}(\mathcal{L})$ such that $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$, there is a

Galois connection

$$\langle \wp(\mathcal{F}(\mathbb{R},\overline{\mathcal{J}_1^n})),\subseteq\rangle \quad \underset{\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}}{\overset{\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}}{\leftrightarrows}} \quad \langle \wp(\mathcal{F}(\mathbb{R},\overline{\mathcal{J}_2^n})),\subseteq\rangle$$

defined by

$$\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}\left(\sum_{u\in\overline{\mathcal{J}_1^n}}\omega_i^u\varepsilon_u\right) \overset{\text{def}}{=} \sum_{u\in\overline{\mathcal{J}_1^n}}\omega^u\varepsilon_{\tau_{\mathcal{J}_1^n,\mathcal{J}_2^n}(u)}$$

$$\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}\left(\sum_{v\in\overline{\mathcal{J}_2^n}}\nu^u\varepsilon_v\right) \overset{\text{def}}{=} \left\{\sum_{u\in\overline{\mathcal{J}_1^n}}\omega^u\varepsilon_u \;:\; \sum_{\tau_{\mathcal{J}_1^n,\mathcal{J}_2^n}(u)=v}\omega^u=\nu^v\right\}$$

Again, for some sets X and Y, $\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}(X)=\{\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}(x) \;:\; x\in X\}$ and $\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}(Y)=\cup_{y\in Y}\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}(y)$.

Let J be an element of the coarser partition $\mathcal{J}_2$. For any first order error term $\omega^u\varepsilon_u$ attached to an error series $r^{\overline{\mathcal{J}_1^n}}=\sum_{u\in\overline{\mathcal{J}_1^n}}\omega^u\varepsilon_u$, the abstraction $\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}(r^{\overline{\mathcal{J}_1^n}})$ defines the coefficient ν_J attached to ε_J as being the sum of the coefficients $\omega_{J'}$ such that $J'\in\mathcal{J}_1$ and $J'\subseteq J$. The function $\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}$ returns the set of error series for which $\sum_{\tau_{\mathcal{J}_1^n,\mathcal{J}_2^n}(u)=v}\omega^u=\nu^v$. Lemma 3 assesses the correctness of the operations defined by Equations (9) to (14) for the domains introduced in this section.

LEMMA 3. *Let ℓ be a control point, let $\mathcal{J}_1$ and $\mathcal{J}_2$ be partitions of $\mathcal{L}$ such that $\mathcal{J}_1\dot{\subseteq}\mathcal{J}_2$ and let $r^{\mathcal{J}_1^n},s^{\mathcal{J}_1^n}\in\mathcal{F}(\mathbb{R},\overline{\mathcal{J}_1^n})$. If $r^{\mathcal{J}_2^n}=\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}(r^{\mathcal{J}_1^n})$, $s^{\mathcal{J}_2^n}=\alpha^{\mathcal{J}_1^n,\mathcal{J}_2^n}(s^{\mathcal{J}_1^n})$ then for any $\diamond\in\{+,-,\times,\div\}$ we have*

$$r^{\mathcal{J}_1^n}\diamond^\ell s^{\mathcal{J}_1^n}\in\gamma^{\mathcal{J}_2^n,\mathcal{J}_1^n}(r^{\mathcal{J}_2^n}\diamond^\ell s^{\mathcal{J}_2^n})$$

Proof. (Multiplication)
We use the notations

$$r^{\mathcal{J}_1^n}=\sum_{u\in\overline{\mathcal{J}_1^n}}\omega_r^u\varepsilon_u,\qquad s^{\mathcal{J}_1^n}=\sum_{u\in\overline{\mathcal{J}_1^n}}\omega_s^u\varepsilon_u,$$

$$r^{\mathcal{J}_2^n}=\sum_{u\in\overline{\mathcal{J}_2^n}}\nu_r^u\varepsilon_u,\qquad s^{\mathcal{J}_2^n}=\sum_{u\in\overline{\mathcal{J}_2^n}}\nu_s^u\varepsilon_u.$$

Let $\tau(u)=\tau_{\mathcal{J}_1^n,\mathcal{J}_2^n}(u)$. We have:

$$t^{\mathcal{J}_1^n}=r^{\mathcal{J}_1^n}\times^\ell s^{\mathcal{J}_1^n}=\uparrow_\circ(\omega_r^\epsilon\omega_s^\epsilon)\varepsilon_\epsilon+\sum_{\substack{u,v\,\in\,\overline{\mathcal{J}_1^n}\\ |u.v|>0}}\omega_r^u\omega_s^v\varepsilon_{u.v}+\downarrow_\circ(\omega_r^\epsilon\omega_s^\epsilon)\varepsilon_\ell$$

$$t^{\mathcal{J}_2^n} = r^{\mathcal{J}_2^n} \times^\ell s^{\mathcal{J}_2^n} = \uparrow_\circ(\nu_r^\epsilon \nu_s^\epsilon)\varepsilon_\epsilon + \sum_{\substack{u,v \in \overline{\mathcal{J}_2^n} \\ |u.v| > 0}} \nu_r^u \nu_s^v \varepsilon_{u.v} + \downarrow_\circ(\nu_r^\epsilon \nu_s^\epsilon)\varepsilon_\ell$$

The main step of the proof consists of showing that for all $u \in \overline{\mathcal{J}_2^n}$, $\sum_{\tau(v)=u} \omega_t^v = \nu_t^u$.

$$\begin{aligned}
\sum_{\tau(v)=u} \omega_t^v &= \sum_{\tau(v_1.v_2)=u} \omega_r^{v_1}\omega_s^{v_2} = \sum_{\substack{\tau(v_1).\tau(v_2) = u_1.u_2 \\ u_1.u_2 = u}} \omega_r^{v_1}\omega_s^{v_2} \\
&= \sum_{u_1.u_2=u} \Bigg(\sum_{\substack{\tau(v_1) = u_1 \\ \tau(v_2) = u_2}} \omega_r^{v_1}\omega_s^{v_2} \Bigg) \\
&= \sum_{u_1.u_2=u} \Bigg(\sum_{\tau(v_1)=u_1} \omega_r^{v_1} \times \sum_{\tau(v_2)=u_2} \omega_s^{v_2} \Bigg) = \sum_{u_1.u_2=u} \nu_r^{u_1}\nu_s^{u_2} = \nu_t^u
\end{aligned}$$

□

The semantics defined by the domain $\mathbb{R}^{\mathcal{J}^n}$ for values and by the reduction rules of Section 3 is denoted $\mathcal{S}^{\mathcal{J}^n}$. Proposition 3 establishes the link between the semantics $\mathcal{S}^{\mathcal{J}_1^n}$ and $\mathcal{S}^{\mathcal{J}_2^n}$ for comparable partitions $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$ of the set $\mathcal{L}$ of labels.

PROPOSITION 3. *Let $\mathcal{J}_1$ and $\mathcal{J}_2$ be partitions of $\mathcal{L}$ such that $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$ and let $a_0^{\ell_0 \mathcal{J}_1^n}$ and $a_0^{\ell_0 \mathcal{J}_2^n}$ be syntactically equivalent expressions such that for all $\ell \in \mathcal{L}$, we have:*

$$\mathcal{R}(a_0^{\ell_0 \mathcal{J}_2^n})(\ell) \subseteq \gamma^{\mathcal{J}_1^n, \mathcal{J}_2^n}(\mathcal{R}(a_0^{\ell_0 \mathcal{J}_1^n})(\ell))$$

If $a_0^{\ell_0 \mathcal{J}_1^n} \to a_1^{\ell_1 \mathcal{J}_1^n}$ then $a_0^{\ell_0 \mathcal{J}_2^n} \to a_1^{\ell_1 \mathcal{J}_2^n}$ such that $a_1^{\ell_1 \mathcal{J}_1^n}$ and $a_1^{\ell_1 \mathcal{J}_2^n}$ are syntactically equivalent expressions and for all $\ell \in \mathcal{L}$, $\mathcal{R}(a_1^{\ell_1 \mathcal{J}_2^n})(\ell) \subseteq \gamma^{\mathcal{J}_1^n, \mathcal{J}_2^n}(\mathcal{R}(a_1^{\ell_1 \mathcal{J}_2^n})(\ell))$.

The proof is by induction on the structure of the expression a_0.

As a consequence, for a given order of error n and for a given chain C of partitions, there is a chain of Galois connections between the semantics based on the partitions of C. Let us assume that

$$C = \Big((\{\{\ell\} \ : \ell \in \mathcal{L}\} \ = \mathcal{J}_0) \ \dot{\subseteq} \ \ldots \ \dot{\subseteq} \ \ldots \ \dot{\subseteq} \ \mathcal{J}_k \ \dot{\subseteq} \ \ldots \ \dot{\subseteq} \ \{\mathcal{L}\} \Big)$$

Then we have:

$$\mathcal{S}^{\mathcal{L}^n}(a_0^{\ell_0}) \ \leftrightarrows \ \ldots \ \mathcal{S}^{\mathcal{J}_k^n}(a_0^{\ell_0}) \ \leftrightarrows \ \ldots \ \leftrightarrows \ \mathcal{S}^{\{\mathcal{L}\}^n}(a_0^{\ell_0})$$

$$\begin{tikzcd}
\mathcal{S}^{\mathcal{J}^*_{k+1}}(a_0^{\ell_0}) \arrow[r, shift right] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right, "\alpha^{n+1,n}"'] \arrow[d, shift left] & \mathcal{S}^{\mathcal{J}^n_{k+1}}(a_0^{\ell_0}) \arrow[l, shift right, "\gamma^{n,n+1}"'] \arrow[r, shift right, "\alpha^{n,n-1}"'] \arrow[d, shift left, "\gamma^{\mathcal{J}^n_{k+1},\mathcal{J}^n_k}"] & \cdots \arrow[l, shift right, "\gamma^{n-1,n}"'] \arrow[r, shift right] \arrow[d, shift left] & \mathcal{S}^{\mathcal{J}^0_{k+1}}(a_0^{\ell_0}) \arrow[l, shift right] \arrow[d, shift left] \\
\mathcal{S}^{\mathcal{J}^*_{k}}(a_0^{\ell_0}) \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \mathcal{S}^{\mathcal{J}^n_{k}}(a_0^{\ell_0}) \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left, "\alpha^{\mathcal{J}^n_k,\mathcal{J}^n_{k+1}}"] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \mathcal{S}^{\mathcal{J}^0_{k}}(a_0^{\ell_0}) \arrow[l, shift right] \arrow[u, shift left] \arrow[d, shift left] \\
\cdots \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] \arrow[d, shift left] & \cdots \arrow[l, shift right] \arrow[u, shift left] \arrow[d, shift left] \\
\mathcal{S}^{\mathcal{L}^*}(a_0^{\ell_0}) \arrow[r, shift right] \arrow[u, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] & \mathcal{S}^{\mathcal{L}^n}(a_0^{\ell_0}) \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] & \cdots \arrow[l, shift right] \arrow[r, shift right] \arrow[u, shift left] & \mathcal{S}^{\mathcal{L}^0}(a_0^{\ell_0}) \arrow[l, shift right] \arrow[u, shift left]
\end{tikzcd}$$

Figure 2. Links between the semantics $\mathcal{S}^{\mathcal{J}^n_k}$ for a given order of error n and for a chain of partitions $\{\{\ell\}\ :\ell \in \mathcal{L}\} \dot{\subseteq}\ \mathcal{J}_1\ \dot{\subseteq}\ \ldots\ \dot{\subseteq}\ \mathcal{J}_k\ \dot{\subseteq}\ \ldots\ \dot{\subseteq}\{\mathcal{L}\}$.

By combining Proposition 2 and Proposition 3, we can also link the semantics $\mathcal{S}^{\mathcal{J}^n_k}$ and $\mathcal{S}^{\mathcal{J}^{n+1}_k}$ for any $\mathcal{J}_k \in C$ and any $n \in \mathbb{N}$. This is summed up in Figure 2. For any integer n and partition $\mathcal{J}_k$, $\mathcal{S}^{\mathcal{J}^n_k}$ describes a particular semantics:

– $\mathcal{S}^{\mathcal{L}^*}$ is the most informative semantics,

– conversely, the semantics $\mathcal{S}^{\mathcal{L}^0}$ that computes one global error term is the least informative semantics,

– for all $k > 0$, $\mathcal{S}^{\mathcal{J}^0_k} = \mathcal{S}^{\mathcal{L}^0}$ (for $n = 0$, any partition yields the same semantics),

– $\mathcal{S}^{\mathcal{J}^2_0}$ computes the global first order and second order errors made during a computation,

– for any $\mathcal{J}_k$, $\mathcal{S}^{\mathcal{J}^1_k}$ computes the contribution to the global error of the first order errors made in the different pieces of code identified by $\mathcal{J}_k$.

Let us remark that the values in $\mathbb{R}^{\mathcal{J}^n_2}$ contain fewer terms than the ones in $\mathbb{R}^{\mathcal{J}^n_1}$ if $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$. Hence, using coarser partitions leads to significantly fewer computations.

7. Dynamic partitioning of the control points

As shown in Figure 2, many semantics can be used to compute in more or less detail the contribution to the global error of the roundoff errors arising during a finite precision calculation. In practice, the number of terms of the series is a crucial parameter of static analyses. In this section, we show how the choice of a particular semantics $\mathcal{S}^{\mathcal{J}^n}$ affects the precision and the execution-time of a static analysis relying on it. Then we discuss the problem of moving from one semantics $\mathcal{S}^{\mathcal{J}_1^n}$ to another semantics $\mathcal{S}^{\mathcal{J}_2^m}$ during an analysis, in order to improve its performance. This problem is called *dynamic partitioning.*

First of all, let $\langle \mathcal{I}_{\mathbb{F}}, \subseteq \rangle$ denote the domain of intervals with floating-point bounds and let us consider the abstraction:

$$\langle \wp(\mathcal{F}(\mathbb{R}, \overline{\mathcal{J}^n})), \subseteq \rangle \quad \underset{\alpha^{\mathcal{I}}}{\overset{\gamma^{\mathcal{I}}}{\leftrightarrows}} \quad \langle \mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}^n}), \sqsubseteq \rangle$$

In $\mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}^n})$, the error series are ordered componentwise, i.e.:

$$\sum_{u \in \overline{\mathcal{J}^n}} \omega_1^u \varepsilon_u \sqsubseteq \sum_{u \in \overline{\mathcal{J}^n}} \omega_2^u \varepsilon_u \iff \forall u \in \overline{\mathcal{J}^n},\ \omega_1^u \subseteq \omega_2^u \tag{29}$$

Let X be a set of floating-point numbers and let $\Phi\ :\ \wp(\mathbb{F}) \to \mathcal{I}_{\mathbb{F}}$ be the function returning the convex hull $\Phi(X)$ of X, i.e. the least interval of $\mathcal{I}_{\mathbb{F}}$ containing the set X. $\alpha^{\mathcal{I}}$ and $\gamma^{\mathcal{I}}$ are defined by:

$$\alpha^{\mathcal{I}} \left(\left\{ \sum_{u \in \overline{\mathcal{J}^n}} \omega_i^u \varepsilon_u\ :\ i \in I \right\} \right) \stackrel{\text{def}}{=} \sum_{u \in \overline{\mathcal{J}^n}} \Phi\left(\{ \omega_i^u\ :\ i \in I \} \right) \varepsilon_u \tag{30}$$

and

$$\gamma^{\mathcal{I}} \left(\sum_{u \in \overline{\mathcal{J}^n}} \nu^u \varepsilon_u \right) \stackrel{\text{def}}{=} \left\{ \sum_{u \in \overline{\mathcal{J}^n}} \omega^u \varepsilon_u\ :\ \forall u \in \overline{\mathcal{J}^n},\ \omega^u \in \nu^u \right\} \tag{31}$$

In the following, we illustrate how the choice of a partition affects a static analysis. Let $m \leq n$, let $\mathcal{J}_1$ and $\mathcal{J}_2$ be two partitions such that $\mathcal{J}_1 \dot{\subseteq} \mathcal{J}_2$ and let $r^{\mathcal{J}_1^n} \in \mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}_1^n})$, $s^{\mathcal{J}_1^n} \in \mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}_1^n})$, $r^{\mathcal{J}_2^m} \in \mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}_2^m})$, and $s^{\mathcal{J}_2^m} \in \mathcal{F}(\mathcal{I}_{\mathbb{F}}, \overline{\mathcal{J}_2^m})$ such that

$$r^{\mathcal{J}_2^m} = \alpha^{\mathcal{I}} \circ \alpha^{\mathcal{J}_1^n, \mathcal{J}_2^m} \circ \gamma^{\mathcal{I}}(r^{\mathcal{J}_1^n})$$

and

$$s^{\mathcal{J}_2^m} = \alpha^{\mathcal{I}} \circ \alpha^{\mathcal{J}_1^n, \mathcal{J}_2^m} \circ \gamma^{\mathcal{I}}(s^{\mathcal{J}_1^n})$$

It often happens that $r^{\mathcal{J}_2^m} \sqsubseteq s^{\mathcal{J}_2^m}$ while $r^{\mathcal{J}_1^n} \not\sqsubseteq s^{\mathcal{J}_1^n}$ and $s^{\mathcal{J}_1^n} \not\sqsubseteq r^{\mathcal{J}_1^n}$. In other words, two values may be comparable in the coarser semantics while being incomparable in the finer. This point is crucial for the static analysis of loops, when the halting condition depends on a comparison between abstract values. The analysis may terminate after less iterations in $\mathcal{S}^{\overline{\mathcal{J}_2^m}}$ than in $\mathcal{S}^{\overline{\mathcal{J}_1^n}}$. As an example, let us consider the loop:

```
|1| while !(x<=y) {
|2|   y = x;
|3|   x = x+a;
|4|   x = b*x;
|5| }
```

We assume that initially:

$$a = [0.0, 0.14] + [0.0, 0.002]\varepsilon_a \quad b = [0.5, 0.7]$$

$$x = [0.0, 0.66] + [0.0, 0.006]\varepsilon_x \quad y = [0.0, 0.0]$$

As in Section 2, we use for this example a simplified set of floating-point numbers whose mantissa is made up of two digits written in base 10. We use a partition that associates one control point per line of code. Let $\alpha = [0.0, 0.002]$ and $\beta = [0.0, 0.006]$. The successive values of x at lines |3| and |4| are:

$$\begin{array}{lrcl}
\texttt{|3|} & x & = & [0.0, 0.80]\varepsilon + \alpha\varepsilon_a + \beta\varepsilon_x \\
\texttt{|4|} & x & = & [0.0, 0.56]\varepsilon + [0.5, 0.7]\alpha\varepsilon_a + [0.5, 0.7]\beta\varepsilon_x \\
 & & = & [0.0, 0.56]\varepsilon + [0.0, 0.0014]\varepsilon_a + [0.0, 0.0042]\varepsilon_x \\
\texttt{|3|} & x & = & [0.0, 0.70]\varepsilon + (\alpha + 0.7\alpha)\varepsilon_a + 0.7\beta\varepsilon_x \\
\texttt{|4|} & x & = & [0.0, 0.49]\varepsilon + (0.7\alpha + 0.7^2\alpha)\varepsilon_a + 0.7^2\beta\varepsilon_x \\
 & & = & [0.0, 0.49]\varepsilon + [0.0, 0.00238]\varepsilon_a + [0.0, 0.00294]\varepsilon_x
\end{array}$$

Let x_1 and x_2 denote the value of x after one and two iterations. We have:

$$x_1 = [0.0, 0.56]\varepsilon + [0.0, 0.0014]\varepsilon_a + [0.0, 0.0042]\varepsilon_x \tag{32}$$

$$x_2 = [0.0, 0.49]\varepsilon + [0.0, 0.00238]\varepsilon_a + [0.0, 0.00294]\varepsilon_x \tag{33}$$

In our semantics, $x_2 \not\sqsubseteq x_1$ because of the coefficient of ε_a and some more iterations must be carried out to complete the static analysis of the program. In addition, while $x_2 \not\sqsubseteq x_1$, the static analyzer may perform a widening that would introduce a huge imprecision. In this case, one could obtain:

$$x = x_1 \nabla x_2 = [0.0, 0.56]\varepsilon + [0.0, +\infty]\varepsilon_a + [0.0, 0.0042]\varepsilon_x$$

However, in the coarser partition that merges the errors ε_a and ε_x, we have:

$$x_1' = [0.0, 0.56]\varepsilon + [0.0, 0.0056]\varepsilon_{a,x} \tag{34}$$

$$x_2' = [0.0, 0.49]\varepsilon + [0.0, 0.00532]\varepsilon_{a,x} \tag{35}$$

and $x_2' \sqsubseteq x_1'$. This illustrates how convergence properties may depend on the precision of the partition. On one hand, we wish to keep the finest partition in order to know precisely which error terms mainly contribute to the global error. On the other hand, we wish to merge some error terms in order to accelerate the convergence of the calculations and to avoid the widenings. This problem often happens in static analyzers based on the semantics introduced in this article. For example, it arises in the Fluctuat abstract interpreter which uses abstract series whose coefficients are intervals of multi-precision floating-point numbers [13, 30]. Reducing the number of iterations in fixed point calculations significantly improves the performance of the tool.

Given two series x_1 and x_2 indexed by a set $\mathcal{J}$ of control points, like those of Equations (32) and (33), we aim at finding the greatest partition $\mathcal{J}'$ of $\mathcal{J}$ that makes $\alpha^{\mathcal{J},\mathcal{J}'}(x_1) \sqsubseteq \alpha^{\mathcal{J},\mathcal{J}'}(x_2)$.

For the sake of simplicity, we focus on the upper bounds of the intervals used as coefficients of the series. The dynamic partitioning problem is formally specified in Definition 2.

DEFINITION 2 (Dynamic Partitioning Problem (DP)). *Given two n-tuples of non-negative integers $W = \langle \omega_1, \dots, \omega_n \rangle$ and $W' = \langle \omega_1', \dots, \omega_n' \rangle$, is there a partition $\mathcal{P}$ of $\{1, \dots, n\}$, of size t, such that*

$$\forall X \in \mathcal{P},\ \sum_{i \in X} \omega_i \leq \sum_{i \in X} \omega_i' \tag{36}$$

An equivalent problem can be defined for the lower bounds of the intervals. For the sake of simplicity and without loss of generality, DP is defined for integer coefficients.

PROPOSITION 4. *DP is NP-complete.*

The proof is by transformation from the well-known NP-complete problem Partition [9], whose definition is recalled below.

DEFINITION 3 (Partition). *Given a finite set of positive integers $A = \{a_1, \dots, a_n\}$, is there a subset I of A such that*

$$\sum_{a_i \in I} a_i = \sum_{a_i \in A \setminus I} a_i$$

Proof. (Proposition 4) DP belongs to NP since it is a trivial matter to check a solution in polynomial-time. Let $A = \{a_1, \dots, a_n\}$ be an arbitrary instance I_1 of Partition. We build, in polynomial-time from

I_1, the instance I_2 of DP defined by Equations (37-41). As in Definition 2, t denotes the number of classes of the partition.

$$\begin{aligned} t &= 2 && (37)\\ \omega_i &= \Big(\sum_{a_j \in A} a_j\Big) + a_i,\ 1 \le i \le n && (38)\\ \omega'_i &= \Big(\sum_{a_j \in A} a_j\Big) - a_i,\ 1 \le i \le n && (39)\\ \omega_{n+1} = \omega_{n+2} &= 0 && (40)\\ \omega'_{n+1} = \omega'_{n+2} &= \sum_{a_j \in A} a_j && (41) \end{aligned}$$

By hypothesis, we assume that there exists a polynomial time algorithm for DP. This algorithm finds a solution to I_2 which satisfies:

$$\forall X \in \mathcal{P},\ \sum_{i \in X} \omega_i \le \sum_{i \in X} \omega'_i \qquad (42)$$

First of all, we note the following fact: since $t = 2$, $\mathcal{P} = \{X, \overline{X}\}$, where $\overline{X} = \{1, 2, \ldots, n+2\} \setminus X$. In addition, $n+1$ and $n+2$ do not belong to the same class. This stems from the fact that $\omega'_i < \omega_i$, for $1 \le i \le n$. If $n+1$ and $n+2$ belong to X (resp. $\overline{X}$), then Equation (42) would not be satisfied by $\overline{X}$ (resp. X). We focus now on the set X for which we have:

$$\begin{aligned} \sum_X \omega_i &\le \sum_X \omega'_i\\ |X| \sum_A a_i + \sum_X a_i &\le |X| \sum_A a_i - \sum_X a_i + \omega'_{n+1}\\ \sum_X a_i &\le \omega'_{n+1} - \sum_X a_i\\ \sum_X a_i &\le \sum_A a_i - \sum_X a_i = \sum_{\overline{X}} a_i \end{aligned}$$

Next, let us focus on $\overline{X}$. Using the same proof as above, we obtain

$$\sum_{\overline{X}} a_i \le \sum_X a_i$$

and the equality of both quantities follows. We found a partition $\mathcal{P}$ of A which is a solution to Partition. Hence, DP is NP-complete. □

Even if DP is NP-complete in the most general case, some heuristics work well in practice. For example, the initial semantics may use a partition that associates different control points to the first i instances of all the operations occurring in a loop. Next, for comparisons, the heuristic merges all the error terms related to the same operation. In practice, this gives good results for stable calculations like the loop of the example developed in this section.

8. Conclusion

The semantics introduced in this article models the propagation of roundoff errors and the introduction of new errors at each stage of a calculation. We use a unified framework, mainly based on the equations of Figure 1, to compute the contribution, to the global error, of the errors due to pieces of code partitioning the program, up to a maximal order of error. Lemma 1 and Proposition 1 are essential to ensure the correctness of the operators of Figure 1. They also represent a stronger correctness criterion for the operators introduced in [11]. Another important point is that $\mathcal{S}^{\mathcal{L}^n}$ not only details the propagation of the errors of order $\leq n$ but also verifies that higher order error terms actually are negligible. Finally we discussed the complexity of the dynamic partitioning problem that enables to optimally move from one semantics to another in the grid of Figure 2. Even if this problem is NP-complete in the most general case, some heuristics work well in practice.

A tool called Fluctuat has been developed which implements an abstract interpretation based on the semantics introduced in this article. The real coefficients of the error series are abstracted by intervals of multi-precision floating-point numbers. This tool is described in [13, 30]. Current work concerns the precision of the abstract interpretation in loops and is proceeding in two directions. First, because narrowing operators [7] do not yield information to improve the precision of the error terms, we really need finely-tuned widening operators for our domain. This should enable us to restrict the number of cases where a loop is stable but is declared unstable by the abstract interpreter because of the approximations made during the analysis. The second way to improve the precision in loops consists of using a relational analysis. A first solution was proposed in [24]. It can be viewed as an automatic differentiation technique applied to the semantic equations of Figure 1. The termination criterion of the analysis relies on an abstract calculation of the Lyapunov exponents [1].

Acknowledgements

I would like to thank Eric Goubault and Sylvie Putot for their many suggestions to improve this work. I also wish to thank Nicky Williams and the reviewers for their helpful comments. This work was supported in part by the RTD project IST-1999-20527 DAEDALUS of the european IST FP5 programme.

References

1. K. T. Alligood, T. D. Sauer, and J. A. Yorke. *Chaos, an Introduction to Dynamical Systems.* Springer-Verlag, 1996. ISBN 0-387-94677-2.
2. ANSI/IEEE. *IEEE Standard for Binary Floating-point Arithmetic*, std 754-1985 edition, 1985.
3. C. Bischof, P. D. Hovland, and B. Norris. Implementation of automatic differentiation tools. In *Proceedings of the ACM-SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Transformations, PEPM'02*, pages 98–107. ACM Press, 2002.
4. S. Boldo and M. Daumas. Representable correcting terms for possibly underflowing floating point operations. In J.-C. Bajard and M. Schulte, editors, *Proceedings of the 16th Symposium on Computer Arithmetic*, pages 79–86. IEEE Press, 2003.
5. J.-M. Chesneaux. L'arithmétique stochastique et le logiciel CADNA. Habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, 1995.
6. 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.
7. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Symbolic Computation*, 2(4):511–547, 1992.
8. M. Daumas, L. Rideau, and L Théry. A generic library for floating-point numbers and its application to exact computing. In *TPHOLs'01, International Conference on Theorem Proving and Higher Order Logics*, number 2152 in Lecture Notes in Computer Science, pages 169–184. Springer-Verlag, 2001.
9. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979. ISBN 0-7167-1045-5.
10. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.
11. E. Goubault. Static analyses of the precision of floating-point operations. In *Static Analysis Symposium, SAS'01*, number 2126 in Lecture Notes in Computer Science, pages 234–259. Springer-Verlag, 2001.
12. E. Goubault, M. Martel, and S. Putot. Concrete and abstract semantics of floating-point operations. Technical Report DRT/LIST/DTSI/SLA/LSL/01-058, CEA, 2001.
13. E. Goubault, M. Martel, and S. Putot. Asserting the precision of floating-point computations: a simple abstract interpreter. In 11^{th} *European Symposium on*

Programming, ESOP'02, number 2305 in Lecture Notes in Computer Science, pages 209–212, 2002.

14. A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation.* Frontiers in Applied Mathematics. SIAM (Publisher), 2000. ISBN 0-89871-451-6.
15. G. Hanrot, V. Lefevre, F. Rouillier, and P. Zimmermann. The MPFR library. Institut de Recherche en Informatique et Automatique, `www.loria.fr/projets/mpfr`, 2001.
16. J. Harrison. A machine-checked theory of floating point arithmetic. In *TPHOLs'99, International Conference on Theorem Proving and Higher Order Logics*, number 1690 in Lecture Notes in Computer Science, pages 113–130. Springer-Verlag, 1999.
17. J. R. Hauser. Handling floating-point exceptions in numeric programs. *ACM Transactions on Programming Languages and Systems*, 18(2):139–174, 1996.
18. N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM (Publisher), 1996. ISBN 0-89871-355-2.
19. D. Knuth. *The Art of Computer Programming - Seminumerical Algorithms.* Addison Wesley, third edition, 1997. Chapter 4, ISBN 0-201-89684-2.
20. P. Langlois. Automatic linear correction of rounding errors. *BIT, Numerical Mathematics*, 41(3):515–539, 2001.
21. P. Langlois and F. Nativel. Improving automatic reduction of round-off errors. In *IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, volume 2, pages 359–364. Wissenshaft und Teknik Verlag, 1997.
22. V. Lefevre, J.M. Muller, and A. Tisserand. Toward correctly rounded transcendentals. *IEEE Transactions on Computers*, 47(11):1235–1243, 1998.
23. M. Martel. Propagation of roundoff errors in finite precision computations: a semantics approach. In 11^{th} *European Symposium on Programming, ESOP'02*, number 2305 in Lecture Notes in Computer Science, pages 194–208. Springer-Verlag, 2002.
24. M. Martel. Static analysis of the numerical stability of loops. In *Static Analysis Symposium, SAS'02*, number 2477 in Lecture Notes in Computer Science, pages 133–150. Springer-Verlag, 2002.
25. D. W. Matula and P. Kornerup. Foundations of finite precision rational arithmetic. *Journal of Computing*, 2:378–387, 1980.
26. C. Michel, M. Rueher, and Y. Lebbah. Solving constraints over floating-point numbers. In *CP'2001, Seventh International Conference on Principles and Practice of Constraint Programming*, number 2239 in Lecture Notes in Computer Science, pages 524–538. Springer-Verlag, 2001.
27. R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, 1963.
28. M. Mrozek. Rigorous error analysis of numerical algorithms via symbolic computations. *Journal of Symbolic Computation*, 22:435–458, 1996.
29. M. Priest. Algorithms for arbitrary precision floating point arithmetic. In P. Kornerup and D. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–144. IEEE Computer Society Press, 1991.
30. S. Putot, E. Goubault, and M. Martel. Static analysis based validation of floating-point computations. In *Numerical Software with Result Verification*, number 2991 in Lecture Notes in Computer Science, pages 306–313, 2004.
31. J. Vignes. A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, 35(3):233–261, 1993.