Exploiting Binary Floating-Point Representations for Constraint Propagation

Roberto Bagnara
BUGSENG srl and Dept. of Mathematics and Computer Science, University of Parma, Italy,
`bagnara@cs.unipr.it`, `http://www.cs.unipr.it/~bagnara`

Matthieu Carlier
INRIA Rennes Bretagne Atlantique, France

Roberta Gori
Dept. of Computer Science, University of Pisa, Italy,
`gori@di.unipi.it`, `http://www.di.unipi.it/~gori`

Arnaud Gotlieb
Certus Software V&V Center, SIMULA Research Laboratory, Norway,
`arnaud@simula.no`, `http://simula.no/people/arnaud`

Floating-point computations are quickly finding their way in the design of safety- and mission-critical systems, despite the fact that designing floating-point algorithms is significantly more difficult than designing integer algorithms. For this reason, verification and validation of floating-point computations is a hot research topic. An important verification technique, especially in some industrial sectors, is testing. However, generating test data for floating-point intensive programs proved to be a challenging problem. Existing approaches usually resort to random or search-based test data generation, but without symbolic reasoning it is almost impossible to generate test inputs that execute complex paths controlled by floating-point computations. Moreover, as constraint solvers over the reals or the rationals do not natively support the handling of rounding errors, the need arises for efficient constraint solvers over floating-point domains. In this paper, we present and fully justify improved algorithms for the propagation of arithmetic IEEE 754 binary floating-point constraints. The key point of these algorithms is a generalization of an idea by B. Marre and C. Michel that exploits a property of the representation of floating-point numbers.

Key words: software verification; testing; floating-point numbers; constraint solving

1. Introduction

During the last decade, the use of floating-point computations in the design of critical systems has become increasingly acceptable. Even in the civil and military avionics domain, which are among the most critical domains for software, floating-point numbers are now seen as a sufficiently-safe, faster and cheaper alternative to fixed-point arithmetic. To the point that, in modern avionics, floating-point is the norm rather than the exception (Burdy et al. 2012).

Acceptance of floating-point computations in the design of critical systems took a long time. In fact, rounding errors can cause subtle bugs which are often missed by non experts (Monniaux

2008), and can lead to catastrophic failures. For instance, during the first Persian Gulf War, the failure of a Patriot missile battery in Dhahran was traced to an accumulating rounding error in the continuous execution of tracking and guidance software: this failure prevented the interception of an Iraqi Scud that hit the barracks in Dhahran, Saudi Arabia, killing 28 US soldiers (Skeel 1992). A careful analysis of this failure revealed that, even though the rounding error obtained at each step of the floating-point computation was very small, the propagation during a long loop-iterating path could lead to dramatic imprecision.

Adoption of floating-point computations in critical systems involves the use of thorough unit testing procedures that are able to exercise complex chains of floating-point operations. In particular, a popular practice among software engineers in charge of the testing of floating-point-intensive computations consists in executing carefully chosen loop-iterating paths in programs. They usually pay more attention to the paths that are most likely to expose the system to unstable numerical computations.[1] For critical systems, a complementary requirement is to demonstrate the infeasibility of selected paths, in order to convince a third-party certification authority that certain unsafe behaviors of the systems cannot be reached. As a consequence, software engineers face two difficult problems:

1. How to accurately predict the expected output of a given floating-point computation?[2]
2. How to find a test input that is able to exercise a given path, the execution of which depends on the results of floating-point computations, or to guarantee that such a path is infeasible?

The first problem has been well addressed in the literature (Kuliamin 2010) through several techniques. Ammann and Knight (1988) report on a technique known as the *data diversity* approach, which uses multiple related program executions of a program to check their results. *Metamorphic testing* (Chan et al. 1998) generalizes this technique by using known numerical relations of the function implemented by a program to check the results of two or more executions. Goubault (2001) proposes using the *abstract interpretation* framework (Cousot and Cousot 1977) to estimate the deviation of the floating-point results with respect to an interpretation over the reals. Scott et al. (2007) propose using a probabilistic approach to estimate round-off error propagation. More recently, Tang et al. (2010) propose to exploit perturbation techniques to evaluate the stability of a numerical program. In addition to these approaches, it is possible to use a (partial) specification, a prototype or an old implementation in order to predict the results for a new implementation.

[1] A computation can be called *numerically stable* if it can be proven not to magnify approximation errors. It can be called *(potentially) unstable* otherwise.

[2] This is the well-known *oracle problem* (see Weyuker 1982).

In contrast, the second problem received only little attention. Beyond the seminal work of Miller and Spooner (1976), who proposed to guide the search of floating-point inputs to execute a selected path, few approaches try to exactly reason over floating-point computations. The work of Miller and Spooner (1976) paved the way to the development of *search-based test data generation* techniques, which consist in searching test inputs by minimizing a cost function, evaluating the distance between the currently executed path and a targeted selected path (Korel 1990, Lakhotia et al. 2010a, McMinn 2004). Although these techniques enable quick and efficient coverage of testing criteria such as "all decisions," they are unfortunately sensitive to the rounding errors incurred in the computation of the branch distance (Arcuri 2009). Moreover, search-based test data generation cannot be used to study path feasibility, i.e., to decide whether a possible execution path involving floating-point computations is feasible or not in the program. In addition, these techniques can be stuck in local minima without being able to provide a meaningful result (Arcuri 2009). An approach to tackle these problems combines program execution and symbolic reasoning (Godefroid et al. 2005), and requires solving constraints over floating-point numbers in order to generate test inputs that exercise a selected behavior of the program under test. However, solving floating-point constraints is hard and requires dedicated filtering algorithms (Michel 2002, Michel et al. 2001). According to our knowledge, this approach is currently implemented in four solvers only: ECLAIR[3], FPCS (Blanc et al. 2006), FPSE[4] (Botella et al. 2006), and GATeL, a test data generator for Lustre programs (Marre and Blanc 2005). It is worth noticing that existing constraint solvers dedicated to continuous domains (such as, e.g., RealPaver (Granvilliers and Benhamou 2006), IBEX and Quimper (Chabert and Jaulin 2009) or ICOS (Lebbah 2009)) correctly handle real or rational computations, but they cannot preserve the solutions of constraints over floating-point computations in all cases (see Section 5 for more on this subject). "Surprising" properties of floating-point computations such as absorption and cancellation (Goldberg 1991) show that the rounding operations can severely compromise the preservation of the computation semantics between the reals and the floats.

Example 1. Consider the C functions `f1` and `f2`:

```c
float f1(float x) {
  float y = 1.0e12F;
  if (x < 10000.0F)
     z = x + y;
  if (z > y)
     ...
```

```c
float f2(float x) {
  float y = 1.0e12F;
  if (x > 0.0F)
    z = x + y;
  if(z == y)
    ...
```

[3] `http://bugseng.com/products/eclair`

[4] `http://www.irisa.fr/celtique/carlier/fpse.html`

For both functions, let us ask the question whether the paths traversing lines 2-3-4-5-6 are feasible. The condition that must be satisfied in order for a certain path to be traversed is called *path condition*. For `f1`, the path conditions $\mathtt{x} < 10000.0$ and $\mathtt{x} + 1.0\mathrm{e}12 > 1.0\mathrm{e}12$, which on the reals are equivalent to $\mathtt{x} \in (0, 10000)$ whereas on the floats they have no solution. Conversely, for `f2` the path conditions are $\mathtt{x} > 0.0$ and $\mathtt{x} + 1.0\mathrm{e}12 = 1.0\mathrm{e}12$, which have no solutions on the reals but are satisfied by all IEEE 754 single precision floating-point numbers in the range $(0, 32767.99\cdots)$.

A promising approach to improve the filtering capabilities of constraints over floating-point variables consists in using some peculiar numerical properties of floating-point numbers. For linear constraints, this led to a relaxation technique where floating-point numbers and constraints are converted into constraints over the reals by using linear programming approaches (Belaid et al. 2012). For interval-based consistency approaches, Marre and Michel (2010) identified a property of the representation of floating-point numbers and proposed to exploit it in filtering algorithms for addition and subtraction constraints. Carlier and Gotlieb (2011) proposed a reformulation of the Marre-Michel property in terms of "filtering by maximum ULP" (*Units in the Last Place*) that is generalizable to multiplication and division constraints.

Bagnara et al. (2013) addressed the question of whether the Marre-Michel property can be useful for the automatic solution of realistic test input generation problems: they sketched (without proofs) a reformulation and correction of the filtering algorithm proposed in (Marre and Michel 2010), along with a uniform framework that generalizes the property identified by Marre and Michel to the case of multiplication and division. Most importantly, (Bagnara et al. 2013) presented the implementation of filtering by maximum ULP in FPSE and some of its critical design choices, and an experimental evaluation on constraint systems that have been extracted from programs engaging into intensive floating-point computations. These results show that the Marre-Michel property and its generalization are effective, practical properties for solving constraints over the floats with an acceptable overhead. The experiments reported in (Bagnara et al. 2013) showed that improvement of filtering procedures with these techniques brings speedups of the overall constraint solving process that can be substantial (we have observed up to an order of magnitude); in the cases where such techniques do not allow significant extra-pruning, the slowdowns are always very modest (up to a few percent on the overall solution time).

The present paper is, on the one hand, the theoretical counterpart of (Bagnara et al. 2013) in that all the results are thoroughly proved; on the other hand, this paper generalizes and extends (Bagnara et al. 2013) as far as the handling of subnormals and floating-point division are concerned. More precisely, the contributions of the paper are:

1. a uniform framework for filtering by maximum ULP is thoroughly defined and justified;

2. the framework is general enough to encompass all floating-point arithmetic operations and subnormals (the latter are not treated in (Bagnara et al. 2013));

3. a second indirect projection by maximum ULP for division (not present in any previous work);

4. all algorithms only use floating-point machine arithmetic operations on the same formats used by the analyzed computations.

The plan of the paper is as follows. Next section presents the IEEE 754 standard of binary floating-point numbers and introduces the notions and notations used throughout the paper. Section 3 recalls the basic principles of interval-based consistency techniques over floating-point variables and constraints. Section 4 presents our generalization of the Marre-Michel property along with a precise definition and motivation of all the required algorithms. Section 5 discusses related work. Section 6 concludes. The most technical proofs are available in the Online Supplement to this paper (Bagnara et al. 2015).

2. Preliminaries

In this section we recall some preliminary concepts and introduce the used notation.

2.1. IEEE 754

This section recalls the arithmetic model specified by the IEEE 754 standard for binary floating-point arithmetic (IEEE Computer Society 2008). Note that, although the IEEE 754 standard also specifies formats and methods for decimal floating-point arithmetic, in this paper we only deal with binary floating-point arithmetic.

IEEE 754 binary floating-point formats are uniquely identified by quantities: $p \in \mathbb{N}$, the number of significant digits (precision); $e_{\max} \in \mathbb{N}$, the maximum exponent; $-e_{\min} \in \mathbb{N}$, the minimum exponent.[5] The *single precision* format has $p = 24$ and $e_{\max} = 127$, the *double precision* format has $p = 53$ and $e_{\max} = 1023$ (IEEE 754 also defines extended precision formats). A finite, non-zero IEEE 754 floating-point number z has the form $(-1)^s b_1.m \times 2^e$ where s is the *sign bit*, b_1 is the *hidden bit*, m is the $(p-1)$-bit *significand* and the *exponent* e is also denoted by e_z or $\exp(z)$. Hence the number is positive when $s = 0$ and negative when $s = 1$. b_1 is termed "hidden bit" because in the *binary interchange format encodings* it is not explicitly represented, its value being encoded in the exponent (IEEE Computer Society 2008).

Each format defines several classes of numbers: normal numbers, subnormal numbers, signed zeros, infinities and NaNs (*Not a Number*). The smallest positive *normal* floating-point number is $f_{\min}^{\mathrm{nor}} = 1.0 \cdots 0 \times 2^{e_{\min}} = 2^{e_{\min}}$ and the largest is $f_{\max} = 1.1 \cdots 1 \times 2^{e_{\max}} = 2^{e_{\max}}(2 - 2^{1-p})$; normal numbers have the hidden bit $b_1 = 1$. The non-zero floating-point numbers whose absolute value

[5] Note that, although the IEEE 754 formats have $e_{\min} = 1 - e_{\max}$, we never use this property and decided to keep the extra-generality, which might be useful to accommodate other formats.

is less than $2^{e_{\min}}$ are called *subnormals*: they always have exponent equal to $e_{\min}$ and fewer than p significant digits as their hidden bit is $b_1 = 0$. Every finite floating-point number is an integral multiple of the smallest subnormal $f_{\min} = 0.0\cdots01 \times 2^{e_{\min}} = 2^{e_{\min}+1-p}$. There are two infinities, denoted by $+\infty$ and $-\infty$, and two *signed zeros*, denoted by $+0$ and -0: they allow some algebraic properties to be maintained (Goldberg 1991).[6] NaNs are used to represent the results of invalid computations such as a division of two infinities or a subtraction of infinities with the same sign: they allow the program execution to continue without being halted by an exception.

IEEE 754 defines five rounding directions: toward negative infinity (*roundTowardNegative* or, briefly, *down*), toward positive infinity (*roundTowardPositive*, a.k.a. *up*), toward zero (*roundTowardZero*, a.k.a. *chop*) and toward the nearest representable value (a.k.a. *near*); the latter comes in two flavors that depend on different tie-break rules for numbers exactly halfway between two representable numbers: *roundTiesToEven* (a.k.a. *tail-to-even*) or *roundTiesToAway* (a.k.a. *tail-to-away*) in which values with even significand or values away from zero are preferred, respectively. This paper is only concerned with roundTiesToEven, which is, by far, the most widely used. The roundTiesToEven value of a real number x will be denoted by $[x]_{\mathrm{n}}$.

The most important requirement of IEEE 754 arithmetic is the accuracy of floating-point computations: add, subtract, multiply, divide, square root, remainder, conversion and comparison operations must deliver to their destination the exact result rounded as per the rounding mode in effect and the format of the destination. It is said that these operations are "correctly rounded."

The accuracy requirement of IEEE 754 can still surprise the average programmer: for example the single precision, round-to-nearest addition of 999999995904 and 10000 (both numbers can be exactly represented) gives 999999995904, i.e., the second operand is absorbed. The maximum error committed by representing a real number with a floating-point number under some rounding mode can be expressed in terms of the function $\mathrm{ulp}\colon \mathbb{R} \to \mathbb{R}$ (Muller 2005). Its value on 1.0 is about 10^{-7} for the single precision format.

2.2. Notation

The set of real numbers is denoted by $\mathbb{R}$ while $\mathbb{F}_{p,e_{\max}}$ denotes a subset of the binary floating-point numbers, defined from a given IEEE 754 format, which includes the infinities $-\infty$ and $+\infty$, the signed zeros $+0$ and -0, but neither subnormal numbers nor NaNs. Subnormals are introduced in the set $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}} = \mathbb{F}_{p,e_{\max}} \cup \bigl\{\, (-1)^s 0.m \times 2^{e_{\min}} \bigm| s \in \{0,1\}, m \neq 0 \,\bigr\}$. In some cases, the exposition can be much simplified by allowing the $e_{\max}$ of $\mathbb{F}_{p,e_{\max}}$ to be ∞, i.e., by considering an idealized set of floats where the exponent is unbounded. Among the advantages is the fact that subnormals in

[6] Examples of such properties are $\sqrt{1/z} = 1/\sqrt{z}$ and $1/(1/x) = x$ for $x = \pm\infty$.

$\mathtt{z} = \mathtt{x} \oplus \mathtt{y}$		$\mathtt{z} = \mathtt{x} \ominus \mathtt{y}$	
$\overline{\mathtt{z}} = \overline{\mathtt{x}} \oplus \overline{\mathtt{y}}$,	(direct)	$\overline{\mathtt{z}} = \overline{\mathtt{x}} \ominus \underline{\mathtt{y}}$,	(direct)
$\underline{\mathtt{z}} = \underline{\mathtt{x}} \oplus \underline{\mathtt{y}}$		$\underline{\mathtt{z}} = \underline{\mathtt{x}} \ominus \overline{\mathtt{y}}$	
$\overline{\mathtt{x}} = \mathrm{mid}(\overline{\mathtt{z}}, \overline{\mathtt{z}}^{+}) \ominus \underline{\mathtt{y}}$	(1^{st} indirect)	$\overline{\mathtt{x}} = \mathrm{mid}(\overline{\mathtt{z}}, \overline{\mathtt{z}}^{+}) \oplus \overline{\mathtt{y}}$	(1^{st} indirect)
$\underline{\mathtt{x}} = \mathrm{mid}(\underline{\mathtt{z}}, \underline{\mathtt{z}}^{-}) \ominus \overline{\mathtt{y}}$		$\underline{\mathtt{x}} = \mathrm{mid}(\underline{\mathtt{z}}, \underline{\mathtt{z}}^{-}) \oplus \underline{\mathtt{y}}$	
$\overline{\mathtt{y}} = \mathrm{mid}(\overline{\mathtt{z}}, \overline{\mathtt{z}}^{+}) \ominus \underline{\mathtt{x}}$	(2^{nd} indirect)	$\overline{\mathtt{y}} = \overline{\mathtt{x}} \ominus \mathrm{mid}(\underline{\mathtt{z}}, \underline{\mathtt{z}}^{-})$	(2^{nd} indirect)
$\underline{\mathtt{y}} = \mathrm{mid}(\underline{\mathtt{z}}, \underline{\mathtt{z}}^{-}) \ominus \overline{\mathtt{x}}$		$\underline{\mathtt{y}} = \underline{\mathtt{x}} \ominus \mathrm{mid}(\overline{\mathtt{z}}, \overline{\mathtt{z}}^{+})$	

Figure 1 **Formulas for direct/indirect projections of addition/subtraction**

$\mathbb{F}^{\text{sub}}_{p,e_{\max}}$ can be represented as normal floating-point numbers in $\mathbb{F}_{p,\infty}$. Given a set of floating-point numbers $\mathbb{F}$, $\mathbb{F}^{+}$ denotes the "non-negative" subset of $\mathbb{F}$, i.e., with $s = 0$.

For a finite, non-zero floating-point number x, we will write $\mathrm{even}(x)$ (resp., $\mathrm{odd}(x)$) to signify that the least significant digit of x's significand is 0 (resp., 1).

When the format is clear from the context, a real decimal constant (such as 10^{12}) denotes the corresponding roundTiesToEven floating-point value (i.e., 999999995904 for 10^{12}).

Henceforth, for $x \in \mathbb{R}$, x^{+} (resp., x^{-}) denotes the smallest (resp., greatest) floating-point number strictly greater (resp., smaller) than x with respect to the considered IEEE 754 format. Of course, we have $f_{\max}^{+} = +\infty$ and $(-f_{\max})^{-} = -\infty$.

Binary arithmetic operations over the floats will be denoted by $\oplus$, $\ominus$, $\otimes$ and $\oslash$, corresponding to $+$, $-$, $\cdot$ and $/$ over the reals, respectively. According to IEEE 754, they are defined, under roundTiesToEven, by

$$x \oplus y = [x + y]_{\mathrm{n}}, \qquad x \ominus y = [x - y]_{\mathrm{n}},$$
$$x \otimes y = [x \cdot y]_{\mathrm{n}}, \qquad x \oslash y = [x / y]_{\mathrm{n}}.$$

As IEEE 754 floating-point numbers are closed under negation, we denote the negation of $x \in \mathbb{F}^{\text{sub}}_{p,e_{\max}}$ simply by $-x$. Note that negation is a bijection. The symbol $\odot$ denotes any of $\oplus$, $\ominus$, $\otimes$ or $\oslash$. A floating-point variable $\mathtt{x}$ is associated with an interval of possible floating-point values; we will write $\mathtt{x} \in [\underline{\mathtt{x}}, \overline{\mathtt{x}}]$, where $\underline{\mathtt{x}}$ and $\overline{\mathtt{x}}$ denote the smallest and greatest value of the interval, $\underline{\mathtt{x}} \leq \overline{\mathtt{x}}$ and either $\underline{\mathtt{x}} \neq +0$ or $\overline{\mathtt{x}} \neq -0$. Note that $[+0, -0]$ is not an interval, whereas $[-0, +0]$ is the interval denoting the set of floating-point numbers $\{-0, +0\}$.

3. Background on Constraint Solving over Floating-Point Variables

In this section, we briefly recall the basic principles of interval-based consistency techniques over floating-point variables and constraints.

3.1. Interval-based Consistency on Arithmetic Constraints

Program analysis usually starts with the generation of an intermediate code representation in a form called *three-address code* (TAC). In this form, complex arithmetic expressions and assignments are decomposed into sequences of assignment instructions of the form

$$\texttt{result} := \texttt{operand}_1 \; \texttt{operator} \; \texttt{operand}_2.$$

A further refinement consists in the computation of the *static single assignment form* (SSA) whereby, labeling each assigned variable with a fresh name, assignments can be considered as if they were equality constraints. For example, the TAC form of the floating-point assignment $\texttt{z} := \texttt{z} * \texttt{z} + \texttt{z}$ is $\texttt{t} := \texttt{z} * \texttt{z}$; $\texttt{z} := \texttt{t} + \texttt{z}$, which in SSA form becomes $\texttt{t}_1 := \texttt{z}_1 * \texttt{z}_1$; $\texttt{z}_2 := \texttt{t}_1 + \texttt{z}_1$, which, in turn, can be regarded as the conjunction of the constraints $t_1 = z_1 \otimes z_1$ and $z_2 = t_1 \oplus z_1$.

In an interval-based consistency approach to constraint solving over the floats, constraints are used to iteratively narrow the intervals associated with each variable: this process is called *filtering*. A *projection* is a function that, given a constraint and the intervals associated with two of the variables occurring in it, computes a possibly refined interval[7] for the third variable (the projection is said to be *over* the third variable). Taking $z_2 = t_1 \oplus z_1$ as an example, the projection over z_2 is called *direct projection* (it goes in the same sense of the TAC assignment it comes from), while the projections over t_1 and z_1 are called *indirect projections*.[8] Note that, for constraint propagation, both direct and indirect projections are applied in order to refine the intervals for t_1, z_1 and z_2. In this paper we propose new filtering algorithms for improving indirect projections.

A projection is called *optimal* if the interval constraints it infers are as tight as possible, that is, if both bounds of the inferred intervals are attainable (and thus cannot be pruned).

Figure 1 gives non-optimal projections for addition and subtraction. For finite $x, y \in \mathbb{F}_{p,e_{\max}}$, $\operatorname{mid}(x, y)$ denotes the number that is exactly halfway between x and y; note that either $\operatorname{mid}(x, y) \in \mathbb{F}_{p,e_{\max}}$ or $\operatorname{mid}(x, y) \in \mathbb{F}_{p+1,e_{\max}}$. Non-optimal projections for multiplication and division can be found in (Botella et al. 2006, Michel 2002). Optimal projections are known for monotonic functions over one argument (Michel 2002), but they are generally not available for other functions. Note, however, that optimality is not required in an interval-based consistency approach to constraint solving, as filtering is just used to remove some, not necessarily all, inconsistent values.

[7] That is, tighter than the original interval.

[8] Note that direct and indirect projections are idempotent, as their inputs and outputs do not intersect. Consider $z = x \oplus y$: direct projection propagates information on x and y onto z, and doing it twice in a row would not enable any further inference. Likewise for indirect projections, which propagate information on z onto x and y.

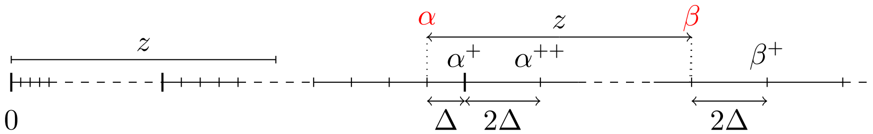

Figure 2 **An illustration of the Marre-Michel property: the segment z, if it has to represent the difference between two floats, cannot be moved past α**

3.2. The Marre-Michel Property

Marre and Michel (2010) published an idea to improve the filtering of the indirect projections for addition and subtraction. This is based on a property of the distribution of floating-point numbers among the reals: the greater a float, the greater the distance between it and its immediate successor. More precisely, for a given float x with exponent e_x, if $x^+ - x = \Delta$, then for y of exponent $e_x + 1$ we have $y^+ - y = 2\Delta$.

Proposition 1. (Marre and Michel 2010, Proposition 1) *Let $z \in \mathbb{F}_{p,\infty}$ be such that $0 < z < +\infty$; let also*

$$
\begin{aligned}
z &= 1.b_2 \cdots b_i \overbrace{0 \cdots 0}^{k} \times 2^{e_z}, && \textit{with } b_i = 1; \\
\alpha &= \overbrace{1.1 \cdots 1}^{p} \times 2^{e_z + k}, && \textit{with } k = p - i; \\
\beta &= \alpha \oplus z.
\end{aligned}
$$

Then, for each $x, y \in \mathbb{F}_{p,\infty}$, $z = x \ominus y$ implies that $x \le \beta$ and $y \le \alpha$. Moreover, $\beta \ominus \alpha = \beta - \alpha = z$.

This property, which can be generalized to subnormals, can intuitively be explained on Figure 2 as follows. Let $z \in \mathbb{F}_{p,\infty}$ be a strictly positive constant such that $z = x \ominus y$, where $x, y \in \mathbb{F}_{p,\infty}$ are unknown. The Marre-Michel property says that y cannot be greater than α. In fact, α is carefully positioned so that $\alpha^{++} - \alpha^+ = 2(\alpha^+ - \alpha)$, $e_\alpha + 1 = e_\beta$ and $z = \beta - \alpha$; if we take $y = \alpha^+$ we need $x > \beta$ if we want $z = x - y$; however, the smallest element of $\mathbb{F}_{p,\infty}$ that is greater than β, β^+, is 2Δ away from β, i.e., too much. Going further with y does not help: if we take $y \ge \alpha^+$, then $y - \alpha$ is an odd multiple of Δ (one Δ step from α to α^+, all the subsequent steps being even multiples of Δ), whereas for each $x \ge \beta$, $x - \beta$ is an even multiple of Δ. Hence, if $y > \alpha$, $\bigl|z - (x - y)\bigr| \ge \Delta = 2^{e_z + 1 - i}$. However, since $k \ne p - 1$, $z^+ - z = z - z^- = 2^{e_z + 1 - p} \le \Delta$. The last inequality, which holds because $p \ge i$, implies $z \ne x \ominus y$. A similar reasoning allows one to see that x cannot be greater than β independently from the value of y. In order to improve the filtering of the addition/subtraction projectors, Marre and Michel (2010) presented an algorithm to maximize the values of α and β over an interval. As we will see, that algorithm is not correct for some inputs. In Section 4.5, the main ideas behind the work presented in (Marre and Michel 2010) will be revisited, corrected and discussed.

4. Filtering by Maximum ULP

In this section we first informally present, by means of worked numerical examples, the techniques that are precisely defined later. We then reformulate the Marre-Michel property so as to generalize it to subnormals and to multiplication and division operators. The filtering algorithms that result from this generalization are collectively called *filtering by maximum ULP*.

4.1. Motivating Example

Consider the IEEE 754 single-precision constraint $\mathtt{z} = \mathtt{x} \oplus \mathtt{y}$ with initial intervals $\mathtt{z} \in [-\infty, +\infty]$, $\mathtt{x} \in [-1.0 \times 2^{50}, 1.0 \times 2^{50}]$ and $\mathtt{y} \in [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$. Forward projection gives

$$\mathtt{z} \in \bigl[-1.\overbrace{0 \cdots 0}^{19}1 \times 2^{50}, 1.\overbrace{0 \cdots 0}^{19}1 \times 2^{50}\bigr],$$

which is optimal, as both bounds are attainable. Suppose now the interval for $\mathtt{z}$ is further restricted to $\mathtt{z} \in [1.0, 2.0]$ due to, say, a constraint from an if-then-else in the program or another indirect projection.

With the classical indirect projection we obtain $\mathtt{x}, \mathtt{y} \in [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$, which, however, is not optimal. For example, pick $x = 1.0 \times 2^{30}$: for $y = -1.0 \times 2^{30}$ we have $x \oplus y = 0$ and $x \oplus y^{+} = 64$. By monotonicity of $\oplus$, for no $y \in [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$ we can have $x \oplus y \in [1.0, 2.0]$.

With our indirect projection, fully explained later, we obtain, from $\mathtt{z} \in [1.0, 2.0]$, the much tighter intervals $\mathtt{x}, \mathtt{y} \in [-1.1 \cdots 1 \times 2^{24}, 1.0 \times 2^{25}]$. These are actually optimal as $-1.1 \cdots 1 \times 2^{24} \oplus 1.0 \times 2^{25} = 1.0 \times 2^{25} \oplus -1.1 \cdots 1 \times 2^{24} = 2.0$. This example shows that filtering by maximum ULP can be stronger than classical interval-consistency based filtering. However, the opposite phenomenon is also possible. Consider again $\mathtt{z} = \mathtt{x} \oplus \mathtt{y}$ with $\mathtt{z} \in [1.0, 2.0]$. Suppose now the constraints for $\mathtt{x}$ and $\mathtt{y}$ are $\mathtt{x} \in [1.0, 5.0]$ and $\mathtt{y} \in [-f_{\max}, f_{\max}]$. As we have seen, our indirect projection gives $\mathtt{y} \in [-1.1 \cdots 1 \times 2^{24}, 1.0 \times 2^{25}]$; in contrast, the classical indirect projection exploits the available information on $\mathtt{x}$ to obtain $\mathtt{y} \in [-4, 1]$. Indeed, classical and maximum ULP filtering for addition and subtraction are orthogonal: both should be applied in order to obtain precise results.

For an example on multiplication, consider the IEEE 754 single-precision constraint $\mathtt{z} = \mathtt{x} \otimes \mathtt{y}$ with initial intervals $\mathtt{z} \in [1.0 \times 2^{-50}, 1.0 \times 2^{-30}]$ and $\mathtt{x}, \mathtt{y} \in [-\infty, +\infty]$. In this case, classical projections do not allow pruning the intervals. However, take $x = 1.1 \times 2^{119}$: for $y = 0$ we have $x \otimes y = 0$ and $x \otimes y^{+} = 1.1 \times 2^{-30}$. By monotonicity of $\otimes$, for no $y \in [-\infty, +\infty]$ we can have $x \otimes y \in [1.0 \times 2^{-50}, 1.0 \times 2^{-30}]$.

On the same example, $\mathtt{x}, \mathtt{y} \in [-1.0 \cdots 0 \times 2^{119}, 1.0 \cdots 0 \times 2^{119}]$ are the constraints inferred by our indirect projection. These are optimal because $1.0 \times 2^{-30} = -1.0 \cdots 0 \times 2^{119} \otimes -1.0 \cdots 0 \times 2^{149} = 1.0 \cdots 0 \times 2^{119} \otimes 1.0 \cdots 0 \times 2^{149}$. As is the case for addition, classical indirect projection can be more precise. Consider again $\mathtt{z} = \mathtt{x} \otimes \mathtt{y}$ with $\mathtt{z} \in [1.0 \times 2^{-50}, 1.0 \times 2^{-30}]$, $\mathtt{x} \in [2.0, 4.0]$ and $\mathtt{y} \in [-f_{\max}, f_{\max}]$. Classical indirect projection infers $\mathtt{y} \in [1.0 \times 2^{-52}, 1.0 \times 2^{-31}]$ by exploiting the information on $\mathtt{x}$.

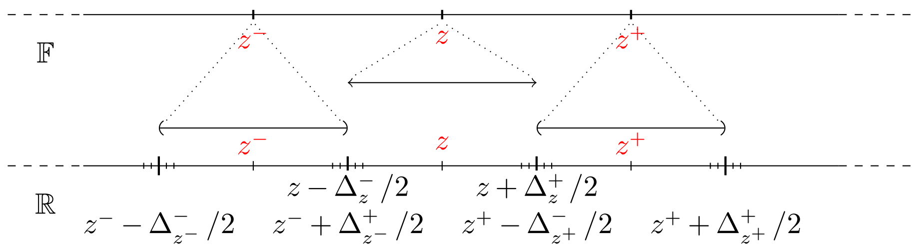

Figure 3 **Rounding of real numbers in the neighborhood of an even floating-point number z under roundTiesToEven**

4.2. Round-To-Nearest Tail-To-Even

We now formally define the roundTiesToEven rounding mode. To do that, we first introduce two functions: Δ_z^+ and Δ_z^- give the distance between z^+ and z and the distance between z and z^-.

Definition 1. The partial functions $\Delta^-\colon \mathbb{F}_{p,e_{\max}}^{\mathrm{sub}} \rightarrowtail \mathbb{R}$ and $\Delta^+\colon \mathbb{F}_{p,e_{\max}}^{\mathrm{sub}} \rightarrowtail \mathbb{R}$ are defined as follows, for each finite $z \in \mathbb{F}_{p,e_{\max}}^{\mathrm{sub}}$:

$$\Delta_z^+ = \begin{cases} 2^{1-p+e_{\max}}, & \text{if } z = f_{\max}; \\ f_{\min}, & \text{if } z = +0 \text{ or } z = -0; \\ z^+ - z, & \text{otherwise;} \end{cases}$$

$$\Delta_z^- = \begin{cases} 2^{1-p+e_{\max}} & \text{if } z = -f_{\max}; \\ f_{\min}, & \text{if } z = +0 \text{ or } z = -0; \\ z - z^-, & \text{otherwise.} \end{cases}$$

Note the special cases when $z = \pm 0$: since both $+0$ and -0 represent the real number 0, the distance between $z^+ = f_{\min}$ and $z = \pm 0$ is $f_{\min}$. We can now define the function $[\cdot]_{\mathrm{n}}$ that captures roundTiesToEven.

Definition 2. For $x \in \mathbb{R}$, $[x]_{\mathrm{n}}$ is defined as follows:

$$[x]_{\mathrm{n}} = \begin{cases} +0, & \text{if } 0 \le x \le \Delta_0^+/2; \\ -0, & \text{if } -\Delta_0^-/2 \le x < 0; \\ z, & \text{if } z \in \mathbb{F}_{p,e_{\max}}^{\mathrm{sub}} \setminus \{-\infty, +\infty\} \text{ and either } \mathrm{even}(z) \text{ and} \\ & z - \Delta_z^-/2 \le x \le z + \Delta_z^+/2, \text{ or } \mathrm{odd}(z) \text{ and} \\ & z - \Delta_z^-/2 < x < z + \Delta_z^+/2; \\ +\infty, & \text{if } x \ge f_{\max} + \Delta_{f_{\max}}^+/2; \\ -\infty, & \text{if } x \le -f_{\max} - \Delta_{-f_{\max}}^-/2. \end{cases}$$

Figure 3 illustrates the roundTiesToEven rounding mode; if z is even, each real number between $z - \Delta_z^-/2$ and $z + \Delta_z^+/2$, *including* extremes, is rounded to the same floating-point number z. As z is even, z^- is odd, and each real number between $z^- - \Delta_{z^-}^-/2$ and $z^- + \Delta_{z^-}^+/2$, *excluding* extremes, is rounded to z^-. Similarly for z^+. Note that point $z - \Delta_z^-/2$ coincides with $z^- + \Delta_{z^-}^+/2$ and $z + \Delta_z^+/2$ coincides with $z^+ - \Delta_{z^+}^-/2$.

All rounding modes are monotonic; in particular, for each $x, y \in \mathbb{R}$, $x \leq y$ implies $[x]_\mathrm{n} \leq [y]_\mathrm{n}$. Moreover, the *chop* and *near* rounding modes are *symmetric*, i.e., the value after rounding does not depend on the sign: for each $x \in \mathbb{R}$, $[x]_\mathrm{n} = -[-x]_\mathrm{n}$.

4.3. Upper Bound

It is worth pointing out that, while arithmetic operations on reals are strictly monotone, that is if $x + y = z$ then $x_1 + y > z$ for any $x_1 > x$, in floating-point arithmetic, operations are just monotone. If $x + y = z$ then we may still have $x_1 + y = z$ for some (or many) $x_1 > x$ since addition over the floats is absorbing. Therefore, for determining the greatest (or the smallest) x_1 satisfying $x_1 + y = z$ and correctly filter intervals over the floats, we need to introduce an appropriate, duly justified, framework.

For each IEEE 754 floating-point operation $\odot \in \{\oplus, \ominus, \otimes, \oslash\}$, in later sections we will define the sets $\mathbb{F}_\odot \subseteq \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ and $\bar{\mathbb{F}}_\odot \subseteq \mathbb{F}_{p,\infty}$. Then we will define functions $\bar{\delta}_\odot \colon \mathbb{F}_\odot \to \bar{\mathbb{F}}_\odot$ (see Definition 3 in Section 4.5 for $\oplus$ and, consequently, $\ominus$, Definition 5 in Section 4.6 for $\otimes$, and Definition 6 in Section 4.7 for $\oslash$) that satisfy the following property, for each $z \in \mathbb{F}_\odot \setminus \{-0, +0, -\infty\}$:

$$\bar{\delta}_\odot(z) = \max\{\, v \in \bar{\mathbb{F}}_\odot \mid \exists y \in \bar{\mathbb{F}}_\odot \,.\, v \odot y = z \,\}. \tag{1}$$

In words, $\bar{\delta}_\odot(z)$ is the greatest float in $\bar{\mathbb{F}}_\odot$ that can be the left operand of $\odot$ to obtain z.

Verifying that a function $\bar{\delta}_\odot$ satisfies (1) is equivalent to proving that it satisfies the following properties, for each $z \in \mathbb{F}_\odot \setminus \{-0, +0, -\infty\}$:

$$\bar{\delta}_\odot(z) \in \{\, v \in \bar{\mathbb{F}}_\odot \mid \exists y \in \bar{\mathbb{F}}_\odot \,.\, v \odot y = z \,\}; \tag{2}$$

$$\forall z' \in \bar{\mathbb{F}}_\odot : z' > \bar{\delta}_\odot(z) \implies z' \notin \{\, v \in \bar{\mathbb{F}}_\odot \mid \exists y \in \bar{\mathbb{F}}_\odot \,.\, v \odot y = z \,\}. \tag{3}$$

Property (3) means $\bar{\delta}_\odot(z)$ is a *correct* upper bound for the possible values of x, whereas (2) implies that $\bar{\delta}_\odot(z)$ is the *most precise* upper bound we could choose.

Note that we may have $\bar{\mathbb{F}}_\odot \not\subseteq \mathbb{F}_{p,e_{\max}}$: property (1) refers to an idealized set of floating-point numbers with unbounded exponents.

Since we are interested in finding the upper bound of $\bar{\delta}_\odot(z)$ for $z \in [\underline{z}, \overline{z}]$, we need the following

Proposition 2. *Let $w, v_1, \ldots, v_n \in \mathbb{F}_\odot \setminus \{-0, +0, -\infty\}$ be such that, for each $i = 1, \ldots, n$, $\bar{\delta}_\odot(w) \geq \bar{\delta}_\odot(v_i)$. Then, for each $w' \in \bar{\mathbb{F}}_\odot$ with $w' > \bar{\delta}_\odot(w)$ and each $z \in \mathbb{F}_\odot \setminus \{-0, +0, -\infty\}$, we have that $w' \notin \{\, v \in \bar{\mathbb{F}}_\odot \mid \exists y \in \bar{\mathbb{F}}_\odot \,.\, v \odot y = z \,\}$.*

Proof. Follows directly from (1).

Let $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$ be a floating-point constraint where $-0, +0, -\infty \notin [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ and let $w \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ be such that $\bar{\delta}_\odot(w) \geq \bar{\delta}_\odot(v)$ for each $v \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$: then no element of $\mathtt{x}$ that is greater than $\bar{\delta}_\odot(w)$ can participate to a solution of the constraint.

Dually, in order to refine the upper bound of $\mathtt{y}$ subject to $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$, it is possible to define a function $\bar{\delta}'_\odot$ satisfying the following property, for each $z \in \mathbb{F}_\odot \setminus \{-0, +0, -\infty\}$:

$$\bar{\delta}'_\odot(z) = \max\{\, v \in \bar{\mathbb{F}}_\odot \mid \exists x \in \bar{\mathbb{F}}_\odot \,.\, x \odot v = z \,\}. \tag{4}$$

Due to (4), a result analogous to the one of Proposition 2 holds for $\bar{\delta}'_\odot$, which allows refining the interval for $\mathtt{y}$. Note, though, that when $\odot$ is commutative (i.e., it is $\oplus$ or $\otimes$), $\bar{\delta}_\odot = \bar{\delta}'_\odot$.

4.4. Lower bound

For computing the lower bound, we will introduce functions $\underline{\delta}_\odot \colon \mathbb{F}_\odot \to \bar{\mathbb{F}}_\odot$ (defined in terms of the corresponding $\bar{\delta}_\odot$ functions in Section 4.5 for $\oplus$ and $\ominus$, in Section 4.6 for $\otimes$, and in Section 4.7 for $\oslash$) satisfying the following property, for each $z \in \mathbb{F}_\odot \setminus \{-0, +0, +\infty\}$:

$$\underline{\delta}_\odot(z) = \min\{\, v \in \bar{\mathbb{F}}_\odot \mid \exists y \in \bar{\mathbb{F}}_\odot \,.\, v \odot y = z \,\}. \tag{5}$$

This property entails a result similar to Proposition 2: given constraint $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$ where $-0, +0, +\infty \notin [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ and $w \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ such that $\underline{\delta}_\odot(w) \leq \underline{\delta}_\odot(v)$ for each $v \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$, the float $\underline{\delta}_\odot(w)$ is a possibly refined lower bound for $\mathtt{x}$.

In a dual way, in order to refine the lower bound of $\mathtt{y}$ subject to $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$, we will define functions $\underline{\delta}'_\odot$ satisfying, for each $z \in \mathbb{F}_\odot \setminus \{-0, +0, +\infty\}$:

$$\underline{\delta}'_\odot(z) = \min\{\, v \in \bar{\mathbb{F}}_\odot \mid \exists x \in \bar{\mathbb{F}}_\odot \,.\, x \odot v = z \,\}. \tag{6}$$

Property (6) ensures that, under $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$ where $-0, +0, +\infty \notin [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$, if $w \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ is such that $\underline{\delta}'_\odot(w) \leq \underline{\delta}'_\odot(v)$ for each $v \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$, then the float $\underline{\delta}'_\odot(w)$ is a possibly refined lower bound for $\mathtt{y}$.

Again, when $\odot$ is commutative $\underline{\delta}_\odot = \underline{\delta}'_\odot$.

4.5. Filtering by Maximum ULP on Addition/Subtraction

In this section we introduce the functions $\bar{\delta}_\oplus$, $\underline{\delta}_\oplus$, $\bar{\delta}_\ominus$, $\bar{\delta}'_\ominus$, $\underline{\delta}_\ominus$ and $\underline{\delta}'_\ominus$. Note that, since $\oplus$ is commutative, we have $\bar{\delta}'_\oplus = \bar{\delta}_\oplus$ and $\underline{\delta}'_\oplus = \underline{\delta}_\oplus$. Moreover, the function $\underline{\delta}_\oplus \colon \mathbb{F}_\oplus \to \bar{\mathbb{F}}_\oplus$ can be defined in terms of the function $\bar{\delta}_\oplus$ as follows: for each $z \in \mathbb{F}_\oplus \setminus \{-0, +0, +\infty\}$, $\underline{\delta}_\oplus(z) = -\,\bar{\delta}_\oplus(-z)$. We see that, if $\bar{\delta}_\oplus$ satisfies Property (1), then $\underline{\delta}_\oplus$ satisfies Property (5). Again, since $\oplus$ is commutative, $\underline{\delta}'_\oplus = \underline{\delta}_\oplus$.

The first step for defining $\bar{\delta}_\oplus$ consists in extending Proposition 1 in order to explicitly handle subnormal numbers. Such extension was already sketched by Marre and Michel (2010): here we fully describe it and prove its correctness. Subnormals, which in $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ are represented by numbers having the hidden bit $b_1 = 0$ and exponent $e_{\min}$, can be represented in $\mathbb{F}_{p,\infty}$ by numbers with $b_1 = 1$ and exponent strictly smaller than $e_{\min}$. Namely, the element of $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$

$$0.0\cdots01b_{j+1}\cdots b_p \times 2^{e_{\min}}$$

can be represented in $\mathbb{F}_{p,\infty}$ by the (normal) float

$$1.b_{j+1}\cdots b_p\overbrace{0\cdots0}^{j-1} \times 2^{e_{\min}-(j-1)}.$$

Based on this observation we can state the following

Proposition 3. *Let $z \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\min}}$ be such that $0 < z < f^{\mathrm{nor}}_{\min}$; define also*

$$\begin{aligned}
z &= 0.0\cdots01b_{j+1}\cdots b_i\overbrace{0\cdots0}^{k} \times 2^{e_{\min}}, && \textit{with } b_i = 1;\\
\alpha &= \overbrace{1.1\cdots1}^{p} \times 2^{e_{\min}+k}, && \textit{with } k = p - i;\\
\beta &= \alpha \oplus z.
\end{aligned}$$

Then, for each $x, y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$, $z = x \ominus y$ implies that $x \le \beta$ and $y \le \alpha$. Moreover, $\beta \ominus \alpha = \beta - \alpha = z$.

Proof. The subnormal z is represented in $\mathbb{F}_{p,\infty}$ by the normal float

$$\hat{z} = 1.b_{j+1}\cdots b_i\overbrace{0\cdots0}^{k}\overbrace{0\cdots0}^{j-1} \times 2^{e_{\min}-(j-1)} = 1.b_{j+1}\cdots b_i\overbrace{0\cdots0}^{k+j-1} \times 2^{e_{\min}-(j-1)}.$$

We can apply Proposition 1 to $\hat{z}$ and obtain $\alpha = 1.1\cdots1 \times 2^{e_{\min}-(j-1)+k+j-1} = 1.1\cdots1 \times 2^{e_{\min}+k}$. Moreover, Proposition 1 assures that

$$\beta = \alpha \oplus 1.b_{j+1}\cdots b_i\overbrace{0\cdots0}^{k+j-1} \times 2^{e_{\min}-(j-1)}$$

is such that, for each $x, y \in \mathbb{F}_{p,\infty}$, $z = x \ominus y$ implies $x \le \beta$ and $y \le \alpha$ and $\beta \ominus \alpha = \beta - \alpha = z$. Since each number in $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ has an equivalent representation in $\mathbb{F}_{p,\infty}$, we only need to prove that $\beta = \alpha \oplus z$, which holds, since

$$\begin{aligned}
\beta &= \alpha \oplus 1.b_{j+1}\cdots b_i\overbrace{0\cdots0}^{k+j-1} \times 2^{e_{\min}-(j-1)}\\
&= \alpha \oplus 0.0\cdots01b_{j+1}\cdots b_i\underbrace{0\cdots0}_{k} \times 2^{e_{\min}}\\
&= \alpha \oplus z.
\end{aligned}$$

□

Using Propositions 1 and 3, we formally define the function $\bar{\delta}_{\oplus}$ as follows.

DEFINITION 3. Let $\mathbb{F}_{\oplus} = \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$, $\bar{\mathbb{F}}_{\oplus} = \mathbb{F}^{+}_{p,\infty}$, and $z \in \mathbb{F}_{\oplus}$ be such that $|z| = b_1.b_2 \cdots b_i 0 \cdots 0 \times 2^{e_z}$, with $b_i = 1$. Similarly to Propositions 1 and 3, let $k = p - i$, $\alpha = 1.1 \cdots 1 \times 2^{e_z + k}$ and $\beta = \alpha \oplus |z|$. Then $\bar{\delta}_{\oplus} \colon \mathbb{F}_{\oplus} \to \bar{\mathbb{F}}_{\oplus}$ is defined, for each $z \in \mathbb{F}_{\oplus}$, by

$$\bar{\delta}_{\oplus}(z) = \begin{cases} +\infty, & \text{if } z = -\infty \text{ or } z = +\infty; \\ \alpha, & \text{if } -\infty < z < 0; \\ +0, & \text{if } z = -0 \text{ or } z = +0; \\ \beta, & \text{if } 0 < z < +\infty. \end{cases}$$

THEOREM 1. *Function $\bar{\delta}_{\oplus}$ is well-defined and satisfies* (2) *and* (3).

Proof. We first show that $\bar{\delta}_{\oplus}(z)$ is well-defined, i.e., that it is a total function from $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ to $\mathbb{F}^{+}_{p,\infty}$. To this aim note that α and β are always non-negative normal floating-point numbers belonging to $\mathbb{F}_{p,\infty}$, and that $\bar{\delta}_{\oplus}(z)$ is defined for each $z \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$. Secondly, let us consider the following cases:

$z = +\infty$: for each $y \neq -\infty$ we have $+\infty \oplus y = +\infty$; thus, as $\bar{\delta}_{\oplus}(z) = +\infty$, (2) holds and (3) vacuously holds.

$f^{\mathrm{nor}}_{\min} \leq z < +\infty$: we can apply Proposition 1 to obtain $z = \beta \ominus \alpha$. Then note that $\beta \ominus \alpha = [\beta - \alpha]_{\mathrm{n}} = [\beta + -\alpha]_{\mathrm{n}} = \beta \oplus -\alpha$. Hence, $\beta \oplus -\alpha = z$. Thus, $\bar{\delta}_{\oplus}(z) \oplus -\alpha = \beta \oplus -\alpha = z$ and (2) is satisfied with $y = -\alpha$. For proving (3), first note that $\beta > -\alpha$ since $\beta > 0$ and $\alpha > 0$. Moreover, by Proposition 1, we know that there does not exist an $x \in \mathbb{F}_{p,\infty}$ with $x > \beta$ such that there exists $y \in \mathbb{F}_{p,\infty}$ that satisfies $x \ominus y = z$. Since $x \ominus y = x \oplus -y$ we can conclude that, for each $z' > \beta = \bar{\delta}_{\oplus}(z)$, it does not exist $y' \in \mathbb{F}_{p,\infty}$ such that $z' \oplus y' = z$. Hence also (3) holds.

$0 < z < f^{\mathrm{nor}}_{\min}$: by applying Proposition 3 instead of Proposition 1 we can reason exactly as in the previous case.

$-\infty < z \leq -f^{\mathrm{nor}}_{\min}$: since $0 < -z < +\infty$ we can apply Proposition 1 to $-z$ and obtain $\beta \ominus \alpha = -z$ and thus $-(\beta \ominus \alpha) = z$. As $[\cdot]_{\mathrm{n}}$ is a symmetric rounding mode, we have $-(\beta \ominus \alpha) = -[\beta - \alpha]_{\mathrm{n}} = [\alpha - \beta]_{\mathrm{n}} = \alpha \oplus -\beta = z$. Thus, $\bar{\delta}_{\oplus}(z) \oplus -\beta = \alpha \oplus -\beta = z$ and (2) is satisfied with $y = -\beta$. For proving (3), first note that $\alpha > -\beta$ since $\alpha > 0$ and $\beta > 0$. Moreover, by Proposition 1, we know that there does not exist an $y \in \mathbb{F}_{p,\infty}$ with $y > \alpha$ such that there exists $x \in \mathbb{F}_{p,\infty}$ that satisfies $x \ominus y = -z$. Since $x \ominus y = -z$ is equivalent to $y \oplus -x = z$, we can conclude that, for each $z' > \alpha = \bar{\delta}_{\oplus}(z)$, it does not exist $y' \in \mathbb{F}_{p,\infty}$ such that $z' \oplus y' = z$. Therefore, also in this case, (3) holds.

$-f^{\mathrm{nor}}_{\min} < z < 0$: by applying Proposition 3 instead of Proposition 1 we can reason exactly as in the previous case. $\square$

As we have already observed, since $\oplus$ is commutative we have $\bar{\delta}'_{\oplus} = \bar{\delta}_{\oplus}$, that is, the same function $\bar{\delta}_{\oplus}$ is used to filter both x and y with respect to $z = x \oplus y$.

We now need algorithms to maximize $\bar{\delta}_{\oplus}$ and minimize $\underline{\delta}_{\oplus}$ over an interval of floating-point values. Since the two problems are dual to each other, we will focus on the maximization of $\bar{\delta}_{\oplus}$.

As $\bar{\delta}_\oplus$ is not monotonic, a nontrivial analysis of its range over an interval is required. When the interval contains only finite, nonzero and positive (resp., negative) values, the range of $\bar{\delta}_\oplus$ has a simple shape. We are thus brought to consider an interval $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ such that $\underline{\mathtt{z}} \notin \{-\infty, -0, +0\}$, $\overline{\mathtt{z}} \notin \{-0, +0, +\infty\}$, and $\underline{\mathtt{z}}$ and $\overline{\mathtt{z}}$ have the same sign. We will now revisit, correct and extend to subnormal floating-point numbers the algorithm originally proposed by Marre and Michel (2010) to maximize $\bar{\delta}_\oplus$ over $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$.

The idea presented in (Marre and Michel 2010) is the following. When dealing with an interval $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ with $\underline{\mathtt{z}} > 0$, α (and thus β and, therefore, our $\bar{\delta}_\oplus$) grows (i) with the exponent and (ii) with the number of successive 0 bits to the right of the significand, i.e., k in Propositions 1 and 3 and in Definition 3. Thus, maximizing these two criteria allows one to maximize α over the interval.

Definition 4. Let $\mathtt{z}$ be a variable over $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$. If we have $0 < \underline{\mathtt{z}} < \overline{\mathtt{z}} < +\infty$, then $\mu_\oplus(\mathtt{z}) \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ is given by:

1. $\mu_\oplus(\mathtt{z}) = 1.0 \cdots 0 \times 2^{e_{\overline{\mathtt{z}}}}$, if $e_{\underline{\mathtt{z}}} \neq e_{\overline{\mathtt{z}}}$;
2. $\mu_\oplus(\mathtt{z}) = b_1.b_2 \cdots b_{i-1} a 0 \cdots 0 \times 2^{e_{\overline{\mathtt{z}}}}$, if $e_{\underline{\mathtt{z}}} = e_{\overline{\mathtt{z}}}$, where, for some $b_i \neq b'_i$:

$$
\begin{aligned}
\underline{\mathtt{z}} &= b_1.b_2 \cdots b_{i-1} b_i \cdots \times 2^{e_{\overline{\mathtt{z}}}};\\
\overline{\mathtt{z}} &= b_1.b_2 \cdots b_{i-1} b'_i \cdots \times 2^{e_{\overline{\mathtt{z}}}};\\
a &= \begin{cases} 0, & \text{if } b_1.b_2 \cdots b_{i-1} 0 \cdots 0 \times 2^{e_{\overline{\mathtt{z}}}} = \underline{\mathtt{z}};\\ 1, & \text{otherwise.} \end{cases}
\end{aligned}
$$

If $0 < \underline{\mathtt{z}} = \overline{\mathtt{z}} < +\infty$, then $\mu_\oplus(\mathtt{z}) = \underline{\mathtt{z}}$. If $-\infty < \underline{\mathtt{z}} \leq \overline{\mathtt{z}} < 0$, then $\mu_\oplus(\mathtt{z}) \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ is simply defined by $\mu_\oplus(\mathtt{z}) = -\mu_\oplus(\mathtt{w})$ where $\mathtt{w} \in [-\overline{\mathtt{z}}, -\underline{\mathtt{z}}]$. We leave $\mu_\oplus(\mathtt{z})$ undefined otherwise.
Note that Definition 4 cannot be usefully extended to intervals containing zeros or infinities, as no interesting bounds can be derived for $\mathtt{x}$ and $\mathtt{y}$ in such cases. Consider, for example, the constraint $\mathtt{x} \oplus \mathtt{y} = \mathtt{z}$ with $\mathtt{z} = +0$: for each $x \in [-f_{\max}, +f_{\max}]$ we have $x \oplus -x = +0$. Hence, when the interval of $\mathtt{z}$ contains zeros or infinities, only the classical filtering (Botella et al. 2006, Michel 2002) is applied.

Theorem 2. *Let* $\mathtt{z}$ *be over* $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ *with* $\underline{\mathtt{z}} \notin \{-\infty, -0, +0\}$ *and* $\overline{\mathtt{z}} \notin \{-0, +0, +\infty\}$ *having the same sign. Then, for each* $z \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$, $\bar{\delta}_\oplus(z) \leq \bar{\delta}_\oplus\bigl(\mu_\oplus(\mathtt{z})\bigr)$.

Proof. Without loss of generality, assume $\underline{\mathtt{z}} > 0$. If $\underline{\mathtt{z}} = \overline{\mathtt{z}}$ the result holds. Let us now assume $\underline{\mathtt{z}} < \overline{\mathtt{z}}$. We start proving that α and β of Definition 3 computed over $\mu_\oplus(\mathtt{z})$ are greater than or equal to the α's and β's computed over any other value in $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$.

We first prove that $\mu_\oplus(\mathtt{z})$ maximizes α. For $z \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ we have

$$\alpha = 1.1 \cdots 1 \times 2^{e_z + k},$$

where k is the number of successive 0's to the right of the significand of z. Let us consider the maximum exponent of the values in z, which is $e_{\overline{\mathsf{z}}}$. Among the values in $[\underline{\mathsf{z}}, \overline{\mathsf{z}}]$ with such an exponent, we want to select the one with the highest number of successive zeros to the right of the significand. Since $\underline{\mathsf{z}} > 0$, the maximum value for α would be attained by the float $1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$, if this belongs to $[\underline{\mathsf{z}}, \overline{\mathsf{z}}]$. This happens in three cases:

1. $e_{\underline{\mathsf{z}}} \neq e_{\overline{\mathsf{z}}}$ and $\mu_{\oplus}(\mathsf{z}) = 1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$, by the first case of Definition 4.

2. $e_{\underline{\mathsf{z}}} = e_{\overline{\mathsf{z}}}$ and $\underline{\mathsf{z}} = 1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$; in this case we have, again, $\mu_{\oplus}(\mathsf{z}) = 1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$, so defined by the second case of Definition 4; in fact, for some $i \in \{2, \ldots, p-1\}$ that depends on $\overline{\mathsf{z}}$, we have

$$\overline{\mathsf{z}} = 1.b_2 \cdots b_{i-1} 10 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}},$$
$$\underline{\mathsf{z}} = 1.b_2 \cdots b_{i-1} 00 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$$

with $b_2 = \cdots = b_{i-1} = 0$, and the algorithm gives $1.b_2 \cdots b_{i-1} a 0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$ with $a = 0$, i.e., $1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$.

3. $e_{\underline{\mathsf{z}}} = e_{\overline{\mathsf{z}}}$, $\underline{\mathsf{z}} = 0.b_2 \cdots b_p \times 2^{e_{\min}}$ and $\overline{\mathsf{z}} = 1.b'_2 \cdots b'_p \times 2^{e_{\min}}$; thus we have, $\mu_{\oplus}(\mathsf{z}) = 1.0 \cdots 0 \times 2^{e_{\min}}$, once again by the second case of Definition 4 where $i = 1$, hence $\mu_{\oplus}(\mathsf{z}) = a.0 \cdots 0 \times 2^{e_{\min}}$. Moreover, since $\underline{\mathsf{z}} > 0$, necessarily $\underline{\mathsf{z}} \neq 0.0 \cdots 0 \times 2^{e_{\min}}$ and we must have $a = 1$.

We are now left with the case when $1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}} \notin [\underline{\mathsf{z}}, \overline{\mathsf{z}}]$. This occurs when $e_{\underline{\mathsf{z}}} = e_{\overline{\mathsf{z}}}$ but either $\underline{\mathsf{z}} > 1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$ or $\overline{\mathsf{z}} < 1.0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$. In both cases, all the floats in $[\underline{\mathsf{z}}, \overline{\mathsf{z}}]$ have the same exponent and the same most significant bit (b_1). Therefore, in order to maximize α, we need to choose among them the one with the greatest number of successive zeros to the right of the significand. The first step is to find the index of the most significant significand bit where $\underline{\mathsf{z}}$ and $\overline{\mathsf{z}}$ differ: since $\underline{\mathsf{z}} < \overline{\mathsf{z}}$, such an index must exist. Let then

$$\underline{\mathsf{z}} = b_1.b_2 \cdots b_{i-1} b_i \cdots \times 2^{e_{\overline{\mathsf{z}}}},$$
$$\overline{\mathsf{z}} = b_1.b_2 \cdots b_{i-1} b'_i \cdots \times 2^{e_{\overline{\mathsf{z}}}},$$

where $b_i = 0$ and $b'_i = 1$ for some $i > 1$. The significand maximizing α is $b_1.b_2 \cdots b_{i-1} 0 \cdots 0$. Indeed, any float having a significand with a larger number of consecutive zeros to the right does not belong to $[\underline{\mathsf{z}}, \overline{\mathsf{z}}]$. However, it is not always the case that $b_1.b_2 \cdots b_{i-1} 0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}$ belongs to $[\underline{\mathsf{z}}, \overline{\mathsf{z}}]$: we must have

$$\underline{\mathsf{z}} = b_1.b_2 \cdots b_{i-1} b_i 0 \cdots 0 \times 2^{e_{\overline{\mathsf{z}}}}. \tag{7}$$

If (7) is true, then the second case of Definition 4 gives

$$\mu_{\oplus}(\mathsf{z}) = b_1.b_2 \cdots b_{i-1} a 0 \cdots \times 2^{e_{\overline{\mathsf{z}}}}, \qquad \text{with } a = 0,$$

which is indeed equal to $\underline{\mathsf{z}}$. On the other hand, if (7) is false, then no float with significand $b_1.b_2\cdots b_{i-1}00\cdots 0$ belongs to $[\underline{\mathsf{z}},\overline{\mathsf{z}}]$, hence the significand maximizing α is necessarily the one with one less zero to the right, i.e., $b_1.b_2\cdots b_{i-1}10\cdots 0$, which is guaranteed to belong to $[\underline{\mathsf{z}},\overline{\mathsf{z}}]$. This is consistent with the second case of Definition 4, which gives

$$\mu_\oplus(\mathsf{z}) = b_1.b_2\cdots b_{i-1}a0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}}, \qquad \text{with } a=1.$$

We have proved that Definition 4 gives a float $\mu_\oplus(\mathsf{z})$ that maximizes the value α. We now prove that $\mu_\oplus(\mathsf{z})$ also maximizes the value of β. By Propositions 1 and 3 and Definition 3, $\beta=\alpha\oplus z$. Note that $\mu_\oplus(\mathsf{z})$ maximizes α; however, since β also depends on z, we have to prove that no $z\in[\underline{\mathsf{z}},\overline{\mathsf{z}}]$ such that $z>\mu_\oplus(\mathsf{z})$ results into a greater β. Observe first that, by construction, $\mu_\oplus(\mathsf{z})$ has the maximum exponent in $[\underline{\mathsf{z}},\overline{\mathsf{z}}]$. Therefore any $z>\mu_\oplus(\mathsf{z})$ in $[\underline{\mathsf{z}},\overline{\mathsf{z}}]$ must have a larger significand. Assume that $\mu_\oplus(\mathsf{z}) = b_1.b_2\cdots b_j0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}}$ with $b_j=1$ for some $j\in\{1,\ldots,p\}$. The exponent of the corresponding α is $e_{\overline{\mathsf{z}}}+p-j$. Suppose now there exists $z>\mu_\oplus(\mathsf{z})$ in $[\underline{\mathsf{z}},\overline{\mathsf{z}}]$ with a larger significand: this must have the form $b_1.b_2\cdots b_\ell 0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}}$ with $b_\ell=1$ and $j<\ell\le p$. The exponent of the corresponding α is $e_{\overline{\mathsf{z}}}+p-\ell$, which is smaller than the α computed for $\mu_\oplus(\mathsf{z})$ by at least one unit. Hence, we can conclude that $b_1.b_2\cdots b_j0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}} + 1.1\cdots 1\times 2^{e_{\overline{\mathsf{z}}}+p-j} > b_1.b_2\cdots b_l0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}} + 1.1\cdots 1\times 2^{e_{\overline{\mathsf{z}}}+p-\ell}$, since $\ell>j$. This shows that the float $\mu_\oplus(\mathsf{z})$ also maximizes the value of β. We have proved that Definition 4 gives a float $\mu_\oplus(\mathsf{z})$ that maximizes the value of both α and β over z. Since Definition 3 defines $\bar\delta_\oplus(z)=\alpha$ for $-\infty<z<0$ and $\bar\delta_\oplus(z)=\beta$ for $0<z<+\infty$, we can conclude that, for each $z\in[\underline{\mathsf{z}},\overline{\mathsf{z}}]$, $\bar\delta_\oplus(z)\le\bar\delta_\oplus(\mu_\oplus(\mathsf{z}))$. $\square$

As we have already pointed out, the algorithm of Definition 4, if restricted to normal numbers, is similar to the algorithm presented in (Marre and Michel 2010). There is an important difference, though, in the case when $\underline{\mathsf{z}}=b_1.b_2\cdots b_{i-1}b_i0\cdots 0\times 2^{e_{\overline{\mathsf{z}}}}$, $\overline{\mathsf{z}}=b_1.b_2\cdots b_{i-1}b_i'\cdots\times 2^{e_{\overline{\mathsf{z}}}}$ and $\underline{\mathsf{z}}>0$. In this case the algorithm of Marre and Michel (2010) erroneously returns $b_1.b_2\cdots b_{i-1}10\cdots 0\times 2^{e_{\overline{\mathsf{z}}}}$ instead of the value that maximizes α, i.e., $\underline{\mathsf{z}}$, which is correctly computed by our algorithm.

For efficiency reasons, filtering by maximum ULP might be applied only when $\bar\delta_\oplus\big(\mu_\oplus(\mathsf{z})\big)\le f_{\max}$ so as to avoid the use of wider floating-point formats.

In order to define $\bar\delta_\ominus$, $\bar\delta'_\ominus$, $\underline{\delta}_\ominus$ and $\underline{\delta}'_\ominus$, we can use the following observation. Since $x\ominus y=[x-y]_{\mathrm{n}}=[x+-y]_{\mathrm{n}}=x\oplus -y$, the constraints $\mathsf{z}=\mathsf{x}\ominus\mathsf{y}$ and $\mathsf{z}=\mathsf{x}\oplus-\mathsf{y}$ are equivalent. Thus we have $\bar\delta_\ominus=\bar\delta_\oplus$ and $\underline{\delta}_\ominus=\underline{\delta}_\oplus$, while $\bar\delta'_\ominus=-\underline{\delta}_\oplus$ and $\underline{\delta}'_\ominus=-\bar\delta_\oplus$ since, if $-y\in[\underline{\delta}_\oplus(z),\bar\delta_\oplus(z)]$, then $y\in[-\bar\delta_\oplus(z),-\underline{\delta}_\oplus(z)]$. Moreover, since $\mu_\oplus(\mathsf{z})$ maximizes $\bar\delta_\oplus$ and minimizes $\underline{\delta}_\oplus$ over an interval of floating-point values z, $\mu_\oplus(\mathsf{z})$ can be used as well to maximize $\bar\delta'_\ominus$ and minimize $\underline{\delta}'_\ominus$ on z.

4.6. Filtering by Maximum ULP on Multiplication

For filtering multiplication constraints of the form $z = x \otimes y$ (and similarly for division), we cannot rely on the same maximum ULP property identified by Marre and Michel upon which the treatment of addition and subtraction rests. This is because the ULP property of z is only loosely related to the ULP property of x and y when they are being multiplied. Our generalized property, instead, covers also multiplication (and division, as we will see in Section 4.7). As indicated in (1) and (5), we have to the determine the maximum and minimum values for x satisfying $z = x \otimes y$.

Consider a strictly positive constant $z \in \mathbb{F}_{p,e_{\max}}$ and two unknowns $x, y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ such that $z = x \otimes y$. If $z \leq f_{\max}/f_{\min}$, there exists a greatest float $x_{\mathrm{m}} \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ such that there exists $y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ satisfying $z = x_{\mathrm{m}} \otimes y$. More precisely, x_{m} must satisfy $z = x_{\mathrm{m}} \otimes f_{\min}$ and it turns out that we can take $x_{\mathrm{m}} = z \oslash f_{\min}$. Since, for $z \leq f_{\max}/f_{\min}$, division of z by $f_{\min} = 2^{e_{\min}+1-p}$ amounts to an exponent shifting, we have that $\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}} \ni x_{\mathrm{m}} = z/f_{\min}$. Moreover, we have that $x_{\mathrm{m}} = z/f_{\min}$ is the greatest float such that $z = x_{\mathrm{m}} \otimes f_{\min}$.[9]

On the other hand, there is no other float $y < f_{\min}$ such that $z = x \otimes y$, since y must be greater than $+0$, for otherwise $x \otimes y$ would not be strictly positive. However, for no $y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ we have $+0 < y < f_{\min}$. Therefore, the greatest value x_{m} such that $z = x_{\mathrm{m}} \otimes f_{\min}$ is the greatest value for x that can satisfy $z = x \otimes y$ for some $y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$.

When dealing with subnormal floating-point numbers a similar argument applies. In fact, also in this case there exists a greatest float $x_{\mathrm{m}} \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ satisfying $z = x_{\mathrm{m}} \otimes y$ for some $y \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$. As before, such x_{m} must satisfy $z = x_{\mathrm{m}} \otimes f_{\min}$. However, it turns out that, when z is subnormal, there may exist values for x_{m} greater than $z/f_{\min}$ that still satisfy $z = x_{\mathrm{m}} \otimes f_{\min}$. This is because the distance between subnormal numbers, being fixed to $f_{\min}$, does not depend on z.

Based on the previous reasoning, we can define $\bar{\delta}_{\otimes}$ and $\underline{\delta}_{\otimes}$.

Definition 5. Let $\mathbb{F}_{\otimes} = \left\{ z \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}} \;\middle|\; |z|/f_{\min} \leq f_{\max} \right\}$ and $\bar{\mathbb{F}}_{\otimes} = \mathbb{F}_{p,e_{\max}}$. Then $\bar{\delta}_{\otimes} \colon \mathbb{F}_{\otimes} \to \bar{\mathbb{F}}_{\otimes}$ is defined, for each $z \in \mathbb{F}_{\otimes}$, by

$$\bar{\delta}_{\otimes}(z) = \begin{cases} |z| \oslash f_{\min}, & \text{if } |z| \geq f^{\mathrm{nor}}_{\min}; \\ \big(|z| \oslash f_{\min}\big) \oplus 2^{-1}, & \text{if } 0 < |z| < f^{\mathrm{nor}}_{\min} \text{ and } \mathrm{even}(z); \\ \Big(\big(|z| \oslash f_{\min}\big) \oplus 2^{-1}\Big)^{-}, & \text{if } 0 < |z| < f^{\mathrm{nor}}_{\min} \text{ and } \mathrm{odd}(z). \end{cases}$$

Theorem 3. *Function $\bar{\delta}_{\otimes}$ is well-defined and satisfies* (2) *and* (3).

Proof. Given in the Online Supplement (Bagnara et al. 2015).

A monotonicity property of $\bar{\delta}_{\otimes}$ simplifies the identification an element of the interval $\mathbf{z}$ that maximizes the value of $\bar{\delta}_{\otimes}$ over $\mathbf{z}$.

[9] See the proof of forthcoming Theorem 3 available in the Online Supplement (Bagnara et al. 2015).

PROPOSITION 4. *Let $z \in \mathbb{F}_\otimes$ be nonzero. If $z > 0$, then $\bar{\delta}_\otimes(z^+) \geq \bar{\delta}_\otimes(z)$; on the other hand, if $z < 0$, then $\bar{\delta}_\otimes(z^-) \geq \bar{\delta}_\otimes(z)$.*

Proof. Given in the Online Supplement (Bagnara et al. 2015).

Since $\otimes$ is commutative, $\bar{\delta}'_\otimes = \bar{\delta}_\otimes$, and the same bounds can be used to filter both x and y in the constraint $z = x \otimes y$.

The function $\underline{\delta}_\otimes : \mathbb{F}_\otimes \to \bar{\mathbb{F}}_\otimes$ is defined dually: for each $z \in \mathbb{F}_\otimes \setminus \{-0, +0\}$, $\underline{\delta}_\otimes(z) = -\bar{\delta}_\otimes(z)$. We can see that properties (2) and (3) of $\bar{\delta}_\otimes$ entail property (5) of $\underline{\delta}_\otimes$. Again, since $\otimes$ is commutative we have $\underline{\delta}'_\otimes = \underline{\delta}_\otimes$.

Thanks to Proposition 4 we know that the value $M \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ that maximizes $\bar{\delta}_\otimes$ is the one with the greatest absolute value, i.e., $M = \max\bigl\{|\underline{\mathtt{z}}|, |\overline{\mathtt{z}}|\bigr\}$. Since $\underline{\delta}_\otimes$ is defined as $-\bar{\delta}_\otimes(z)$, the value that minimizes $\underline{\delta}_\otimes$ is again M. Hence, if $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ does not contain zeros, $\bar{\delta}_\otimes(M)$ (resp., $\underline{\delta}_\otimes(M)$) is an upper bound (resp., a lower bound) of $\mathtt{x}$ with respect to the constraint $\mathtt{z} = \mathtt{x} \otimes \mathtt{y}$.

The restriction to intervals $\mathtt{z}$ not containing zeros is justified by the fact that, e.g., if $z = 0$ then $z = x \otimes y$ holds with $x = f_{\max}$ and $y = 0$, hence, in this case, no useful filtering can be applied to x. The same thing happens when $\max\bigl\{|\underline{\mathtt{z}}|, |\overline{\mathtt{z}}|\bigr\}/f_{\min} > f_{\max}$. Moreover, whenever the interval of $\mathtt{y}$ does not contain zeros, filtering by maximum ULP for multiplication, in order to refine $\mathtt{x}$, is subsumed by the standard indirect projection, which, in this case, can usefully exploit the information on $\mathtt{y}$. In contrast, when the interval of $\mathtt{y}$ does contain zeros, our filter is able to derive bounds that cannot be obtained with the standard indirect projection, which, in this case, does not allow any refinement of the interval. Thus, for multiplication (and, as we will see, for division as well), the standard indirect projection and filtering by maximum ULP are mutually exclusive: one applies when the other cannot derive anything useful. Commenting on a previous version of the present paper, Claude Michel observed that one could modify the standard indirect projections with interval splitting so that indirect projections are always applied to source intervals not containing zeros. This idea rests on the observation that, for $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$ with $\odot \in \{\otimes, \oslash\}$, when the interval of $\mathtt{z}$ is a subset of the finite non zero floats neither $\mathtt{x}$ nor $\mathtt{y}$ do have any support for ± 0 and $\pm\infty$. For multiplication, ordinary standard indirect projection would be modified as follows, assuming that $\mathtt{z}$ is positive and that we want to apply the standard indirect projection to $\mathtt{z}$ and $\mathtt{y}$ in order to refine $\mathtt{x}$ (the other cases being similar):

- we apply the ordinary standard indirect projection to $\mathtt{z}$ and $\mathtt{y} \cap [-f_{\max}, -f_{\min}]$, intersecting the resulting interval with $[-f_{\max}, -f_{\min}]$;
- we apply the ordinary standard indirect projection to $\mathtt{z}$ and $\mathtt{y} \cap [f_{\min}, f_{\max}]$, intersecting the resulting interval with $[f_{\min}, f_{\max}]$;
- finally, we use the convex union of the two intervals so computed to refine $\mathtt{x}$.

We believe that, when the applied ordinary (i.e., non-splitting) standard indirect projection is as precise as the one specified by Michel (2002), the refining interval computed for $\mathtt{x}$ by the modified procedure is very precise: it coincides with the result of the ordinary standard indirect projection, when $0 \notin \mathtt{y}$ and thus filtering by maximum ULP is not applicable, or it coincides with the result of filtering by maximum ULP, when $0 \in \mathtt{y}$ and therefore the ordinary standard indirect projection would not help.[10] This approach has the advantage to be applicable to any rounding mode. On the other hand the standard indirect projections specified in (Michel 2002) require working on rationals or on larger floating-point formats, whereas one of our aims is to always work with machine floating-point numbers of the same size of those used in the analyzed computation.

EXAMPLE 2. Consider the IEEE 754 single-precision constraint $\mathtt{z} = \mathtt{x} \otimes \mathtt{y}$ with $\mathtt{z}$ subnormal, $\mathtt{z} \in [-0.00000000000000010001001 \times 2^{-126}, -0.00000000000010000000000 \times 2^{-126}]$, and $\mathtt{x}$ and $\mathtt{y}$ unconstrained, $\mathtt{x}, \mathtt{y} \in [-\infty, +\infty]$. Our indirect projection infers the constraints $\mathtt{x}, \mathtt{y} \in [-1.00000000001 \times 2^{10}, 1.00000000001 \times 2^{10}]$, while classical inverse projections do not allow pruning the intervals for x and y, no matter what they are.

4.7. Filtering by Maximum ULP on Division

We now define filtering by maximum ULP for floating-point constraints of the form $\mathtt{z} = \mathtt{x} \oslash \mathtt{y}$. We begin defining the first indirect projection. We will then tackle the problem of defining the second indirect projection, which, as we will see, is significantly more involved than the first one: the solution we propose is new to this paper.

4.7.1. The First Indirect Projection A role similar to the one of $f_{\min}$ in the definition of filtering by maximum ULP on multiplication is played by $f_{\max}$ in the definition of the first indirect projection for division.

DEFINITION 6. Let us define the sets $\mathbb{F}'_{\oslash} = \bigl\{ z \in \mathbb{F}^{\mathrm{sub}}_{p,e_{\max}} \bigm| |z| \otimes f_{\max} \leq f_{\max} \bigr\}$ and $\bar{\mathbb{F}}'_{\oslash} = \mathbb{F}_{p,e_{\max}}$. Let also $q = 1 - p + e_{\min} + e_{\max}$.[11] Then $\bar{\delta}_{\oslash} \colon \mathbb{F}'_{\oslash} \to \bar{\mathbb{F}}'_{\oslash}$ is defined, for each $z \in \mathbb{F}'_{\oslash}$, by

$$\bar{\delta}_{\oslash}(z) = \begin{cases} |z| \otimes f_{\max}, & \text{if } f^{\mathrm{nor}}_{\min} \leq |z| \leq 1; \\ \bigl(|z| \otimes f_{\max}\bigr) \oplus 2^{q}, & \text{if } 0 \leq |z| < f^{\mathrm{nor}}_{\min} \\ & \quad \wedge \bigl(|z| \neq 1 \times 2^{e_z} \vee e_z = e_{\min} - 1\bigr); \\ \Bigl(\bigl(|z| \otimes f_{\max}\bigr) \oplus 2^{q}\Bigr)^{-}, & \text{otherwise.} \end{cases}$$

Observe that we have $|z| \otimes f_{\max} \leq f_{\max}$ if and only if $|z| \leq 1$. In fact, for $z = 1^{+} = 1 + 2^{1-p}$, we obtain

$$|z| \otimes f_{\max} = (1 + 2^{1-p}) \otimes f_{\max}$$

[10] We are indebted to Claude Michel for this observation.

[11] In the very common case where $e_{\min} = 1 - e_{\max}$ we have $q = 2 - p$.

$$\begin{aligned}
&= \big[(1+2^{1-p})f_{\max}\big]_{\mathrm{n}} \\
&= \big[f_{\max} + (2-2^{1-p})2^{e_{\max}+1-p}\big]_{\mathrm{n}} \\
&= +\infty, \qquad (8)
\end{aligned}$$

where (8) holds by Definition 2, since $(2-2^{1-p})2^{e_{\max}+1-p} > \Delta^{+}_{f_{\max}}/2 = 2^{e_{\max}-p}$. By monotonicity of $\otimes$ we can conclude that $z \in \mathbb{F}'_{\oslash}$ if and only if $|z| \leq 1$.

THEOREM 4. *$\bar{\delta}_{\oslash}$ is well-defined and satisfies* (2) *and* (3)*.*

Proof. Given in the Online Supplement (Bagnara et al. 2015).

The function $\underline{\delta}_{\oslash}$ is defined, for each $z \in \mathbb{F}'_{\oslash}$, by $\underline{\delta}_{\oslash} = -\bar{\delta}_{\oslash}(z)$.

As for multiplication, a monotonicity property of $\bar{\delta}_{\oslash}$ enables quickly identifying the value of $\mathtt{z}$ that maximizes the function.

PROPOSITION 5. *Let $z \in \mathbb{F}_{\oslash}$ be nonzero. If $z > 0$, then $\bar{\delta}_{\oslash}(z^{+}) \geq \bar{\delta}_{\oslash}(z)$; on the other hand, if $z < 0$, then $\bar{\delta}_{\oslash}(z^{-}) \geq \bar{\delta}_{\oslash}(z)$.*

Proof. Given in the Online Supplement (Bagnara et al. 2015).

By monotonicity, the value $M \in [\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ that maximizes $\bar{\delta}_{\oslash}$ is the one that has the greatest absolute value, i.e., $M = \max\{|\underline{\mathtt{z}}|, |\overline{\mathtt{z}}|\}$. Since $\underline{\delta}_{\oslash}$ is defined as $-\bar{\delta}_{\oslash}(z)$, M is also the value that minimizes $\underline{\delta}_{\oslash}$. Hence, if $[\underline{\mathtt{z}}, \overline{\mathtt{z}}]$ does not contain zeros, $\bar{\delta}_{\oslash}(M)$ (resp., $\underline{\delta}_{\oslash}(M)$) is an upper bound (resp. a lower bound) of $\mathtt{x}$ with respect to the constraint $\mathtt{z} = \mathtt{x} \oslash \mathtt{y}$. The restriction to intervals not containing zeros is justified by the fact that, e.g., if $z = 0$ then $z = x \oslash y$ holds with $x = f_{\max}$ and $y = \infty$; hence, in this case, no useful filtering can be applied to x. The same happens when $\max\{|\underline{\mathtt{z}}|, |\overline{\mathtt{z}}|\} \otimes f_{\max} > f_{\max}$. In addition, the same phenomenon we saw for multiplication manifests itself here: whenever the interval of the variable $\mathtt{y}$ does not contain infinities, filtering by maximum ULP for division in order to refine $\mathtt{x}$ is subsumed by the standard indirect projection. On the other hand, when the interval of $\mathtt{y}$ does contain infinities, the standard indirect projection gives nothing whereas filtering by maximum ULP provides nontrivial bounds. Thus, the standard indirect projection and filtering by maximum ULP for division are mutually exclusive: one applies when the other cannot derive anything useful. And, just as for multiplication, if using rationals or extended floating-point formats is an option, then a pruning variant (one that cuts off infinities) of the indirect projection specified in (Michel 2002) will be equally precise.

EXAMPLE 3. Consider the IEEE 754 single-precision constraint $\mathtt{z} = \mathtt{x} \oslash \mathtt{y}$ with initial intervals $\mathtt{z} \in [-1.0 \times 2^{-110}, -1.0 \times 2^{-121}]$ and $\mathtt{x}, \mathtt{y} \in [-\infty, +\infty]$. We have

$$\begin{aligned}
\bar{\delta}_{\oslash}(1.0 \times 2^{-110}) &= 1.0 \times 2^{-110} \cdot 1.1 \cdots 1 \times 2^{127} \\
&= 1.1 \cdots 1 \times 2^{17},
\end{aligned}$$

$$\begin{aligned}\underline{\delta}_{\oslash}(1.0\times 2^{-110}) &= -1.0\times 2^{-110}\cdot 1.1\cdots 1\times 2^{127}\\ &= -1.1\cdots 1\times 2^{17}.\end{aligned}$$

Filtering by maximum ULP improves upon classical filtering, which would not restrict any interval, with $\mathtt{x}\in[-1.1\ldots 1\times 2^{17}, 1.1\ldots 1\times 2^{17}]$.

For an example involving subnormals, consider $\mathtt{z}=\mathtt{x}\oslash\mathtt{y}$ with initial interval for $\mathtt{z}$ equal to $[0.00000000000000000000001\times 2^{-126}, 0.01\times 2^{-126}]$ and $\mathtt{x},\mathtt{y}\in[-\infty,+\infty]$: our algorithm produces the constraint $\mathtt{x}\in[-1.00000000000000000000001\times 2^{-46}, 1.00000000000000000000001\times 2^{-46}]$ whereas classical filtering is unable to infer anything on x.

4.7.2. The Second Indirect Projection The discussion in Section 4.7.1 shows that, for $|z|\le 1$, we have $\bar{\delta}'_{\oslash}(z)=f_{\max}$. We thus need to study $\bar{\delta}'_{\oslash}(z)$ for $|z|>1$. It turns out that, due to rounding, the restriction of $\bar{\delta}'_{\oslash}$ over that subdomain is not a simple function. Given $z\in\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$, $\bar{\delta}'_{\oslash}(z)$ is the maximum y such that $x\oslash y=z$. Note that, in order to maximize y, x must be maximized as well. A qualitative reasoning on the reals tells us that, since $f_{\max}/(f_{\max}/z)=z$, y should be roughly equal to $f_{\max}/|z|$. Indeed, it can be proved that, for $|z|>1$, $f_{\max}\oslash\bigl(f_{\max}\oslash|z|\bigr)$ is equal to z, z^- or z^+ depending on the value of z. This allows the determination of a rather small upper bound to the values that $\mathtt{z}$ may take, which is ultimately our goal for filtering $\mathtt{y}$ values. To this aim we define the function $\tilde{\delta}'_{\oslash}$.

Definition 7. The function $\tilde{\delta}'_{\oslash}\colon\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}\to\mathbb{F}^{+}_{p,e_{\max}}$ is defined, for each $z\in\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$, as follows:

$$\tilde{\delta}'_{\oslash}(z)=\begin{cases} f_{\max}\oslash|z|^{--}, & \text{if } 1^+<|z|\le f_{\max};\\ f_{\max}, & \text{otherwise.}\end{cases}$$

It turns out that $\tilde{\delta}'_{\oslash}(z)$ satisfies the dual of Property (3), i.e., it is a correct upper bound, while it does not satisfy the dual of Property (2), i.e., smaller correct upper bounds might exist.

Theorem 5. *Let $\mathbb{F}''_{\oslash}=\mathbb{F}^{\mathrm{sub}}_{p,e_{\max}}$ and $\bar{\mathbb{F}}''_{\oslash}=\mathbb{F}^{+}_{p,e_{\max}}$. Let $\bar{\delta}'_{\oslash}\colon\mathbb{F}''_{\oslash}\to\bar{\mathbb{F}}''_{\oslash}$ be a function satisfying* (4). *Then, for $0<|z|\le 1^+$ or $z=+\infty$, $\bar{\delta}'_{\oslash}(z)\le\tilde{\delta}'_{\oslash}(z)$; moreover, for $1^+<|z|\le f_{\max}$, $\bar{\delta}'_{\oslash}(z)<\tilde{\delta}'_{\oslash}(z)$.*

Proof. Given in the Online Supplement (Bagnara et al. 2015).

Dually, a lower bound for the function $\underline{\delta}'_{\oslash}$ can be obtained by means of the function $\underline{\tilde{\delta}}'_{\oslash}$, defined by $\underline{\tilde{\delta}}'_{\oslash}(z)=-\tilde{\delta}'_{\oslash}(z)$.

The value $N\in[\underline{\mathtt{z}},\overline{\mathtt{z}}]$ that maximizes $\tilde{\delta}'_{\oslash}$ is the one that has the smallest absolute value, i.e., $N=\min\bigl\{|\underline{\mathtt{z}}|,|\overline{\mathtt{z}}|\bigr\}$. Since $\underline{\tilde{\delta}}'_{\oslash}$ is defined as $-\tilde{\delta}'_{\oslash}(z)$, N is also the value that minimizes $\underline{\tilde{\delta}}'_{\oslash}$. Thus, if $[\underline{\mathtt{z}},\overline{\mathtt{z}}]$ does not contain zeros, $\tilde{\delta}'_{\oslash}(N)$ (resp., $\underline{\tilde{\delta}}'_{\oslash}(N)$) is an upper bound (resp. a lower bound) for $\mathtt{x}$ with respect to the constraint $\mathtt{z}=\mathtt{x}\oslash\mathtt{y}$. The restriction to intervals not containing zeros is justified by the fact that if, e.g., $z=0$, then the equality $z=x\oslash y$ holds with $y=\infty$ for each x such that $0 \leq x \leq f_{\max}$. Hence, as in the case of the first projection, no useful filtering can be applied to y. Analogously to the case of the filter for the first projection, this filter is useful whenever the interval of x contains infinities. In this case, in fact, it is able to derive useful bounds for y where the standard indirect projection does not allow any refinement of the interval. Just as is the case for multiplication and the first indirect projection of division, the standard indirect projection and filtering by maximum ULP are mutually exclusive: one applies when the other cannot derive anything useful.

Note that, only for this projection, we have chosen to compute a (very small) upper bound that, in general, is not the least upper bound. We did so in order to trade precision for efficiency: this way we have an algorithm that only uses floating-point machine arithmetic operations on the same format used by the analyzed constraint $\mathtt{z} = \mathtt{x} \oslash \mathtt{y}$. When using rationals or larger floating-point formats is an option, a pruning variant (as in the previous case, one that cuts off infinities) of a second indirect projection satisfying the precision constraints set forth in (Michel 2002) may result in extra precision at a higher computational cost.

Example 4. Consider the IEEE 754 single-precision division constraint $\mathtt{z} = \mathtt{x} \oslash \mathtt{y}$ with initial intervals $\mathtt{z} \in [1.0\cdots010 \times 2^{110}, 1.0 \times 2^{121}]$ and $\mathtt{x}, \mathtt{y} \in [-\infty, +\infty]$. We have

$$
\begin{aligned}
\tilde{\delta}'_{\oslash}(1.0\cdots01 \times 2^{110}) &= 1.1\cdots1 \times 2^{127} \oslash \bigl((1.0\cdots01 \times 2^{110})^-\bigr)^- \\
&= 1.1\cdots1 \times 2^{127} \oslash 1.1\cdots1 \times 2^{109} \\
&= 1.0 \times 2^{18}, \\
\underline{\tilde{\delta}}'_{\oslash}(1.0\cdots01 \times 2^{110}) &= -1.1\cdots1 \times 2^{127} \oslash \bigl((1.0\cdots01 \times 2^{110})^-\bigr)^- \\
&= -1.0 \times 2^{18}.
\end{aligned}
$$

Filtering by maximum ULP improves upon classical filtering, which gives nothing, with the constraint $\mathtt{y} \in [-1.0 \times 2^{18}, 1.0 \times 2^{18}]$.

4.8. Synthesis

Table 1 provides a compact presentation of filtering by maximum ULP.

Table 1 Filtering by maximum ULP synopsis

Constraint	$\mathtt{x} \subseteq \cdot$	$\mathtt{y} \subseteq \cdot$	Condition(s)
$\mathtt{z} = \mathtt{x} \oplus \mathtt{y},\ 0 < \mathtt{z} \le f_{\max}$	$[\ \underline{\delta}_{\oplus}(\zeta\,),\ \ \bar{\delta}_{\oplus}(\zeta\,)]$	$[\ \underline{\delta}_{\oplus}(\zeta\,),\ \ \bar{\delta}_{\oplus}(\zeta\,)]$	$\zeta = \mu_{\oplus}(\mathtt{z}),\quad -f_{\max} \le \underline{\delta}_{\oplus}(\zeta),\quad \bar{\delta}_{\oplus}(\zeta) \le f_{\max}$
$\mathtt{z} = \mathtt{x} \oplus \mathtt{y},\ -f_{\max} \le \mathtt{z} < 0$	$[-\bar{\delta}_{\oplus}(\zeta'), -\underline{\delta}_{\oplus}(\zeta')]$	$[-\bar{\delta}_{\oplus}(\zeta'), -\underline{\delta}_{\oplus}(\zeta')]$	$\zeta' = \mu_{\oplus}(-\mathtt{z}),\quad -f_{\max} \le \underline{\delta}_{\oplus}(\zeta'),\quad \bar{\delta}_{\oplus}(\zeta') \le f_{\max}$
$\mathtt{z} = \mathtt{x} \ominus \mathtt{y},\ 0 < \mathtt{z} \le f_{\max}$	$[\ \underline{\delta}_{\oplus}(\zeta\,),\ \ \bar{\delta}_{\oplus}(\zeta\,)]$	$[-\bar{\delta}_{\oplus}(\zeta\,), -\underline{\delta}_{\oplus}(\zeta\,)]$	$\zeta = \mu_{\oplus}(\mathtt{z}),\quad -f_{\max} \le \underline{\delta}_{\oplus}(\zeta),\quad \bar{\delta}_{\oplus}(\zeta) \le f_{\max}$
$\mathtt{z} = \mathtt{x} \ominus \mathtt{y},\ -f_{\max} \le \mathtt{z} < 0$	$[-\bar{\delta}_{\oplus}(\zeta'), -\underline{\delta}_{\oplus}(\zeta')]$	$[\ \underline{\delta}_{\oplus}(\zeta'),\ \ \bar{\delta}_{\oplus}(\zeta')]$	$\zeta' = \mu_{\oplus}(-\mathtt{z}),\quad -f_{\max} \le \underline{\delta}_{\oplus}(\zeta'),\quad \bar{\delta}_{\oplus}(\zeta') \le f_{\max}$
$\mathtt{z} = \mathtt{x} \otimes \mathtt{y},\ \lvert z \rvert \le 2^{2-p}(2 - 2^{1-p})$	$[\ \underline{\delta}_{\otimes}(m),\ \ \bar{\delta}_{\otimes}(m)]$	$[\ \underline{\delta}_{\otimes}(m),\ \ \bar{\delta}_{\otimes}(m)]$	$m = \max\bigl\{\lvert\underline{\mathtt{z}}\rvert, \lvert\overline{\mathtt{z}}\rvert\bigr\}$
$\mathtt{z} = \mathtt{x} \oslash \mathtt{y},\ 0 < \lvert\mathtt{z}\rvert \le 1$	$[\ \underline{\delta}_{\oslash}(m),\ \ \bar{\delta}_{\oslash}(m)]$	$[-f_{\max},\quad +f_{\max}]$	$m = \max\bigl\{\lvert\underline{\mathtt{z}}\rvert, \lvert\overline{\mathtt{z}}\rvert\bigr\}$
$\mathtt{z} = \mathtt{x} \oslash \mathtt{y},\ 1 < \lvert\mathtt{z}\rvert \le f_{\max}$		$[\ \underline{\tilde{\delta}}'_{\oslash}(n),\ \ \tilde{\delta}'_{\oslash}(n)]$	$n = \min\bigl\{\lvert\underline{\mathtt{z}}\rvert, \lvert\overline{\mathtt{z}}\rvert\bigr\}$

$$\bar{\delta}_{\oplus}(z) = \begin{cases} \beta, & \text{if } 0 < z < +\infty, \\ \alpha, & \text{if } -\infty < z < 0; \end{cases} \qquad \underline{\delta}_{\oplus}(z) = -\bar{\delta}_{\oplus}(-z);$$

$$\bar{\delta}_{\otimes}(z) = \begin{cases} \lvert z\rvert \oslash f_{\min}, & \text{if } z \ge f_{\min}^{\mathrm{nor}}; \\ \bigl(\lvert z\rvert \oslash f_{\min}\bigr) \oplus 2^{-1}, & \text{if } 0 < z < f_{\min}^{\mathrm{nor}} \text{ and } \mathrm{even}(z); \\ \Bigl(\bigl(\lvert z\rvert \oslash f_{\min}\bigr) \oplus 2^{-1}\Bigr)^{-}, & \text{if } 0 < z < f_{\min}^{\mathrm{nor}} \text{ and } \mathrm{odd}(z); \end{cases} \qquad \underline{\delta}_{\otimes}(z) = -\bar{\delta}_{\otimes}(z);$$

$$\bar{\delta}_{\oslash}(z) = \begin{cases} \lvert z\rvert \otimes f_{\max}, & \text{if } f_{\min}^{\mathrm{nor}} \le \lvert z\rvert \le 1; \\ \bigl(\lvert z\rvert \otimes f_{\max}\bigr) \oplus 2^{q}, ^{(*)} & \text{if } 0 \le \lvert z\rvert < f_{\min}^{\mathrm{nor}} \wedge \bigl(\lvert z\rvert \ne 1 \times 2^{e_z} \vee e_z = e_{\min} - 1\bigr); \\ \Bigl(\bigl(\lvert z\rvert \otimes f_{\max}\bigr) \oplus 2^{q}\Bigr)^{-}, & \text{otherwise}; \end{cases} \qquad \underline{\delta}_{\oslash}(z) = -\bar{\delta}_{\oslash}(z);$$

$$\tilde{\delta}'_{\oslash}(z) = \begin{cases} f_{\max} \oslash \lvert z\rvert^{--}, & \text{if } 1^{+} < \lvert z\rvert \le f_{\max}; \\ f_{\max}, & \text{otherwise}; \end{cases} \qquad \underline{\tilde{\delta}}'_{\oslash}(z) = -\tilde{\delta}'_{\oslash}(z);$$

$(*)\quad q = 1 - p + e_{\min} + e_{\max}.$

5. Discussion

This work is part of a long-term research effort concerning the correct, precise and efficient handling of floating-point constraints (Belaid 2013, Belaid et al. 2012, Botella et al. 2006, Carlier and Gotlieb 2011, Marre and Michel 2010, Michel 2002, Michel et al. 2001) for software verification purposes.

Restricting the attention to test data generation other authors have considered using search-based techniques with a specific notion of distance in their fitness function (Lakhotia et al. 2010a,b). For instance, search-based tools like AUSTIN and FloPSy can generate a test input for a specific path by evaluating the path covered by some current input with respect to a targeted path in the program. However, they cannot *solve* the constraints of path conditions, since: 1) they cannot determine unsatisfiability when the path is infeasible, and 2) they can fail to find a test input while the set of constraints is satisfiable (Bagnara et al. 2013).

Recently, Borges et al. (2012) combined a search-based test data generation engine with the RealPaver (Granvilliers and Benhamou 2006) interval constraint solver, which is well-known in the Constraint Programming community. Even though constraint solvers over continuous domains (e.g., RealPaver (Granvilliers and Benhamou 2006), Quimper (Chabert and Jaulin 2009) or ICOS (Lebbah 2009)) and the work described in the present paper are based on similar principles, the treatment of intervals is completely different. While our approach preserves all the solutions over the floats, it is not at all concerned with solutions over the reals. In contrast, RealPaver preserves solutions over the reals by making the appropriate choices in the rounding modes used for computing the interval bounds, but RealPaver can lose solutions over the floats. For instance, a constraint like $(\mathtt{x} > 0.0 \wedge \mathtt{x} \oplus 10000.0 \leq 10000.0)$ is shown to be unsatisfiable on the reals by RealPaver, while it is satisfied by many IEEE 754 floating-point values of single or double precision format for $\mathtt{x}$ (Botella et al. 2006). Note that RealPaver has recently been used to tackle test input generation in presence of transcendental functions (Borges et al. 2012), but this approach, as mentioned by the authors of the cited paper, is neither correct nor complete due to the error rounding of floating-point computations.

The CBMC model checker (Clarke et al. 2004) supports floating-point arithmetic using a bit-precise floating-point decision procedure based on propositional encoding. According to (D'Silva et al. 2012) "CBMC translates the floating-point arithmetic to large propositional circuits which are hard for SAT solvers." The technique presented in this paper is orthogonal to decision procedures over floating-point computations, such as those used in CBMC (Clarke et al. 2004) or CDFL (D'Silva et al. 2012). Of course, implementing the filtering procedures suggested here would require dedicated bitwise encodings, but this would enable to perform more constraint-based reasoning over these computations.

6. Conclusion

This paper concerns constraint solving over binary floating-point numbers. Interval-based consistency techniques are very effective for the solution of such numerical constraints, provided precise and efficient filtering algorithms are available. We reformulated and corrected the filtering algorithm proposed by Marre and Michel (2010) for addition and subtraction. We proposed a uniform framework that generalizes the property identified by Marre and Michel to the case of multiplication and division. We also revised, corrected and extended our initial ideas, sketched in Carlier and Gotlieb (2011), to subnormals and to the effective treatment of floating-point division. All algorithms have been proved correct. In order to gain further confidence on the algorithms, we have exhaustively tested a first prototype, symbolic implementation on floating-point numbers with a small number of bits (e.g., $p = 6$ and $e_{\max} = 3$). The implementation working on IEEE 754 formats was also tested with a variety of methodologies with the help of test-suites like the one by the IBM Labs in Haifa (2008).

An important objective of this work has been to allow maximum efficiency by defining all algorithms in terms of IEEE 754 elementary operations on the same formats as the ones of the filtered constraints. Indeed, the computational cost of filtering by maximum ULP as defined in the present paper and properly implemented is negligible. In fact, all the filters defined in this paper can be directly translated into constant-time algorithms (as IEEE 754 formats have fixed size) based on IEEE 754 elementary operations and simple bitwise manipulations. Moreover, for multiplication and division, precise conditions are given in order to decide whether standard filtering or our filtering by maximum ULP is applied: it is one or the other, never both of them. As shown in (Bagnara et al. 2013), the improvement of filtering procedures with these techniques brings significant speedups of the overall constraint solving process, with only occasional, negligible slowdowns. Note that the choice of different heuristics concerning the selection of constraints and variables to subject to filtering and the labeling strategy has a much more dramatic effect on solution time, even though the positive or negative effects of such heuristics change wildly from one analyzed program to the other. Filtering by maximum ULP contributes to reducing this variability. To understand this, consider the elementary constraint $\mathtt{z} = \mathtt{x} \odot \mathtt{y}$: if `x` and `y` are subject to labeling before `z`, then filtering by maximum ULP will not help. However, `z` might be labeled before `x` or `y`: this can happen under *any* labeling heuristic and constitutes a performance bottleneck. In the latter case, filtering by maximum ULP may contribute to a much improved pruning of the domains of `x` and `y` and remove the bottleneck.

Future work includes coupling filtering by maximum ULP with sophisticated implementations of classical filtering based on multi-intervals and with dynamic linear relaxation algorithms (Denmat et al. 2007) using linear relaxation formulas such as the ones proposed by Belaid et al.

(2012). Another extension, by far more ambitious, concerns the correct handling of transcendental functions (i.e., sin, cos, exp, ...): as IEEE 754 only provides recommendations rather than formal requirements for these functions, solutions will be dependent on the particular implementation and/or be imprecise; in other words, generated test inputs will not be applicable to other implementations and/or may fail to exercise the program paths they were supposed to traverse.

References

Ammann, P. E., J. C. Knight. 1988. Data diversity: An approach to software fault tolerance. *IEEE Transactions on Computers* **37** 418–425.

Arcuri, A. 2009. Theoretical analysis of local search in software testing. O. Watanabe, T. Zeugmann, eds., *Proceedings of the 5th International Symposium on Stochastic Algorithms: Foundations and Applications (SAGA 2009)*, *Lecture Notes in Computer Science*, vol. 5792. Springer-Verlag, Berlin, Sapporo, Japan, 156–168.

Bagnara, R., M. Carlier, R. Gori, A. Gotlieb. 2013. Symbolic path-oriented test data generation for floating-point programs. *Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation*. IEEE Press, Luxembourg City, Luxembourg.

Bagnara, R., M. Carlier, R. Gori, A. Gotlieb. 2015. Online Supplement to "Exploiting Binary Floating-Point Representations for Constraint Filtering". Available at `https://www.informs.org/Pubs/IJOC/Online-Supplements`.

Belaid, M. S. 2013. Résolution de contraintes sur les flottants dédié à la vérification de programmes. Thèse pour obtenir le titre de "Docteur en Sciences", École doctorale STIC, Université de Nice — Sophia Antipolis, Nice, France.

Belaid, M. S., C. Michel, M. Rueher. 2012. Boosting local consistency algorithms over floating-point numbers. M. Milano, ed., *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, *Lecture Notes in Computer Science*, vol. 7514. Springer-Verlag, Berlin, Québec City, Canada, 127–140.

Blanc, B., F. Bouquet, A. Gotlieb, B. Jeannet, T. Jeron, B. Legeard, B. Marre, C. Michel, M. Rueher. 2006. The V3F project. *Proceedings of the 1st Workshop on Constraints in Software Testing, Verification and Analysis (CSTVA'06)*. Nantes, France.

Borges, M., M. d'Amorim, S. Anand, D. Bushnell, C. S. Pasareanu. 2012. Symbolic execution with interval solving and meta-heuristic search. *Proceedings of the 5th IEEE International Conference on Software Testing, Verification and Validation*. IEEE Computer Society, Montreal, Canada, 111–120.

Botella, B., A. Gotlieb, C. Michel. 2006. Symbolic execution of floating-point computations. *Software Testing, Verification and Reliability* **16** 97–121.

Burdy, L., J.-L. Dufour, T. Lecomte. 2012. The B method takes up floating-point numbers. *Proceedings of the 6th International Conference & Exhibition on Embedded Real Time Software and Systems (ERTS 2012)*. Toulouse, France. Available at `http://www.erts2012.org/Site/0P2RUC89/5C-2.pdf`.

Carlier, M., A. Gotlieb. 2011. Filtering by ULP maximum. *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2011)*. IEEE Computer Society, Boca Raton, Florida, USA, 209–214.

Chabert, G., L. Jaulin. 2009. Contractor programming. *Artificial Intelligence* **173** 1079–1100.

Chan, F. T., T. Y. Chen, S. C. Cheung, M. F. Lau, S. M. Yiu. 1998. Application of metamorphic testing in numerical analysis. *Proceedings of the IASTED International Conference on Software Engineering (SE'98)*. ACTA Press, Las Vegas, Nevada, USA, 191–197.

Clarke, E. M., D. Kroening, F. Lerda. 2004. A tool for checking ANSI-C programs. K. Jensen, A. Podelski, eds., *Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of the 10th International Conference (TACAS 2004)*, *Lecture Notes in Computer Science*, vol. 2988. Springer, Barcelona, Spain, 168–176.

Cousot, P., R. Cousot. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*. ACM Press, Los Angeles, CA, USA, 238–252.

Denmat, T., A. Gotlieb, M. Ducassé. 2007. Improving constraint-based testing with dynamic linear relaxations. *Proceedings of the 18th IEEE International Symposium on Software Reliability (ISSRE 2007)*. IEEE Computer Society, Trollhättan, Sweden, 181–190.

D'Silva, V., L. Haller, D. Kroening, M. Tautschnig. 2012. Numeric bounds analysis with conflict-driven learning. C. Flanagan, B. König, eds., *Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of the 18th International Conference (TACAS 2012)*, *Lecture Notes in Computer Science*, vol. 7214. Springer, Tallinn, Estonia, 48–63.

Godefroid, P., N. Klarlund, K. Sen. 2005. DART: Directed automated random testing. V. Sarkar, M. W. Hall, eds., *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI 2005)*. ACM, Chicago, IL, USA, 213–223.

Goldberg, D. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* **23** 5–48.

Goubault, E. 2001. Static analyses of the precision of floating-point operations. P. Cousot, ed., *Static Analysis: 8th International Symposium, SAS 2001*, *Lecture Notes in Computer Science*, vol. 2126. Springer-Verlag, Berlin, Paris, France, 234–259.

Granvilliers, L., F. Benhamou. 2006. Algorithm 852: RealPaver: An interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software* **32** 138–156.

IBM Labs in Haifa, FPgen Team. 2008. Floating-point test-suite for IEEE. Available at `https://www.research.ibm.com/haifa/projects/verification/fpgen/papers/ieee-test-suite-v2.pdf`. Version 1.02.

IEEE Computer Society. 2008. *IEEE Standard for Floating-Point Arithmetic*. The Institute of Electrical and Electronics Engineers, Inc., IEEE Std 754-2008 (revision of IEEE Std 754-1985) ed.

Korel, B. 1990. Automated software test data generation. *IEEE Transactions on Software Engineering* **16** 870–879.

Kuliamin, V. V. 2010. Standardization and testing of mathematical functions. A. Pnueli, I. Virbitskaite, A. Voronkov, eds., *Perspectives of Systems Informatics, Revised Papers from the 7th International Andrei Ershov Memorial Conference (PSI 2009), Novosibirsk, Russia, June 15–19, 2009*, *Lecture Notes in Computer Science*, vol. 5947. Springer-Verlag, Berlin, 257–268.

Lakhotia, K., M. Harman, H. Gross. 2010a. AUSTIN: A tool for search based software testing for the C language and its evaluation on deployed automotive systems. *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*. IEEE Computer Society, 101–110.

Lakhotia, K., N. Tillmann, M. Harman, J. De Halleux. 2010b. FloPSy: Search-based floating point constraint solving for symbolic execution. *Proceedings of the 22nd IFIP WG 6.1 International Conference on Testing Software and Systems*. Springer-Verlag, Berlin, Heidelberg, Natal, Brazil, 142–157.

Lebbah, Y. 2009. ICOS: a branch and bound based solver for rigorous global optimization. *Optimization Methods and Software* **24** 709–726.

Marre, B., B. Blanc. 2005. Test selection strategies for Lustre descriptions in GATeL. Y. Gurevich, A. K. Petrenko, A. Kossatchev, eds., *Proceedings of the Workshop on Model Based Testing (MBT 2004)*, *Electronic Notes in Theoretical Computer Science*, vol. 111. Elsevier Science Publishers B. V., Barcelona, Spain, 93–111.

Marre, B., C. Michel. 2010. Improving the floating point addition and subtraction constraints. D. Cohen, ed., *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP 2010)*, *Lecture Notes in Computer Science*, vol. 6308. Springer, St. Andrews, Scotland, UK, 360–367.

McMinn, P. 2004. Search-based software test data generation: A survey. *Software Testing, Verification and Reliability* **14** 105–156.

Michel, C. 2002. Exact projection functions for floating point number constraints. *Proceedings of the 7th International Symposium on Artificial Intelligence and Mathematics*. Fort Lauderdale, FL, USA.

Michel, C., M. Rueher, Y. Lebbah. 2001. Solving constraints over floating-point numbers. T. Walsh, ed., *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP 2001)*, *Lecture Notes in Computer Science*, vol. 2239. Springer-Verlag, Berlin, Paphos, Cyprus, 524–538.

Miller, W., D. L. Spooner. 1976. Automatic generation of floating-point test data. *IEEE Transactions on Software Engineering* **2** 223–226.

Monniaux, D. 2008. The pitfalls of verifying floating-point computations. *ACM Transactions on Programming Languages and Systems* **30**.

Muller, J.-M. 2005. On the definition of ulp(x). Rapport de recherche 5504, INRIA.

Scott, N. S., F. Jézéquel, C. Denis, J.-M. Chesneaux. 2007. Numerical 'health check' for scientific codes: The CADNA approach. *Computer Physics Communications* **176** 507–521.

Skeel, R. 1992. Roundoff error and the Patriot missile. *SIAM News* **25** 11.

Tang, E., E. T. Barr, X. Li, Z. Su. 2010. Perturbing numerical calculations for statistical analysis of floating-point program (in)stability. P. Tonella, A. Orso, eds., *Proceedings of the 19th International Symposium on Software Testing and Analysis (ISSTA 2010)*. ACM, Trento, Italy, 131–142.

Weyuker, E. J. 1982. On testing non-testable programs. *The Computer Journal* **25** 465–470.

Acknowledgments

We are grateful to Abramo Bagnara (BUGSENG srl, Italy) for the many fruitful discussions we had on the subject of this paper, and to Paul Zimmermann (INRIA Lorraine, France) for the help he gave us proving a crucial result. We are also indebted to Claude Michel for several constructive remarks that allowed us to improve the paper. Finally, we wish to express our gratitude to the anonymous reviewers for the many useful suggestions they contributed.