Optimal Inverse Projection of Floating-Point Addition

Diane Gallois-Wong, Sylvie Boldo, Pascal Cuoq

Numerical Algorithms manuscript No.
(will be inserted by the editor)

Optimal Inverse Projection of Floating-Point Addition

Diane Gallois-Wong · Sylvie Boldo · Pascal Cuoq

Received: date / Accepted: date

Abstract In a setting where we have intervals for the values of floating-point variables x, a, and b, we are interested in improving these intervals when the floating-point equality $x \oplus a = b$ holds. This problem is common in constraint propagation, and called the inverse projection of the addition. It also appears in abstract interpretation for the analysis of programs containing IEEE 754 operations. We propose floating-point theorems that provide optimal bounds for all the intervals. Fast loop-free algorithms compute these optimal bounds using only floating-point computations at the target precision.

Keywords Floating-Point · Inverse Projection · Abstract Interpretation

1 Introduction

Since the 1970s, floating-point (FP) arithmetic is both standardized and available on all general-purpose processors [5]. Many FP operations are available, including *correctly-rounded* ones, meaning that the result is the same as if the computation was done on an infinite number of bits and rounded afterwards. This is in particular the case for the addition, subtraction, multiplication, division, square root, and FMA. This gives us both accuracy and reproducibility.

The topic of this article is the inverse projection of the FP addition. For the mathematical addition $+$, the equation $b = x + a$ exactly defines the solution x:

This work is supported by a grant from the "Fondation CFM pour la Recherche".

D. Gallois-Wong
Université Paris-Sud
LRI, CNRS & Univ. Paris-Sud, Université Paris-Saclay, bâtiment 650, Université Paris-Sud, F-91405 Orsay Cedex, France

S. Boldo
Inria
LRI, CNRS & Univ. Paris-Sud, Université Paris-Saclay, bâtiment 650, Université Paris-Sud, F-91405 Orsay Cedex, France

P. Cuoq
TrustInSoft, 222 cour Avenue du Maine, 75014 PARIS, France

it is *the* real number $b - a$. In other words, the mathematical subtraction is the inverse projection of the mathematical addition.

Now let us consider the FP addition $\oplus$ in *binary64* and the equation $b = x \oplus a$. Then the value of x is not always uniquely defined. Certainly, in some cases x may be a singleton, such as in the equation $1 = x \oplus 2^{-100}$. Another example is $1 = x \oplus 1$, where any FP value in the interval $[-2^{-54}, 2^{-53}]$ is a solution for x. When we only have intervals for a and b, the set of solutions may not be an interval. For instance, let $a \in [2, 2^+]$ and $b = 1$. Then x may be -1 and $(-1)^{--}$ as $(-1) \oplus 2 = 1$ and $(-1)^{--} \oplus 2^+ = 1$. But the middle value $(-1)^-$ is not suitable as both $(-1)^- \oplus 2 = 1 - 2^{52}$ and $(-1)^- \oplus 2^+ = 1 + 2^{52}$. This shows that solving FP equations can be more complicated than mathematical ones.

This problem arises from an industrial application: a static analyzer that aims at automatically analyzing a C program in order to detect possible undefined behavior. Even when the focus of the analyzer is not on the numerical aspects of the program, these numerical aspects influence the control flow and the values of array indexes. Consequently, an unsound treatment of floating-point can lead to unsound detection of undefined behavior, whereas an over-approximated modelization of floating-point can lead to false positives, that is, warnings about "potential" undefined behavior that does not happen in any real execution.

More context is provided in Section 2, and has lead to several variations around the previously-described problem. Let us consider that we know that $b == x \oplus a$ (we are for instance after a test) and that we may have initial intervals for the variables a, b, and x. Then we might want the tightest possible interval for x, given the precise values of b and a (as seen above). We may also want to provide or improve over a given interval for b, given intervals for a and x. This gives us several possible improvements on the various input intervals. In this article, we are interested in identifying the tightest intervals that contains all the solution for x (and b, but this is simpler, see Section 4.4) to the equation $b = x \oplus a$, when a, b, and x are already constrained to known intervals.

The floating-point equation $b = x \oplus a$ has been studied before, especially in the field of constraint propagation. The oldest work that we know of is by Michel, Rueher, and Lebbah [9]. It aims at ensuring that an inverse interval is empty, meaning that a subcase may never happen and may be filtered out. It relies on interval analysis and provides a decision procedure to filter intervals. Then, Michel [8] focuses on one-argument monotone functions. He provides an optimal inverse projection if extra precision is available for computations, and he applies this technique in the context of test-case generation. Note that his *FB-2B-consistent* property is similar to our optimality. Botella, Gotlieb, and Michel provide inverse projections for many FP operations (addition, subtraction, multiplication, division, but also comparisons) [1]. Unfortunately, these projections are not optimal. The most similar work has been done by Marre and Michel [7]: they have focused on FP addition and subtraction. In particular, they prove results similar to our Lemma 4 and Lemma 5. We re-prove them for the sake of completeness and because the notations are quite different. These previous results sometimes provide the optimal bounds, but not in all cases as this article does.

In the discussions of this problem that we have found in the context of constraint propagation (as explained above), a loop is necessary to improve these intervals, while our optimal projections allow us to compute optimal results directly. An example of multiple successive improvements is the following one: let

$x \in [-2^{100}, -2^{52}]$, $a \in [1, 2^{100}]$, and $b \in [\frac{1}{2}, \frac{3}{2}]$. Naïve propagation of the information takes three steps to reach the optimal result: $x \in [-2^{53} + 1, -2^{52}]$, $a \in [2^{52} + 1, 2^{53}]$, and $b = 1$.

This article is organized as follows. Section 2 explains how the implementation of an general-purpose static analyzer based on Abstract Interpretation has lead to this question. Section 3 provides the FP definitions and lemmas needed in Section 4, where we provide the main theorems for the optimal inverse projection of the FP addition. Section 5 provides the corresponding algorithms while Section 6 concludes and gives some perspectives.

2 Context

Abstract Interpretation is a technique to analyze programs by the use of sound approximation [4]. In Abstract Interpretation, an abstract domain with a lattice structure defines the way information about the program is represented and we discuss in this section a new abstract domain for floating-point programs.

The context for this article is the value analysis of C programs by nonrelational Abstract Interpretation. That is, our goal is to map all the program's variables to sets of values at each point of execution.

Representing sets of values for variables in extension would be inefficient. Instead, the sets of values associated to variables are picked as elements of a predetermined abstract domain. The design of an abstract domain involves trade-offs between efficiency and precision. For instance, the well-known "intervals" abstract domain [3] is extremely efficient for floating-point variables, requiring only, for each operation of the target program, a few operations and comparisons in order to compute from the intervals associated to the operands a superset of the values the result can take. The "intervals" abstract domain often matches the reality of the target program, designed in terms of ranges of valid values for inputs and intermediate computations. Still, this domain can be imprecise; for instance, when analyzing a program that converts an integer variable determined to lie between 0 and 2 to a `double`, the resulting floating-point interval, [+0.0;2.0], contains roughly 2^{62} values that cannot actually happen during a real execution, in addition to the values +0.0, 1.0 and 2.0 that can actually happen.

In this article, we assume that the target C programs run in round-to-nearest mode all the time (that is, the rounding mode is never changed from its default) and to have been compiled with strict IEEE 754 compliance, `FLT_EVAL_METHOD` set to 0 [6] and `#pragma STDC FP_CONTRACT OFF`, so that most known caveats [11] can be ignored.

The abstract domain we are interested in is described in Section 2.1. Some examples of its usefulness are given in Section 2.2 and a comparison with other methods is given in Section 2.3. Finally, one of the questions this abstract domain raises is precisely stated in Section 2.4.

2.1 A precise nonrelational abstract domain for floating-point operations

The abstract domain which raises the problems partially addressed in this article is an improvement over the "intervals" abstract domain. In this new abstract domain,

a set of floating-point values is represented as a record of boolean flags indicating the individual presence of $+0.0$, -0.0, $+\infty$, $-\infty$, and NaN, as well as a floating-point interval of finite negative values, and an interval of finite positive values. Thus it is possible to represent the information that a floating-point variable is either $+0.0$ or between -1 and -2^{-52} or between 2^{-52} and 1, but not -0.0 or any finite value strictly between -2^{-52} and 2^{-52}. This single lattice element is displayed as $\{+0\} \cup [-1; -2^{-52}] \cup [2^{-52}; 1]$ for the end user. Programmatically, the static analyzer being implemented in OCaml, it looks like the following record:

```
{  poszero   = true;
   negzero   = false;
   posinf    = false;
   neginf    = false;
   nan       = false;
   negfinite = (-1.0, -. (ldexp 1.0 (-52)));
   posfinite = (1.0, ldexp 1.0 (-52)) }
```

The abstract domain, the abstract versions of the basic operations, and their projections, are defined in advance and may then be applied to any program. For this reason, it pays to define them with care and to implement them efficiently.

2.2 Examples

To convince the reader of the advantages of this particular abstract domain, a few examples where it behaves better than simple interval arithmetic follow.

2.2.1 Direct computations

Let us assume that the `double` variables `a` and `b` have been inferred respectively to lie in [-2.0; -1.0] and in [1.0; 2.0], and that the next instruction in the analyzed program is the following assignment:

```
x = a + b;
```

After this assignment, the `double` variable `x` can be inferred to be in the set $\{+0\} \cup [-1; -2^{-52}] \cup [2^{-52}; 1]$. This is more accurate than the result obtained with the "intervals" abstract domain. In the "intervals" abstract domain, the lattice element that best can represent the result is $[-1.0; 1.0]$.

For a second example, let us revisit the assignment `x = (double)a;` where the integer variable `a` is known to be between 0 and 2. We already pointed out that the resulting set of values for the `double` variable `x` when relying on the "interval" abstract domain is $[+0.0; 2.0]$. With the improved abstract domain described in this section, the set of values that `x` can take can be represented as $\{+0.0\} \cup [1.0; 2.0]$. This set also contains parasitic values that `x` cannot take during a real execution, but it contains much fewer of these parasitic values than $[0.0; 2.0]$ does (only $2^{52} - 1$ of them instead of nearly 2^{62}).

The design choice of representing NaN, zeroes and infinities as separate boolean flags attempts to capture many of the natural properties of the IEEE 754 basic operations. Consider now the following snippet, reached with `a` $\in [1.0; 2^{600}]$, `b` $\in [1.0; 2^{600}]$, `c` $\in [0.125; 0.25]$, and `d` $\in \{+0\} \cup [2^{-52}; 1]$:

```
1 x = (a * b) * c;
2 if (x == INFINITY) exit(1); // overflow
3 y = x - 0x1.0p1022;
4 z = d / y;
```

The abstract domain described in this section precisely captures that after the first instruction, `x` is in $\{+\infty\} \cup [0.125; 0\text{x}1.\text{fffffffffffp}1021]$. Thanks to this precise value for `x`, the abstract domain can guarantee that the variable `y` is computed as a negative value and that `z` is not computed as NaN. The "intervals" abstract domain would characterize the value of `x` at the end of the first line as $[1.0; +\infty]$ and[1] would infer that the range [-0x1.0p1022; 0x1.7fffffffffffp1023] results from the subtraction at the third line. This range contains zero and positive numbers. As a consequence NaN would appear to be a possible result from the division of zero by zero at the fourth line.

The operational steps for direct computations (that is to say the processing of assignments) in the improved abstract domain are relatively straightforward, and we will not discuss them any further in this article.

2.2.2 Conditionals

In the last example, the abstract interpreter needed to propagate to line 3 the information that thanks to the conditional at line 2, `x` could not be $+\infty$. In general, the abstract interpreter needs to propagate, in the "then" and "else" branches of a conditional, abstract states that reflect that the floating-point condition is respectively true and false. The floating-point conditions encountered in the analyzed program are not always of the form "variable == constant". For an arbitrary condition, determining the values of variables that make it true or false is arbitrarily difficult.

The basic problem we study in this article is expressed in terms of an equation $a \oplus x = b$ with variables the values of which are initially known as intervals of finite floats of the same sign, and must be improved as such. Solving this problem is general enough to provide a treatment for inequations. Optimizing the bounds of the `double` variables `a`, `b`, `x` for the "then" branch of the conditional `if (a + x <= b) ...` is equivalent to optimizing $a_{min}, a_{max}, x_{min}, x_{max}, b_{min}$ for the problem $a + x = b$ where a and x are assumed to be in the same range as the eponymous program variables, and b is assumed to be in the set $\{+\infty\} \cup [b_{min}, 0\text{x}1.\text{fffffffffffp}1023]$.

In the same spirit, with the aforementioned assumptions about the target compilation platform, the strict inequality `a + x < b` it equivalent to the inequality `a + x <= b`$^-$, where `b`$^-$ is the floating-point number immediately below `b`.

2.3 Comparison with other abstract domains

2.3.1 Interval analysis

The above discussion may remind the reader of interval analysis [12], where intervals of floating-point values are used to approximate the sets of reals that can

[1] Note that designing interval arithmetic operations to work with infinite bounds requires special care, but is not impossible.

result from real operations. The important nuance here is that we use intervals of floating-point values to approximate the sets of floating-point values that can result from floating-point operations. In other words, our goal is only to accurately predict how the floating-point program actually behaves, and not to predict the behavior it would have had if it computed over reals. For this reason, round-to-nearest is used to compute the bounds of resulting intervals. Indeed, the minimum `double` that can be reached by the `double` addition of [a;b] and [c;d] is the `double` addition of a and c, even if the result of the real addition could be lower than that.

2.3.2 Relational abstract interpretation of floating-point programs

Relational abstract interpretation infers, in addition to sets of values for individual variables, relations that are guaranteed to hold between these variables. Similarly to non-relational abstract domains, the shape of the relations is fixed in advance, in effect limiting the sets of values that can be inferred for the tuple of the program's variables, in exchange for compactness and efficiency.

As an example, if the `double` variable `y` is known to be in [8; 14] and `x` in the program is initialized through the assignment `x = y + 1.75;`, a relational abstract interpreter may infer that henceforth `x` is equal to $\texttt{y} + 1.75 + e$, where e is an error term bounded by the ULP of the binade [8; 16]. This allows to automatically gain information about `x` when information is gained about `y` later, or vice-versa. On the other hand, this sort of design choice does not allow these domains to extract the "last bit" sort of information that our nonrelational abstract domain is able to recover on the examples that it is advantageous for. As an illustration, our nonrelational abstract domain can automatically infer the absence of underflow for simple floating-point programs, whereas relational abstract domains typically represent only convex sets of values, and include all denormal values in sets that contain two normal values of opposite signs. Choosing an abstract domain is a matter of choices and trade-offs.

In order to take full advantage of the inferred relations' symmetries, relational abstract domains are generally defined in terms of constraints of a pre-determined shape over a field. The field $\mathbb{Q}$ of rationals would typically be used to represent relations between floating-point variables.

The rational parameters typically used in the ideal presentation of a relational abstract domain can be too costly to represent and compute with for practical use on floating-point programs. These rational parameters can in turn be approximated soundly by floating-point parameters in the implementation of the abstract domain [10, 2]. Of course, rather than removing a layer of abstraction, this should be seen as adding a second one. While this approach improves the practicality of the relational abstract domain in practice (trading a little accuracy for much improved space and time requirements), it makes these implementations even less likely to retrieve the last-bit information that our nonrelational abstract domain is designed to go after.

2.3.3 Propagation of floating-point constraints

In the non-relational abstract interpretation setting, any relationship that may exist because of previous computations (for instance the variable `b` having been computed from `a` by the assignment `b = a * a * a;`) has already been lost at

the time the conditional `if (x + a == b) ...` is interpreted. Thus, the goal is to extract as much information as possible, quickly and without relying on propagation between constraints that are not available anyway. The situation would be similar even if we were using relational abstract domains: at best, a rough over-approximation of existing relations between variables can be expected to be available.

The abstract domain sketched out above borrows insights from the field of constraint propagation [7], and the optimal solution we describe to the basic problem may be useful in the same context, in order to minimize the number of times the same constraints need to be awakened. In this context, it may be useful to remark about the information flow within the ternary constraint $a \oplus x = b$. As expected, information about a and x provide information about b. Moreover, new information about a and b may improve the information on x: this is the topic of this article. Examples exist of intervals for a, b and x that cannot be improved by the constraint $a \oplus x = b$, but where new information about x can be injected that does not allow the constraint to improve the range of a, does not allow to improve the range of b, but allows to improve the range of x. This is not unique in constraint propagation: a similar situation arises with the integer constraint $x\%a = b$ when $x \in [0, 10]$, $a \in \{2\}$ and $b \in \{0\}$, and the new information $x \in [0, 9]$ arrives. The surprise comes from the fact that floating-point addition, a seemingly simple operation, suffices to cause this behavior to emerge. This is the reason why the problem must be phrased: "given the constraint $a \oplus x = b$ and initial information about a, b, *and* x, compute the best bounds for x". An implementation shortcut for simpler constraints is to compute a solution for x that ignores the initial information about x, and then to intersect this solution with the information already known about x. For the reason above, this shortcut does not work when implementing the integer constraint $x\%a = b$, and it does not work when implementing the floating-point constraint $a \oplus x = b$.

2.4 Formalising the Problem

Let $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$, $\mathbf{a} = [\underline{\mathbf{a}}, \overline{\mathbf{a}}]$, $\mathbf{b} = [\underline{\mathbf{b}}, \overline{\mathbf{b}}]$ floating point intervals. Let

$$X = \{x \in \mathbf{x} \mid \exists a \in \mathbf{a}.\ x \oplus a \in \mathbf{b}\}.$$

We want to compute $\min X$ and $\max X$, which always exist for sets of floating point numbers (with the convention that the minimum of the empty set is $+\infty$ and its maximum is $-\infty$).

We want to do the same for the following sets A and B. Note that the problem about A is exactly the same as for X since $\oplus$ is commutative.

$$A = \{a \in \mathbf{a} \mid \exists x \in \mathbf{x}.\ x \oplus a \in \mathbf{b}\}$$
$$B = \{b \in \mathbf{b} \mid \exists x \in \mathbf{x}.\ \exists a \in \mathbf{a}.\ x \oplus a = b\}$$

The same questions arise when $\oplus$ is replaced with $\otimes$, and also $\oslash$, but this is out of the scope of this article. The equation with $\ominus$ on the other hand does not add any conceptual difficulty to the equation using $\oplus$.

Thanks to the compatibility of the opposite with round-to-nearest rounding modes, we can assume that $\overline{\mathbf{b}} \geqslant 0$ without loss of generality. Moreover, as explained

in §2.1, it is useful to split variable domains into positive values, negative values, zeroes, infinities and NaNs. When at least one of the intervals $\mathbf{x}$, $\mathbf{a}$ and $\mathbf{b}$ is either empty or the singleton corresponding to one of the specials values, the problem is rather easy. Therefore, we will often assume that $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, and $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ (resp. $\underline{\mathbf{a}}$ and $\overline{\mathbf{a}}$) are either both positive or both negative with $\underline{\mathbf{x}} \leqslant \overline{\mathbf{x}}$.

3 FP Definitions and Preliminary Results

In order to provide algorithms that compute optimal inverse projections, we rely on various theorems given in Section 4. Before that, and in order to try to make this section more readable, we first pose notations and give some useful lemmas.

We define the set $\mathcal{F}$ of floating point numbers of the form

$$(-1)^s * m * 2^e = (-1)^s * b_0.b_1...b_{p-1} * 2^e$$

where s is the sign bit, the significand m has a fixed precision p and its bits are $(b_i)_{0 \leqslant i < p}$, and the exponent e is an integer greater than or equal to a fixed number e_{min}. The smallest positive floating point number is then $2^{e_{min}-p+1}$. Moreover, b_0 the first bit of m is always 1 unless the exponent is $e = e_{min}$. We also add an element 0 to $\mathcal{F}$ (we could distinguish 0_+ and 0_- but it does not matter here). The other special values from the IEEE-754 standard [5] are $\pm\infty$ and NaNs. Overflows and these special values are not handled in the following proofs, but may be handled in the real program using additional tests.

A floating point is *even* if its last bit b_{p-1} is 0, *odd* if it is 1. The FP number 0 is even. As explained, we assume the default rounding mode: round-to-nearest ties-to-even, noted $\circ$. We denote $\oplus$, $\ominus$, $\otimes$, $\oslash$ the FP operations, that are assumed to have a correct rounding: $x \oplus y = \circ(x+y)$. The rounding towards $+\infty$ is denoted by $\triangle$ and towards $-\infty$ by $\triangledown$.

We denote x^+ or $succ\left(\ x\ \right)$ the successor of $x \in \mathcal{F}$, defined as the expected minimum: $x^+ = \min\{y \in \mathcal{F} \mid y > x\}$. Similarly, we denote x^- or $pred\left(\ x\ \right)$ the predecessor of x.

Contrarily to the usual notation ulp [13] (with a choice on what is worth $ulp(2^e)$), we have two different ulps. First, $ulp^+(x) = x^+ - x$, then $ulp^-(x) = x - x^-$ for $x \in \mathcal{F}$. Moreover, $ulp^+(r) = ulp^-(r) = \triangle(r) - \triangledown(r)$ for $r \notin \mathcal{F}$. Except for powers of two, the two ulps are equal (and equal to the usual ulp). For a power of 2 in the normal range, $ulp^+\left(2^e\right) = 2^{e-p+1}$ is also equal to the usual $ulp\left(2^e\right)$ while $ulp^-\left(2^e\right) = 2^{e-p}$. Note also that neither ulp^+ nor ulp^- is a symmetric function, however there is some symmetry between the two definitions: $ulp^+(-x) = ulp^-(x)$. Note also that $ulp^+(0) = ulp^-(0) = 2^{e_{min}-p+1}$ that is the smallest positive (subnormal) number.

We denote $\mathcal{I}(\mathcal{F})$ the set of floating point intervals. The usual interval notations $[a, b]$, (a, b) will represent a floating point interval, that is the set of the floating point numbers which are between a and b (included or excluded) and we usually denote $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$. In particular, bold symbols such as $\mathbf{x}$ are intervals and $\underline{\mathbf{x}}$ is assumed to be its smaller element. Besides, x_{min} usually denotes the minimum of a previously-defined set for the sake of readability inside the proofs.

We will also rely on the modulo notation for FP numbers. $x \equiv 0[v]$ means that x is an integer multiple of v (with v usually a power of 2).

A last notation is $y \begin{smallmatrix}^{even(x)?}\\ \leqslant\\ <\end{smallmatrix} z$ that stands for $\begin{cases} y \leqslant z \text{ if } even(x) \\ y < z \text{ if } odd(x) \end{cases}$ and $y \begin{smallmatrix}^{even(x)?}\\ <\\ \leqslant\end{smallmatrix} z$ for $\begin{cases} y < z \text{ if } even(x) \\ y \leqslant z \text{ if } odd(x) \end{cases}$.

Now let us state numerous small properties that will be useful below.

Property 1 For all $x, y \in \mathcal{F}, e \in \mathbb{Z}, r, s \in \mathbb{R}$:
(ulp^* may be replaced with either ulp^+ or ulp^- at each position)

$$|r| < |s| \implies ulp^*(r) \leqslant ulp^*(s) \tag{0.1}$$
$$|x| \geqslant |r| \implies x \equiv 0[ulp^*(r)] \tag{0.2}$$
$$|x| \geqslant 2^e \Rightarrow x \equiv 0[2^{e-p+1}] \tag{0.3}$$
$$r \equiv 0[ulp^*(r)] \implies r \in \mathcal{F} \tag{0.4}$$
$$|r| \leqslant |s| \wedge r \equiv 0[ulp^*(s)] \implies r \in \mathcal{F} \tag{0.5}$$
$$r \equiv 0[2^e] \Rightarrow \circ(r) \equiv 0[2^e] \tag{0.6}$$
$$\left.\begin{array}{r} \circ(r) = x \iff x - \frac{ulp^-(x)}{2} \begin{smallmatrix}^{even(x)?}\\ \leqslant\\ <\end{smallmatrix} r \begin{smallmatrix}^{even(x)?}\\ \leqslant\\ <\end{smallmatrix} x + \frac{ulp^+(x)}{2} \\ \circ(r) > x \iff x + \frac{ulp^+(x)}{2} \begin{smallmatrix}^{even(x)?}\\ <\\ \leqslant\end{smallmatrix} r \\ \circ(r) < x \iff r \begin{smallmatrix}^{even(x)?}\\ <\\ \leqslant\end{smallmatrix} x - \frac{ulp^-(x)}{2} \end{array}\right\} \tag{0.7}$$
$$ulp^+(\bigtriangledown(r)) = ulp^+(r) \tag{0.8}$$
$$|r - s| < ulp^*(s) \wedge r \equiv 0[ulp^*(s)] \implies r \in \{\bigtriangledown(s), \bigtriangleup(s)\} \tag{0.9}$$
$$ulp^+(r) \leqslant \frac{1}{4} ulp^+(s) \Longrightarrow |r| \leqslant \frac{1}{2}|s| \tag{0.10}$$
$$x > 2^{e_{min}+1} \implies ulp^-(x/2) = ulp^-(x)/2 = ulp^-(x) \oslash 2 \tag{0.11}$$

Now let us state two simple lemmas.

Lemma 1 *Let $x, y \in \mathcal{F}$ and $v \in 2^{\mathbb{Z}}$. If $x \leqslant y \leqslant x + v$ and $ulp^+(x) \leqslant \frac{v}{4}$ then $ulp^+(y) \leqslant \frac{v}{2}$.*

Proof When $|x| > v$, then $|y| < 2|x|$, hence the result. We now assume that $|x| \leqslant v$. Then $|y| \leqslant 2v$ and $ulp^+(2v) = \max(2v * 2^{-p+1}, 2^{e_{min}-p+1}) \leqslant \frac{v}{4}$ as $\frac{v}{4} \geqslant ulp^+(x) \geqslant 2^{e_{min}-p+1}$. □

Lemma 2 *Let $r, s, t \in \mathbb{R}$ such that $s \leqslant r < t$ and $t - s \geqslant ulp^+(r)$, then there is at least a floating point number in the real interval $[s, t)$.*

Proof If $r \in \mathcal{F}$, the result trivially holds. If $\bigtriangleup(r) \in [s, t)$, the result also trivially holds. Let us assume that $r \notin \mathcal{F}$ and $t \leq \bigtriangleup(r)$. Then $ulp^+(r) = \bigtriangleup(r) - \bigtriangledown(r)$ with $\bigtriangleup(r) < r < \bigtriangledown(r)$. Therefore, we consider $\bigtriangledown(r) \in \mathcal{F}$ and prove it belongs to $[s, t)$. First, $t > r > \bigtriangledown(r)$. Second, $\bigtriangledown(r) = \bigtriangleup(r) - ulp^+(r) \geq \bigtriangleup(r) + s - t \geq s$. □

4 Theorems on the possible values of the addition inverse projection

This section provides the main theorems for the optimal inverse projection of the floating point addition (computing optimal bounds for x when $x \oplus a = b$ given intervals for x, a and b). In §4.1, we decompose this inverse projection problem into two smaller problems that involve fewer variables. Note that this decomposition holds for any binary operator that is non-decreasing in both arguments; in particular, it could be used to handle multiplication in future work. §4.2 and §4.3 each solve one of the two smaller problems found in §4.1. For the sake of completeness, we present in §4.4 a theorem for fast direct computation of addition (optimal bounds for b when $x \oplus a = b$ given intervals for x, a and b).

Note that given intervals $\mathbf{a}$ and $\mathbf{b}$, it is easy to determine whether a given floating point number x verifies $\exists a \in \mathbf{a}, x \oplus a \in \mathbf{b}$. Indeed, we can compare $b_1 = x \oplus \underline{\mathbf{a}}$ and $b_2 = x \oplus \overline{\mathbf{a}}$ with $\underline{\mathbf{b}}$ and $\overline{\mathbf{b}}$. If $b_1 \in \mathbf{b}$ or $b_2 \in \mathbf{b}$ then x trivially verifies the property. If $b_1 > \overline{\mathbf{b}}$ or $b_2 < \underline{\mathbf{b}}$ then the property is false by monotony. If $b_1 < \underline{\mathbf{b}}$ and $b_2 > \overline{\mathbf{b}}$, we compute $a_1 = \min\{a \in \mathcal{F} \mid x \oplus a \geqslant \underline{\mathbf{b}}\}$; then x verifies the property if and only if $x \oplus a_1 \leqslant \overline{\mathbf{b}}$ (indeed, if $x \oplus a_1 \leqslant \overline{\mathbf{b}}$ then $a_1 \in \mathbf{a}$ because $x \oplus \underline{\mathbf{a}} = b_1 < \underline{\mathbf{b}} \leqslant x \oplus a_1 \leqslant \overline{\mathbf{b}} < b_2 = x \oplus \overline{\mathbf{a}}$ so the property is true, otherwise $\forall y \in \mathcal{F}, x \oplus y \notin \mathbf{b}$). While §4.2 presents a faster method to compute a_1, a naive approach using dichotomy would also work.

The difficulty lies in finding the extrema of the set of floating point numbers verifying the property $P(x) = \exists a \in \mathbf{a}, x \oplus a \in \mathbf{b}$ within a given interval. Dichotomy does not work here as the property is not convex.

4.1 Decomposition into two smaller problems

In this section, we consider a generic binary operator $\star$ on $\mathcal{F}$, non-decreasing in both arguments. In the remainder of this article, we will apply this to the floating point addition. However it could be used for other operators such as floating point multiplication. The only property about $\mathcal{F}$ that we need for now is:

Property 2 $(\mathcal{F}, \leqslant)$ is a totally ordered set. Moreover, any subset of $\mathcal{F}$ has a minimum and a maximum (with the convention $+\infty$ and $-\infty$ respectively for the empty set).

Theorem 1 *Let $\star$ a binary operator on $\mathcal{F}$,* non-decreasing in both arguments. *Let $\mathbf{x}, \mathbf{a}, \mathbf{b} \in \mathcal{I}(\mathcal{F})$ and let*

$$X \overset{def}{=} \{x \in \mathbf{x} \mid \exists a \in \mathbf{a}.\ x \star a \in \mathbf{b}\}$$
$$X_0 \overset{def}{=} \{x \in [\underline{\mathbf{x}}, +\infty) \mid\ x \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}} \ \wedge\ \exists y \in \mathcal{F}.\ x \star y \in \mathbf{b}\}.$$

Let $x_0 \overset{def}{=} \min X_0$. If $x_0 > \overline{\mathbf{x}}$ or $x_0 \star \underline{\mathbf{a}} > \overline{\mathbf{b}}$ then X is empty, otherwise $\min X = x_0$.

Proof For any $x \in X$, we have $x \geqslant \underline{\mathbf{x}}$ and there is an $a \in \mathbf{a}$ such that $x \star a \in \mathbf{b}$ then $x \star \overline{\mathbf{a}} \geqslant x \star a \geqslant \underline{\mathbf{b}}$ so $x \in X_0$; therefore $X \subset X_0$.

- Case $x_0 > \overline{\mathbf{x}}$. From $X \subset X_0$ we get $\min X \geqslant \min X_0 = x_0 > \overline{\mathbf{x}}$ so $\min X \notin X$ which means $X = \emptyset$ (and $\min X = +\infty$).

- Case $x_0 \star \underline{\mathbf{a}} > \overline{\mathbf{b}}$. By contradiction, assume there exists $x \in X$. Then there exists $a \in \mathbf{a}$ such that $x \star a \in \mathbf{b}$. But from $X \subset X_0$ we get $x \in X_0$ so $x \geqslant x_0$. Moreover $a \in \mathbf{a}$ means that $a \geqslant \underline{\mathbf{a}}$. Then $x \star a \geqslant x_0 \star \underline{\mathbf{a}} > \overline{\mathbf{b}}$, which contradicts $x \star a \in \mathbf{b}$. Therefore, X is empty.
- Case $x_0 \leqslant \overline{\mathbf{x}}$ and $x_0 \star \underline{\mathbf{a}} \leqslant \overline{\mathbf{b}}$. In particular, this means that $x_0 \in \mathbf{x}$. From $X \subset X_0$, we get $\forall x \in X,\ x_0 \leqslant x$. To show that $x_0 = \min X$, we now only need to show that $x_0 \in X$.
  By definition of X_0, $x_0 \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}$ and there exists $y_0 \in \mathcal{F}$ such that $x_0 \star y_0 \in \mathbf{b}$.
  - If $y_0 \in \mathbf{a}$ then $x_0 \in X$.
  - If $y_0 < \underline{\mathbf{a}}$ then $\underline{\mathbf{b}} \leqslant x_0 \star y_0 \leqslant x_0 \star \underline{\mathbf{a}} \leqslant \overline{\mathbf{b}}$ with $\underline{\mathbf{a}} \in \mathbf{a}$, so $x_0 \in X$.
  - If $y_0 > \overline{\mathbf{a}}$ then $\underline{\mathbf{b}} \leqslant x_0 \star \overline{\mathbf{a}} \leqslant x_0 \star y_0 \leqslant \overline{\mathbf{b}}$ with $\overline{\mathbf{a}} \in \mathbf{a}$, so $x_0 \in X$.

  In every case $x_0 \in X$. As explained above, we conclude that $x_0 = \min X$.

□

Corollary 1 *Same notations as Theorem 1. Let* $x_1 \stackrel{def}{=} \min\{x \in \mathcal{F} \mid x \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}\}$ *and* $x_2 \stackrel{def}{=} \min\{x \in \mathcal{F} \mid x \geqslant \max(\underline{\mathbf{x}}, x_1) \ \wedge\ \exists y \in \mathcal{F}.\ x \star y \in \mathbf{b}\}$. *If* $x_2 > \overline{\mathbf{x}}$ *or* $x_2 \star \underline{\mathbf{a}} > \overline{\mathbf{b}}$ *then* X *is empty, otherwise* $\min X = x_2$.

Proof Immediate using that for any $x \in \mathcal{F}$, by monotony of $\star$, $x \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}$ is equivalent to $x \geqslant \min\{x' \in \mathcal{F} \mid x' \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}\}$. □

This transforms our problem about six numbers $\underline{\mathbf{x}}, \overline{\mathbf{x}}, \underline{\mathbf{a}}, \overline{\mathbf{a}}, \underline{\mathbf{b}}, \overline{\mathbf{b}}$ (finding the minimum of $X = \{x \in \mathbf{x} \mid \exists a \in \mathbf{a}.\ x \star a \in \mathbf{b}\}$) into two simpler problems:

- for any two numbers $\overline{\mathbf{a}}, \underline{\mathbf{b}} \in \mathcal{F}$, finding the minimum of $\{x \in \mathcal{F} \mid x \star \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}\}$;
- for any three numbers $x_3, \underline{\mathbf{b}}, \overline{\mathbf{b}} \in \mathcal{F}$, finding the minimum of $\{x \in \mathcal{F} \mid x \geqslant x_3 \ \wedge\ \exists y \in \mathcal{F}.\ x \star y \in \mathbf{b}\}$.

For FP addition, the first problem is handled in §4.2, the second one in §4.3.

As there is no asymmetric hypothesis (for example we have not assumed $\underline{\mathbf{b}} \geqslant 0$ here), this is easily applicable to the maximum of X by inversing the inequalities and swapping lower bounds with upper bounds.

4.2 For $a, b \in \mathcal{F}$, minimum of $\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$

The minimum of $\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$ can be computed easily by dichotomy, as $\oplus$ is monotone. It can also be computed using [8]. The conditions are indeed: the real function $x \mapsto x + a$ is strictly increasing, it has an exactly rounded implementation $x \mapsto x \oplus a$, and its real inverse function $x \mapsto x - a$ also has an exactly rounded implementation $x \mapsto x \ominus a$.

This method seems faster than dichotomy but requires changing rounding modes, reading a flag to know whether the result of a computation has needed to be rounded and, last but not least, it requires an additional bit of precision.

The method described here is also fast, and its implementation stays within the target set of floating point numbers. We do need to compute predecessors and successors, or equivalently ulps that can be done efficiently [14]. We prove that there are at most three possible candidates for the minimum of $\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$, all of them easy to compute, so we can simply take the smallest of them that verifies the inequality.

Theorem 2 *Let $a, b \in \mathcal{F}$ with $b > 0$ and let $x_{min} \stackrel{def}{=} \min\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$.*

– *If $b > 2^{e_{min}+1}$ and $|b \ominus a| < b \oslash 2$ then*

$$\begin{cases} x_{min} = (b \ominus a) \ominus (ulp^-(b) \oslash 2) & \text{if } b \text{ is even;} \\ x_{min} = succ\big(\ (b \ominus a) \ominus (ulp^-(b) \oslash 2)\ \big) & \text{if } b \text{ is odd.} \end{cases}$$

– *If $b \leqslant 2^{e_{min}+1}$ or $|b \ominus a| \geqslant b \oslash 2$ then*

$$x_{min} \in \{(b \ominus a)^-,\ b \ominus a,\ (b \ominus a)^+\}.$$

Proof Let $a, b \in \mathcal{F}$ with $b > 0$ and let $x_{min} \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$. From (0.7): $x_{min} = \min\{x \in \mathcal{F} \mid b - \frac{ulp^-(b)}{2} \underset{<}{\overset{even(b)?}{\leqslant}} x + a\}$, in particular x_{min} is in this set but ${x_{min}}^-$ is not, so

$$\begin{cases} {x_{min}}^- < b - a - \frac{ulp^-(b)}{2} \leqslant x_{min} & \text{if } b \text{ is even;} \\ {x_{min}}^- \leqslant b - a - \frac{ulp^-(b)}{2} < x_{min} & \text{if } b \text{ is odd.} \end{cases}$$

– If $b > 2^{e_{min}+1}$ and $|b \ominus a| < b \oslash 2$:
  We will show that $b - a - \frac{ulp^-(b)}{2} \in \mathcal{F}$. Combined with the inequalities above and the fact that there is no floating point number strictly between ${x_{min}}^-$ and x_{min}, this will mean that $b-a-\frac{ulp^-(b)}{2} = x_{min}$ if b is even, and $b-a-\frac{ulp^-(b)}{2} = {x_{min}}^-$ if b is odd. Moreover, we will show that $(b \ominus a) \ominus (ulp^-(b) \oslash 2) = b - a - \frac{ulp^-(b)}{2}$, hence $x_{min} = (b \ominus a) \ominus (ulp^-(b) \oslash 2)$ if b is even, and $x_{min} = succ\big(\ {x_{min}}^-\ \big) = succ\big(\ (b \ominus a) \ominus (ulp^-(b) \oslash 2)\ \big)$ if b is odd.
  From $|b \ominus a| < b \oslash 2$ we get $|b - a| < \frac{b}{2}$ since $\circ$ is monotone and compatible with $|.|$. This means $-\frac{b}{2} < a - b < \frac{b}{2}$ so $\frac{b}{2} < a < \frac{3}{2}b$. Then, from Sterbenz's property [15]: $b - a \in \mathcal{F}$ and $a - b \in \mathcal{F}$. Then, since we also have $\frac{b}{2} \in \mathcal{F}$ from $b > 2^{e_{min}-p+1}$, $a - b < \frac{b}{2}$ becomes $a - b \leqslant (\frac{b}{2})^- = \frac{b}{2} - ulp^-(\frac{b}{2})$, rewritten as $-\frac{b}{2} \leqslant b - a - ulp^-(\frac{b}{2})$. Together with $b - a - ulp^-(\frac{b}{2}) \leqslant b - a \leqslant |b - a| < \frac{b}{2}$, this means that $|b - a - ulp^-(\frac{b}{2})| \leqslant \frac{b}{2}$. Furthermore, from $0 < \frac{b}{2} < a$ and (0.2): $a \equiv 0[ulp^-(\frac{b}{2})]$, and similarly $b \equiv 0[ulp^-(\frac{b}{2})]$, so $b - a - ulp^-(\frac{b}{2}) \equiv 0[ulp^-(\frac{b}{2})]$. Using (0.5), we obtain $b - a - ulp^-(\frac{b}{2}) \in \mathcal{F}$.
  Moreover, $ulp^-(b/2) = ulp^-(b)/2 = ulp^-(b) \oslash 2$ from (0.11) with $b > 2^{e_{min}+1}$, so $(b \ominus a) \ominus (ulp^-(b) \oslash 2) = \circ(\circ(b - a) - ulp^-(\frac{b}{2})) = \circ(b - a - ulp^-(\frac{b}{2})) = b - a - \frac{ulp^-(b)}{2}$ since $b - a \in \mathcal{F}$ and $b - a - ulp^-(\frac{b}{2}) \in \mathcal{F}$. As explained above, this allows us to prove the first case of the theorem.
– If $|b \ominus a| \geqslant b \oslash 2$ or $b \leqslant 2^{e_{min}+1}$:
  To show that $x_{min} \in \{(b \ominus a)^-,\ b \ominus a,\ (b \ominus a)^+\}$, it is sufficient to prove $(b \ominus a)^- \leqslant x_{min} \leqslant (b \ominus a)^+$.
  We always have $b - a < (b \ominus a)^+$ (indeed, otherwise by monotony of $\circ$ we would get $b \ominus a \geqslant (b \ominus a)^+$), so $b - a - \frac{1}{2}ulp^-(b) < (b \ominus a)^+$. From the inequalities above, we obtain ${x_{min}}^- < (b \ominus a)^+$ (for b even as well as for b odd), which means $x_{min} \leqslant (b \ominus a)^+$.
  To prove $(b \ominus a)^- \leqslant x_{min}$, it is enough to show that $b - a - \frac{1}{2}ulp^-(b) > (b \ominus a)^{--}$. Indeed, from the inequalities above, we then obtain $(b \ominus a)^{--} <$

x_{min}, which means $(b \ominus a)^- \leqslant x_{min}$. Let us prove $b - a - \frac{1}{2}ulp^-(b) > (b \ominus a)^{--}$ in both cases of the assumption " $|b \ominus a| \geqslant b \oslash 2$ or $b \leqslant 2^{e_{min}+1}$ ".

- If $|b \ominus a| \geqslant b \oslash 2$ then $ulp^-(b \ominus a) \geqslant ulp^-(b \oslash 2) \geqslant \frac{1}{2}ulp^-(b)$. Then from $(b \ominus a) - \frac{1}{2}ulp^-(b \ominus a) \leqslant b - a$ (0.7), we get $b - a - \frac{1}{2}ulp^-(b) \geqslant (b \ominus a) - \frac{1}{2}ulp^-(b \ominus a) - \frac{1}{2}ulp^-(b) \geqslant (b \ominus a) - \frac{3}{2}ulp^-(b \ominus a)$. Also, $\frac{1}{2}ulp^-(b \ominus a) \leqslant ulp^-((b \ominus a)^-)$, so $b - a - \frac{1}{2}ulp^-(b) \geqslant (b \ominus a) - ulp^-(b \ominus a) - ulp^-((b \ominus a)^-) = (b \ominus a)^{--}$. Moreover, we notice that the inequality $\frac{1}{2}ulp^-(b \ominus a) \leqslant ulp^-((b \ominus a)^-)$ is actually strict unless $(b \ominus a)^-$ is a power of 2, in which case $b \ominus a$ is odd and the inequality $(b \ominus a) - ulp^-(b \ominus a)/2 \leqslant b - a$ is actually strict. In all cases, we actually obtain $b - a - \frac{1}{2}ulp^-(b) > (b \ominus a)^{--}$.
- If $b \leqslant 2^{e_{min}+1}$ then $ulp^-(b) = 2^{e_{min}-p+1} = \min\{ulp^-(z) \mid z \in \mathcal{F}\}$ so $ulp^-(b \ominus a) \geqslant ulp^-(b)$. Using 0.7 as above, $b - a - \frac{1}{2}ulp^-(b) \geqslant (b \ominus a) - \frac{1}{2}ulp^-(b \ominus a) - \frac{1}{2}ulp^-(b) \geqslant (b \ominus a) - ulp^-(b \ominus a) = (b \ominus a)^- > (b \ominus a)^{--}$.

As explained above, we have proven $(b \ominus a)^- \leqslant x_{min} \leqslant (b \ominus a)^+$, therefore $x_{min} \in \{(b \ominus a)^-,\ b \ominus a,\ (b \ominus a)^+\}$.

□

The previous theorem states that when $b \leqslant 2^{e_{min}+1}$ or $|b \ominus a| \geqslant b \oslash 2$, there are three possible values for $x_{min} \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$: $(b \ominus a)^-$, $b \ominus a$ or $(b \ominus a)^+$. Let us give examples that any of these three values is indeed possible.

If $a = \frac{1}{2}$ and $b = 1$, then $(b \ominus a)^-$ is equal to $(\frac{1}{2})^- = \frac{1}{2} - 2^{-54}$ and verifies $(b \ominus a)^- \oplus a = 1 \geqslant b$, while its own predecessor does not verify this: $(b \ominus a)^{--} \oplus a < b$, therefore $x_{min} = (b \ominus a)^-$. If $a = 2^{-54}$ and $b = 1$, then we also have $x_{min} = (b \ominus a)^-$.

If $a = \frac{1}{4}$ and $b = 1$, then $x_{min} = b \ominus a$. We also have this if $a = 2^{-53}$ and $b = 1$.

Finally, if $a = 2^{-53}$ and $b = 1^- = 1 - 2^{-53}$, then $x_{min} = (b \ominus a)^+$.

Once the possible values of x_{min} have been narrowed down to two or three numbers, it is easy to add each of them to a and compare the sum with b, then take the smallest candidate which meets the condition. This is done by algorithm XminPT in §5. We can also compute $\max\{x \in \mathcal{F} \mid x \oplus a \leqslant b\}$ as $(\min\{x \in \mathcal{F} \mid x \oplus a \geqslant b^+\})^-$.

4.3 Given x_0 and an interval $\mathbf{b}$, minimum of $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge\ \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}\}$

The goal of this subsection is to compute, for any $x_0 \in \mathcal{F}$ and $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ such that $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, the minimum of $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge\ \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}\}$, which we will here denote x_{min}.

Theorem 3 provides an exact and efficient computation of x_{min}. We build up to it with several lemmas, mainly aiming to tie whether a floating point number x verifies $\exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}$ to other properties. Let us give some insight about the organization of this section.

First of all, we prove that for each interval $\mathbf{b}$, there are bounds l_g and u_g outside of which no x can verify the property $\exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}$. Definition 1 builds these bounds, and carefully chosen numbers $g \in \mathbb{Z}$ and $b_g \in \mathbf{b}$ that appear in the bounds

themselves as well as later in a few proofs. Lemma 3 provides various properties about g, b_g, l_g and u_g for later use. Lemma 4 shows that $\exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}$ is false as soon as $x > u_g$, and Lemma 5 is the same for $x \leqslant l_g$. This already tells us that if $x_0 > u_g$, then $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge\ \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}\}$ is empty and $x_{min} = +\infty$ by convention. On the other hand, if $x_0 \leqslant l_g$, then $x_{min} = {l_g}^+$ because we also have $\exists y \in \mathcal{F}.\ {l_g}^+ \oplus y \in \mathbf{b}$ (this is shown in the proof of Theorem 3).

Then, we study what happens for $x_0 \in (l_g, u_g]$. Lemma 6 is an auxilliary lemma used in both following lemmas. Lemma 7 states that for such an x_0, if $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ then either $\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b}$ or $\exists y \in \mathcal{F}.\ {x_0}^+ \oplus y \in \mathbf{b}$, which means that either $x_{min} = x_0$ or $x_{min} = {x_0}^+$. Lemma 8 handles the case $\underline{\mathbf{b}} = \overline{\mathbf{b}} \stackrel{\text{def}}{=} b$, where x_{min} is either x_0 or a more complex expression $\min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(b - x_0) \geqslant b\}$. This is enough to always compute x_{min}, as it is easy to test whether $x_{min} = x_0$, that is whether $\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b}$: indeed this is equivalent to $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}}$ where $y_0 = \min\{y \in \mathcal{F} \mid x_0 \oplus y \geqslant \underline{\mathbf{b}}\}$, as explained in the proof of Theorem 3; moreover we can easily compute y_0 thanks to Theorem 2 of the previous subsection. Therefore, for $x_0 \in (l_g, u_g]$, if $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}}$ then $x_{min} = x_0$, otherwise $x_{min} = {x_0}^+$ if $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ or $x_{min} = \min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(\underline{\mathbf{b}} - x_0) \geqslant \underline{\mathbf{b}}\}$ if $\underline{\mathbf{b}} = \overline{\mathbf{b}}$.

Finally, Theorem 3 gathers these results into a computation of x_{min} covering all cases. Its proof is mostly a more detailed and formal version of the explanations above, with proofs of a few properties too small to have their own lemma, such as $\exists y \in \mathcal{F}.\ {l_g}^+ \oplus y \in \mathbf{b}$.

Definition 1 For $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ such that $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, we denote *ExpAndWitness*($\mathbf{b}$) the couple (g, b_g) and *gBounds*($\mathbf{b}$) the couple (l_g, u_g) defined as

$$\begin{aligned} g &= \max\{e \in \mathbb{Z} \mid \exists b \in \mathbf{b}.\ b = 0[2^e]\}, \\ b_g &= \max\{b \in \mathbf{b} \mid b = 0[2^g]\}, \\ l_g &= -2^{g+p}, \\ u_g &= (2^{g+p})^- \oplus b_g. \end{aligned}$$

The integer g is the exponent of the greatest power of 2 that divides at least an element of $\mathbf{b}$, and this element is b_g. This means b_g is the rounder element of $\mathbf{b}$, as in the one whose mantissa ends with the most zeroes. We have written its definition with a maximum because this makes it obviously well defined, but it is actually the only element of $\mathbf{b}$ divisible by 2^g. Indeed, if there were at least two elements of $\mathbf{b}$ divisible by 2^g, this interval would in particular contain two consecutive multiples of 2^g, of which at least one would then be divisible by 2^{g+1} which would contradict the definition of g.

For example, if $\mathbf{b}$ contains a power of 2, then b_g is the greatest power of 2 it contains and g is its exponent. If $\mathbf{b}$ is included in $(2^e, 2^{e+1})$ for some $e \in \mathbb{Z}$, and if it contains $2^e + 2^{e-1}$ then $b_g = 2^e + 2^{e-1}$ and $g = e - 1$. If it is included in $(2^e, 2^e + 2^{e-1})$ and contains $2^e + 2^{e-2}$ then this is b_g and $g = e - 2$. And so on, which suggests how we can compute b_g and thus g by dichotomy.

The floating point numbers l_g and u_g are the bounds outside of which no x can verify $\exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}$. Equivalently: $(\exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}) \Rightarrow l_g < x \leqslant u_g$. These bounds may seem a little arbitrary, but they actually delimit the x for which either the ulp of x or the ulp of $(b_g - x)$ is less than or equal to 2^g.

The numbers b_g, l_g and u_g can also be found in [7]. Although the definitions look very different, b_g is actually equal to ζ from Section 3.2, and l_g and u_g are

respectively $-(\alpha^+)$ and β from Proposition 1. The minus sign comes from the problem being presented with $\oplus$ here instead of $\ominus$ there, and we consider the successor of α because, in order to have simpler expressions, we used one strict and one loose inequality in the property $(\exists y \in \mathcal{F}.\, x \oplus y \in \mathbf{b}) \Rightarrow l_g < x \leqslant u_g$ instead of two loose ones. This article has already shown that there is no solution outside of the bounds. We prove it again in Lemma 4 and Lemma 5 for the sake of completeness. Moreover, these proofs are shorter than the following ones so they may help the reader get used to notations and proof methods.

Lemma 3 *For all* $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ *such that* $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, *let* $(g, b_g) = \mathit{ExpAndWitness}(\mathbf{b})$, *then*

$$\forall b \in \mathbf{b}.\ b \not\equiv 0[2^{g+1}] \quad (2.1)$$

$$b_g \equiv 2^g[2^{g+1}] \quad (2.2)$$

$$g \geqslant e_{min} - p + 1,\ \mathit{ulp}^-(2^{g+p}) = 2^g,\ \mathit{ulp}^+(2^{g+p}) = \mathit{ulp}^-(2^{g+p+1}) = 2^{g+1} \quad (2.3)$$

$$b_g < 2^{g+p} \quad (2.4)$$

$$\overline{\mathbf{b}} < b_g + 2^g = b_g \oplus 2^g \quad (2.5)$$

$$(2^{g+p})^- \oplus b_g = (\underbrace{2^{g+p} + (b_g \oplus 2^g)}_{\in \mathcal{F}})^- \quad (2.6)$$

$$(2^{g+p})^- \oplus b_g = (2^{g+p})^- + b_g < 2^{g+p} + b_g < 2^{g+p+1} \quad (2.7)$$

$$\underline{\mathbf{b}} < \overline{\mathbf{b}} \quad \Rightarrow \quad \mathit{even}(b_g) \quad (2.8)$$

$$b_g - {l_g}^+ \in \mathcal{F} \quad (2.9)$$

Proof (2.1) is immediate from the definition of g.

(2.2) follows from $b_g \not\equiv 0[2^{g+1}]$ and $b_g \equiv 0[2^g]$.

(2.3): b_g is a positive finite floating point number so $b_g \equiv 0[2^{e_{min}-p+1}]$, but $b_g \not\equiv 0[2^{g+1}]$ so $g + 1 > e_{min} - p + 1$. Then 2^{g+p} and 2^{g+p+1} are **normal** floating point numbers hence the values for the ulps.

(2.4): $b_g \not\equiv 0[2^{g+1}]$ so $|b_g| < 2^{g+p}$ from (0.3).

(2.5): we have just seen that $b_g < 2^{g+p}$ so $b_g \leqslant (2^{g+p})^- = 2^{g+p} - 2^g$. Then $|b_g + 2^g| \leqslant 2^{g+p}$ with $b_g + 2^g \equiv 0[2^g]$ so $b_g + 2^g \in \mathcal{F}$ from (0.5), and $b_g \oplus 2^g = b_g + 2^g$. But $b_g + 2^g = 0[2^{g+1}]$ so $b_g + 2^g \notin \mathbf{b}$, and $b_g + 2^g > b_g \geqslant \underline{\mathbf{b}}$ so $b_g + 2^g > \overline{\mathbf{b}}$.

(2.6) and (2.7): from above, $b_g < 2^{g+p}$ so $(2^{g+p} - 2^g) + b_g < 2^{g+p} + b_g + 2^g < 2^{g+p+1} + 2^g$. Moreover, from $b_g \equiv 2^g[2^{g+1}]$ we get $(2^{g+p} - 2^g) + b_g \equiv 2^{g+p} + b_g + 2^g \equiv 0[2^{g+1}]$ so actually $0 < (2^{g+p} - 2^g) + b_g < 2^{g+p} + b_g + 2^g \leqslant 2^{g+p+1}$, and from (0.5) we obtain $(2^{g+p})^- + b_g = (2^{g+p} - 2^g) + b_g \in \mathcal{F}$ and $2^{g+p} + b_g + 2^g \in \mathcal{F}$. This means that $(2^{g+p})^- \oplus b_g = \circ((2^{g+p})^- + b_g) = (2^{g+p})^- + b_g$ and, using (2.5), $2^{g+p} + (b_g \oplus 2^g) = 2^{g+p} + b_g + 2^g \in \mathcal{F}$. Also, $2^{g+p} < 2^{g+p} + (b_g \oplus 2^g) \leqslant 2^{g+p+1}$ so $\mathit{ulp}^-(2^{g+p} + (b_g \oplus 2^g)) = \mathit{ulp}^+(2^{g+p}) = 2^{g+1}$ so $\mathit{pred}\big(\ (2^{g+p} + (b_g \oplus 2^g))\ \big) = (2^{g+p} + b_g + 2^g) - \mathit{ulp}^-(2^{g+p} + (b_g \oplus 2^g)) = 2^{g+p} + b_g - 2^g = (2^{g+p})^- \oplus b_g$. Finally, $(2^{g+p})^- + b_g < 2^{g+p} + b_g < 2^{g+p+1}$ is trivial using (2.4) again.

(2.8): by definition, g is the greatest integer such that $\mathbf{b}$ contains an element divisible by 2^g, and b_g is a witness for g, verifying $b_g \in \mathbf{b}$ and $b_g \equiv 0[2^g]$. This means that b_g is divisible by every power of two that divides at least an element of $\mathbf{b}$. For any odd floating point number x, both its neighbors (predecessor and

successor) are divisible by a greater power of two than x. If $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ then any element of $\mathbf{b}$ has at least one neighbor in $\mathbf{b}$, in particular for any odd element of $\mathbf{b}$ there is another element divisible by a greater power of two, therefore b_g cannot be odd.

(2.9): we have shown that $ulp^+(-2^{g+p}) = ulp^-(2^{g+p}) = 2^g$ (2.3) so ${l_g}^+ = (-2^{g+p})^+ = -2^{g+p}+2^g$. Then, using (2.2), $b_g - {l_g}^+ \equiv 2^g + 2^{g+p} - 2^g \equiv 0[2^{g+1}]$ with $-2^{g+p+1} < b_g - {l_g}^+ = b_g + 2^{g+p} - 2^g < 2^{g+p+1}$ (2.7) where $ulp^-(2^{g+p+1}) = 2^{g+1}$, therefore $b_g - {l_g}^+ \in \mathcal{F}$ from (0.5).

□

Lemma 4 *For all $x \in \mathcal{F}$ and $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, if $x > (2^{g+p})^- \oplus b_g$ then $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\} = \emptyset$ where $(g, b_g) = ExpAndWitness(\mathbf{b})$.*

Proof Let $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, let $(g, b_g) = ExpAndWitness(\mathbf{b})$, and let $x \in \mathcal{F}$ such that $x > (2^{g+p})^- \oplus b_g$. This proof proceeds by contradiction: assume that $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\}$ contains at least one element y. Then $x \oplus y \leqslant \overline{\mathbf{b}} < b_g \oplus 2^g$ using (2.5), so $x + y < b_g \oplus 2^g$. From (2.6): $x \geqslant 2^{g+p} + (b_g \oplus 2^g)$, so $y < (b_g \oplus 2^g) - x \leqslant -2^{g+p}$. Also $x \geqslant 2^{g+p}$, so from (0.3): $x \equiv y \equiv 0[2^{g+1}]$. Then $x + y \equiv 0[2^{g+1}]$, so using (0.6): $x \oplus y \equiv 0[2^{g+1}]$, whereas $x \oplus y \in \mathbf{b}$ and $\forall b \in \mathbf{b}.\, b \neq 0[2^{g+1}]$ (2.1). By contradiction, we obtain $\{a \in \mathcal{F} \mid x \oplus a \in \mathbf{b}\} = \emptyset$. □

Lemma 5 *For all $x \in \mathcal{F}$ and $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, if $x \leqslant -2^{g+p}$ then $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\} = \emptyset$ where $(g, b_g) = ExpAndWitness(\mathbf{b})$.*

Proof Similar to the proof of Lemma 4. Let $\mathbf{b}, g, b_g$ as above and let $x \in \mathcal{F}$ such that $x \leqslant -2^{g+p}$. By contradiction, assume that $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\}$ contains at least one element y. Then $y \geqslant -x \geqslant 2^{g+p}$ from $x \oplus y \geqslant \underline{\mathbf{b}} > 0$, so $x \equiv y \equiv 0[2^{g+1}]$ so $x \oplus y = 0[2^{g+1}]$ which contradicts (2.1). □

In the following lemmas, we use the notation $Y_{x,b} \stackrel{\text{def}}{=} \{y \in \mathcal{F} \mid x \oplus y = b\}$.

Note that $Y_{x,b} \neq \emptyset$ is equivalent to $\exists y \in \mathcal{F}.\, x \oplus y = b$. So when the interval $\mathbf{b}$ is reduced to the singleton $\{b\}$, that is when $\underline{\mathbf{b}} = \overline{\mathbf{b}} = b$, the set whose minimum we are looking for, $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ \exists y \in \mathcal{F}.\, x \oplus y \in \mathbf{b}\}$, is equal to $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ Y_{x,b} \neq \emptyset\}$. And even when $\mathbf{b}$ is not a singleton, the set $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ \exists y \in \mathcal{F}.\, x \oplus y \in \mathbf{b}\}$ can be written as $\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ \exists b \in \mathbf{b}.\, Y_{x,b} \neq \emptyset\}$.

Lemma 6 *Let $x_0, b \in \mathcal{F}$, $g \in \mathbb{Z}$ such that $b > 0$, $b \equiv 2^g[2^{g+1}]$ and $-2^{g+p} < x_0 \leqslant (2^{g+p})^- \oplus b$. Let $v \stackrel{def}{=} ulp^+(b - x_0)$. If $Y_{x_0,b} = \emptyset$ and $Y_{{x_0}^+,b} = \emptyset$ then $ulp^+(x_0) \leqslant \frac{v}{4}$ and $ulp^-(b) = \frac{v}{2}$ (and $\frac{v}{4} \in \mathcal{F}$).*

Proof Let $x_0, b \in \mathcal{F}$, $g \in \mathbb{Z}$, $v \stackrel{\text{def}}{=} ulp^+(b - x_0)$, and assume $b > 0$, $b \equiv 2^g[2^{g+1}]$, $-2^{g+p} < x_0 \leqslant (2^{g+p})^- \oplus b$, $Y_{x_0,b} = \emptyset$ and $Y_{{x_0}^+,b} = \emptyset$.

First of all, we notice that $b - x_0 \notin \mathcal{F}$. Indeed, if we had $b - x_0 \in \mathcal{F}$, we would have $x_0 \oplus (b - x_0) = \circ(b) = b$ so $b - x_0 \in Y_{x_0,b} \stackrel{\text{def}}{=} \{y \in \mathcal{F} \mid x_0 \oplus y = b\}$, which contradicts the assumption $Y_{x_0,b} = \emptyset$. Therefore, $b - x_0 \notin \mathcal{F}$.

By contradiction, assume $ulp^+(x_0) \geqslant v$.

Then v divides $ulp^+(x_0)$ as both are powers of 2, so from $x_0 \equiv 0[ulp^+(x_0)]$, we get $x_0 \equiv 0[v]$.

By assumption and by (2.7): $x_0 \leqslant (2^{g+p})^- \oplus b < 2^{g+p}+b$, so $-2^{g+p} < b-x_0$. If $|x_0| \geqslant 2^{g+p}$ then, since we assumed -2^{g+p}, we get $x_0 \geqslant 2^{g+p} > b$ (2.4), so $-2^{g+p} < b - x_0 < 0$ so $|b - x_0| < 2^{g+p}$. This means $|x_0| \geqslant 2^{g+p} \Rightarrow |b - x_0| < 2^{g+p}$, which we rewrite as $|x_0| < 2^{g+p} \vee |b - x_0| < 2^{g+p}$. From (0.1) and (2.3): $\forall y \in \mathcal{F}$, $|y| < 2^{g+p} \Rightarrow ulp^+(y) \leqslant ulp^-(2^{g+p}) = 2^g$. Then $ulp^+(x_0) \leqslant 2^g \vee ulp^+(b - x_0) \leqslant 2^g$. But $ulp^+(x_0) \geqslant v$ and $ulp^+(b - x_0) \stackrel{\text{def}}{=} v$, therefore $v \leqslant 2^g$. Since v and 2^g are powers of 2, this means v divides 2^g; from the assumption $b \equiv 2^g[2^{g+1}]$ we get $b \equiv 0[2^g]$ then $b \equiv 0[v]$.

We have shown $x_0 \equiv 0 \equiv b[v]$. This means $b-x_0 \equiv 0[ulp^+(b-x_0)]$, so $b-x_0 \in \mathcal{F}$ from (0.4), whereas we have shown above that $b-x_0 \notin \mathcal{F}$. Therefore, $ulp^+(x_0) < v$.

By contradiction, assume $ulp^+(x_0) = \frac{v}{2}$.

Then from $x_0 \equiv 0[ulp^+(x_0)]$, we get $x_0 \equiv 0[\frac{v}{2}]$.

By assumption and by (2.7): $-2^{g+p} < x_0 \leqslant (2^{g+p})^- \oplus b < 2^{g+p} + b$, so $-2^{g+p} < b-x_0 < b+2^{g+p} < 2^{g+p+1}$, so $v \stackrel{\text{def}}{=} ulp^+(b-x_0) \leqslant ulp^-(2^{g+p+1}) = 2^{g+1}$ by (0.1) and (2.3). Since v is a power of 2, this means $\frac{v}{2}$ divides 2^g, moreover $b \equiv 0[2^g]$ so $b \equiv 0[\frac{v}{2}]$.

We have shown $x_0 \equiv 0 \equiv b[\frac{v}{2}]$, so there are two possible cases: either $b \equiv x_0[v]$ or $b \equiv x_0 + \frac{v}{2}[v]$. In the first case $b - x_0 \equiv 0[v]$, which leads to the contradiction $b - x_0 \in \mathcal{F}$ as above. In the second case, $b - x_0 \equiv \frac{v}{2}[v]$ with $v = ulp^+(b - x_0)$, so the nearest floating point numbers on either side of $b - x_0$ are $b - x_0 - \frac{v}{2}$ and $b - x_0 + \frac{v}{2}$ (0.9); in particular $b - {x_0}^+ = b - (x_0 + ulp^+(x_0)) = b - x_0 - \frac{v}{2} \in \mathcal{F}$ with ${x_0}^+ \oplus (b - {x_0}^+) = \circ(b) = b$, so $b - {x_0}^+ \in Y_{{x_0}^+,b}$ which contradicts $Y_{{x_0}^+,b} = \emptyset$. Therefore, $ulp^+(x_0) \neq \frac{v}{2}$.

We have proven $ulp^+(x_0) < v$ and $ulp^+(x_0) \neq \frac{v}{2}$. Since $ulp^+(x_0)$ and v are both powers of 2, this means that $ulp^+(x_0) \leqslant \frac{v}{4}$.

This gives us $|x_0| \leqslant \frac{1}{2}|b - x_0|$ using (0.10) with $v \stackrel{\text{def}}{=} ulp^+(b - x_0)$. Then, since $b > 0$, $b = |b-x_0+x_0| \geqslant |b-x_0|-|x_0| \geqslant |b-x_0|-\frac{1}{2}|b-x_0| = \frac{1}{2}|b-x_0|$. Moreover, we already know that $b-x_0 \notin \mathcal{F}$ so $\frac{1}{2}|b-x_0| \notin \mathcal{F}$ while $b \in \mathcal{F}$, so the inequality is strict: so $b > \frac{1}{2}|b-x_0|$. Then, using (0.1), $ulp^-(b) \geqslant ulp^+(\frac{1}{2}(b-x_0)) \geqslant \frac{1}{2}ulp^+(b-x_0) = \frac{v}{2}$.

By contradiction, assume $\frac{1}{2}(ulp^-(b) + ulp^+(b)) \geqslant v$. From Lemma 2, there is a floating point number y in $\left[\, b - x_0 - \frac{1}{2}ulp^-(b), b - x_0 + \frac{1}{2}ulp^+(b) \,\right)$, and y verifies $b - \frac{1}{2}ulp^-(b) \leqslant x_0 + y < b + \frac{1}{2}ulp^+(b)$. If $b - \frac{1}{2}ulp^-(b) < x_0 + y < b + \frac{1}{2}ulp^+(b)$ then $x_0 \oplus y = b$ from (0.7) so $y \in Y_{x_0,b}$, which contradicts $Y_{x_0,b} = \emptyset$. Otherwise $x_0 + y = b - \frac{1}{2}ulp^-(b)$, moreover we know that $ulp^+(b) \geqslant \frac{v}{2} \geqslant 2ulp^+(x_0)$, so ${x_0}^+ + y = x_0 + ulp^+(x_0) + y = b - \frac{1}{2}ulp^-(b) + ulp^+(x_0) < b + ulp^+(x_0) \leqslant b + \frac{1}{2}ulp^+(b)$, so $b - \frac{1}{2}ulp^-(b) = x_0 + y < {x_0}^+ + y < b + \frac{1}{2}ulp^+(b)$; then ${x_0}^+ \oplus y = b$ from (0.7) so $y \in Y_{{x_0}^+,b}$, which contradicts $Y_{{x_0}^+,b} = \emptyset$. Therefore, $\frac{1}{2}(ulp^-(b) + ulp^+(b)) < v$.

$b > 0$ so $ulp^-(b) \leqslant ulp^+(b)$, so from above we get $ulp^-(b) < v$. We have also proven $ulp^-(b) \geqslant \frac{v}{2}$, and both are powers of 2, therefore $ulp^-(b) = \frac{v}{2}$.

□

Lemma 7 *Let $x_0 \in \mathcal{F}$ and $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, and let $(l_g, u_g) = gBounds(\mathbf{b})$.*

If $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ *and* $l_g < x_0 \leqslant u_g$ *then*

$$\exists y \in \mathcal{F},\, x_0 \oplus y \in \mathbf{b} \quad \vee \quad \exists y \in \mathcal{F},\, {x_0}^+ \oplus y \in \mathbf{b}.$$

Proof Usual notations including (g, b_g), same assumptions as lemma. By contradiction, assume $\{y \in \mathcal{F} \mid x_0 \oplus y \in \mathbf{b}\} = \emptyset$ and $\{y \in \mathcal{F} \mid {x_0}^+ \oplus y \in \mathbf{b}\} = \emptyset$.

Since $b_g \in \mathbf{b}$, we have $Y_{x_0,b_g} \stackrel{\text{def}}{=} \{y \in \mathcal{F} \mid x_0 \oplus y = b_g\} \subset \{y \in \mathcal{F} \mid x_0 \oplus y \in \mathbf{b}\} = \emptyset$ so $Y_{x_0,b_g} = \emptyset$, and similarly $Y_{{x_0}^+,b_g} = \emptyset$. Then, from Lemma 6 we get $ulp^+(x_0) \leqslant \frac{v}{4}$ and $ulp^-(b_g) = \frac{v}{2}$ where $v \stackrel{\text{def}}{=} ulp^+(b_g - x_0)$. Since $b_g \geqslant 0$, we also have $ulp^+(b_g) \geqslant ulp^-(b_g) = \frac{v}{2}$.

Let $y_d \stackrel{\text{def}}{=} \bigtriangledown(b_g - x_0)$ and $y_u \stackrel{\text{def}}{=} \bigtriangleup(b_g - x_0)$. We are going to prove that $x_0 \oplus y_u \in [b_g, {b_g}^+]$ and $x_0 \oplus y_d \in [{b_g}^-, b_g]$.

We have $y_d \leqslant b_g - x_0 \leqslant y_u$ so $x_0 \oplus y_d \leqslant b_g$ and $x_0 \oplus y_u \geqslant b_g$, but $\{y \in \mathcal{F} \mid x_0 \oplus y = b_g\} = \emptyset$ so $x_0 \oplus y_d < b_g$ and $x_0 \oplus y_u > b_g$. Then, from (0.7): $x_0 + y_d \leqslant b_g - \frac{1}{2}ulp^-(b_g) = b_g - \frac{v}{4}$, and $x_0 + y_u \geqslant b_g + \frac{1}{2}ulp^+(b_g) \geqslant b_g + \frac{v}{4}$.

We have $b_g - x_0 \notin \mathcal{F}$, because otherwise we would have $x_0 \oplus (b_g - x_0) = b_g$ hence the contradiction $b_g - x_0 \in \{y \in \mathcal{F} \mid x_0 \oplus y \in \mathbf{b}\} = \emptyset$. This means that $y_d < b_g - x_0 < y_u$ and $y_u = y_d + ulp^+(b_g - x_0) = y_d + v$.

On the one hand, we obtain $x_0 + y_u = x_0 + y_d + v \leqslant b_g - \frac{v}{4} + v$. But we already know $ulp^+(b_g) \geqslant \frac{v}{2}$, and since $b_g \geqslant 0$ we have $ulp^+(b_g) \leqslant ulp^+({b_g}^+)$, so $b_g + \frac{3}{4}v \leqslant b_g + ulp^+(b_g) + \frac{1}{2}ulp^+(b_g) = {b_g}^+ + \frac{1}{2}ulp^+(b_g) \leqslant {b_g}^+ + \frac{1}{2}ulp^+({b_g}^+)$. Therefore $x_0 + y_u \leqslant {b_g}^+ + \frac{1}{2}ulp^+({b_g}^+)$. Moreover, from (2.8) b_g is even so ${b_g}^+$ is odd, and we have $b_g \leqslant x_0 + y_u \leqslant {b_g}^+ + \frac{1}{2}ulp^+({b_g}^+)$, so from (0.7) we obtain $x_0 \oplus y_u \in [b_g, {b_g}^+]$.

On the other hand, we get $x_0 + y_d = x_0 + y_u - v \geqslant b_g + \frac{v}{4} - v = b_g - \frac{v}{2} - \frac{v}{4} = b_g - ulp^-(b_g) - \frac{1}{2}ulp^-(b_g) = {b_g}^- - \frac{1}{2}ulp^+({b_g}^-)$. Since b_g is even, ${b_g}^-$ is odd which implies $ulp^-(b_g) = ulp^+(b_g)$. Then ${b_g}^- - \frac{1}{2}ulp^-({b_g}^-) \leqslant x_0 + y_d \leqslant b_g$ where ${b_g}^-$ is odd, so from (0.7) we obtain $x_0 \oplus y_d \in [{b_g}^-, b_g]$.

Finally, $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ means that $\mathbf{b}$ contains at least two elements; moreover it contains b_g. Then, either ${b_g}^+ \in \mathbf{b}$ or ${b_g}^- \in \mathbf{b}$, so either $x_0 \oplus y_u \in \mathbf{b}$ or $x_0 \oplus y_d \in \mathbf{b}$, which contradicts our assumption $\{y \in \mathcal{F} \mid x_0 \oplus y \in \mathbf{b}\} = \emptyset$.

Therefore, $\{y \in \mathcal{F} \mid x_0 \oplus y \in \mathbf{b}\} \neq \emptyset$ or $\{y \in \mathcal{F} \mid {x_0}^+ \oplus y \in \mathbf{b}\} \neq \emptyset$.

(Note: in most of the proof ${x_0}^+$ does not appear, however we did use the assumption $\{y \in \mathcal{F} \mid {x_0}^+ \oplus y \in \mathbf{b}\} = \emptyset$ to be able to apply Lemma 6.)

□

Lemma 8 *Let* $x_0, b \in \mathcal{F}$, $g \in \mathbb{Z}$ *such that* $b > 0$, $b \equiv 2^g[2^{g+1}]$ *and* $-2^{g+p} < x_0 \leqslant (2^{g+p})^- \oplus b$. *Let* $x_{min} \stackrel{def}{=} \min\{x \in \mathcal{F} \mid x \geqslant x_0 \wedge Y_{x,b} \neq \emptyset\}$. *Then* x_{min} *is either* x_0 *or* $\min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(b - x_0) \geqslant b\}$.

Proof Same notations, assumptions and définition of x_{min}. Let $y_d \stackrel{\text{def}}{=} \bigtriangledown(b - x_0)$ and let $x_1 \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \oplus y_d \geqslant b\}$. The lemma states that $x_{min} \in \{x_0, x_1\}$.

Let us assume that $x_{min} \neq x_0$, and let us prove that $x_{min} = x_1$.

By definition of x_{min}, $x_{min} \neq x_0$ means that $Y_{x_0,b} = \emptyset$. By definition of $\bigtriangledown$, $y_d \leqslant b - x_0 < {y_d}^+$, in particular $x_0 + y_d \leqslant b$ so $x_0 \oplus y_d \leqslant \circ(b) = b$. But we just

saw that $Y_{x_0,b} \stackrel{\text{def}}{=} \{y \in \mathcal{F} \mid x_0 \oplus y = b\} = \emptyset$ so $x_0 \oplus y_d \neq b$, therefore $x_0 \oplus y_d < b$. Similarly, $x_0 \oplus {y_d}^+ \geqslant b$ but $x_0 \oplus {y_d}^+ \neq b$ so $x_0 \oplus {y_d}^+ > b$.

To prove that $x_1 = x_{min} \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \geqslant x_0 \wedge Y_{x,b} \neq \emptyset\}$, we will first prove that $x_1 \geqslant x_0$, then that $Y_{x_1,b} \neq \emptyset$, which will mean that $x_1 \in \{x \in \mathcal{F} \mid x \geqslant x_0 \wedge Y_{x,b} \neq \emptyset\}$. Then, to show that x_1 is actually the minimum of this set, it will be sufficient to prove that $Y_{x,b} = \emptyset$ for any $x \in \mathcal{F}$ such that $x_0 \leqslant x < x_1$.

By definition of x_1, $x_1 \oplus y_d \geqslant b$, and we have seen that $x_0 \oplus y_d < b$ so $x_1 > x_0$.

We prove $Y_{x_1,b} \neq \emptyset$ by case analysis:

- If $Y_{{x_0}^+,b} \neq \emptyset$, then let $y \in Y_{{x_0}^+,b}$ such that ${x_0}^+ \oplus y = b$. We have $y \leqslant y_d$, indeed otherwise $y' \geqslant {y_d}^+$ so ${x_0}^+ \oplus y \geqslant x_0 \oplus {y_d}^+ > b$. Then ${x_0}^+ \oplus y_d \geqslant {x_0}^+ \oplus y = b$ while $x_0 \oplus y_d < b$, so ${x_0}^+ = \min\{x \in \mathcal{F} \mid x \oplus y_d \geqslant b\} \stackrel{\text{def}}{=} x_1$, so $Y_{{x_0}^+,b} \neq \emptyset$ means $Y_{x_1,b} \neq \emptyset$.
- If $Y_{{x_0}^+,b} = \emptyset$, then from Lemma 6 we get $ulp^+(x_0) \leqslant \frac{v}{4}$ and $ulp^-(b) = \frac{v}{2}$ where $v \stackrel{\text{def}}{=} ulp^+(b - x_0)$. (Since $b \geqslant 0$, we also have $ulp^+(b) \geqslant ulp^-(b) = \frac{v}{2}$.)
  We will prove that $x_1 + y_d \leqslant b$. This will give $x_1 \oplus y_d \leqslant b$, while also $x_1 \oplus y_d \geqslant b$ by definition of x_1, so $x_1 \oplus y_d = b$ which means $y_d \in Y_{x_1,b}$.
  We already know that $x_0 \oplus {y_d}^+ > b$, and from (0.8) $ulp^+(y_d) = ulp^+(b - x_0) = v$, so $\circ(x_0 + y_d + v) > b$. Then, $\triangle(x_0 + v) \oplus y_d = \circ(\triangle(x_0 + v) + y_d) \geqslant \circ(x_0 + v + y_d) > b$, so by definition of x_1, $x_1 \leqslant \triangle(x_0 + v)$. Then ${x_1}^- \leqslant pred\big(\ \triangle(x_0 + v)\ \big) \leqslant x_0 + v$. As we already have $ulp^+(x_0) \leqslant \frac{v}{4}$ and $x_1 > x_0$ which means ${x_1}^- \geqslant x_0$, from Lemma 1 we obtain $ulp^+({x_1}^-) \leqslant \frac{v}{2}$.
  By definition of x_1, we know that ${x_1}^- \oplus y_d < b$, so ${x_1}^- + y_d \leqslant b - \frac{v}{4}$ by (0.7) with $ulp^-(b) = \frac{v}{2}$.
  If $ulp^+({x_1}^-) < \frac{v}{2}$, then $ulp^+({x_1}^-) \leqslant \frac{v}{4}$ as they are all powers of 2, so $x_1 + y_d = {x_1}^- + ulp^+({x_1}^-) + y_d \leqslant {x_1}^- + y_d + \frac{v}{4} \leqslant b - \frac{v}{4} + \frac{v}{4} = b$.
  If $ulp^+({x_1}^-) = \frac{v}{2}$ then $x_1 + y_d = {x_1}^- + ulp^+({x_1}^-) + y_d = {x_1}^- + y_d + \frac{v}{2} \leqslant b + \frac{v}{4}$. Also $ulp^-(x_1) = ulp^+({x_1}^-) = \frac{v}{2}$, moreover we already have $ulp^-(b) = \frac{v}{2}$ and $ulp^+(y_d) = v$, so $x_1 \equiv b \equiv y_d \equiv 0[\frac{v}{2}]$. Then $x_1 + y_d \leqslant b + \frac{v}{4}$ with $x_1 + y_d \equiv b[\frac{v}{2}]$, therefore $x_1 + y_d \leqslant b$ (as there is no number equal to b modulo $\frac{v}{2}$ in the real interval $(b, b + \frac{v}{4}]$).
  In both cases we obtain $x_1 + y_d \leqslant b$. As explained above, using the definition of b we get $x_1 \oplus y_d = b$, which means $y_d \in Y_{x_1,b}$ hence $Y_{x_1,b} \neq \emptyset$.

Finally, let $x \in \mathcal{F}$ such that $x_0 \leqslant x < x_1$, and let us prove that $Y_{x,b} = \emptyset$. On the one hand, by definition of x_1, $x < x_1$ means that $x \notin \{x' \in \mathcal{F} \mid x' \oplus y_d \geqslant b\}$ so $x \oplus y_d < b$. On the other hand, $x \oplus {y_d}^+ \geqslant x_0 \oplus {y_d}^+$ and we have already shown $x_0 \oplus {y_d}^+ > b$, so $x \oplus {y_d}^+ > b$. Moreover, as consecutive floating point numbers, y_d and ${y_d}^+$ satisfy: for all $y \in \mathcal{F}$, either $y \leqslant y_d$ or $y \geqslant {y_d}^+$. Then, for all $y \in \mathcal{F}$, either $x \oplus y \leqslant x \oplus y_d < b$ or $x \oplus y \geqslant x \oplus {y_d}^+ > b$ so in any case $x \oplus y \neq b$. Therefore $Y_{x,b} = \emptyset$.

Having assumed $x_{min} \neq x_0$, we have proven $x_1 \geqslant x_0$ and $Y_{x_1,b} \neq \emptyset$ and $(\forall x \in \mathcal{F},\ x_0 \leqslant x < x_1 \Rightarrow Y_{x,b} = \emptyset)$, which means that $x_{min} = x_1$ as explained above. Therefore, $x_{min} \in \{x_0, x_1\}$ where $x_1 \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \oplus y_d \geqslant b\}$. □

We can finally prove the main theorem of this section. Algorithm XminITV in §5 relies on it.

Theorem 3 *Let $x_0 \in \mathcal{F}$ and $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, and let $(l_g, u_g) = gBounds(\mathbf{b})$.*

We define $x_{min} \stackrel{def}{=} \min\{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}\}$.

- *If $x_0 \leqslant l_g$ then $x_{min} = {l_g}^+$.*
- *If $x_0 > u_g$ then $x_{min} = +\infty$ (by convention, minimum of an empty set).*

For the remaining cases, we introduce $y_0 \stackrel{def}{=} \min\{y \in \mathcal{F} \mid x_0 \oplus y \geqslant \underline{\mathbf{b}}\}$.

- *If $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}}$ then $x_{min} = x_0$.*
- *If $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 > \overline{\mathbf{b}}$ and $\underline{\mathbf{b}} < \overline{\mathbf{b}}$ then $x_{min} = {x_0}^+$.*
- *If $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 > \overline{\mathbf{b}}$ and $\underline{\mathbf{b}} = \overline{\mathbf{b}}$ then $x_{min} = \min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(\underline{\mathbf{b}} - x_0) \geqslant \underline{\mathbf{b}}\}$.*

Proof Same notations and definitions.

Let $E \stackrel{\text{def}}{=} \{x \in \mathcal{F} \mid x \geqslant x_0 \ \wedge \ \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}\}$ and $x_{min} = \min E$.

- Case $x_0 \leqslant l_g$. From Lemma 5, for all $x \in \mathcal{F}$ such that $x \leqslant l_g$, $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\} = \emptyset$ so $x \notin E$, therefore $x_{min} > l_g$. On the other hand, from (2.9) $b_g - {l_g}^+ \in \mathcal{F}$ which means ${l_g}^+ \oplus (b_g - {l_g}^+) = b_g \in \mathbf{b}$ with ${l_g}^+ \geqslant x_0$ so ${l_g}^+ \in E$, therefore $x_{min} \leqslant {l_g}^+$. Combining both inequalities, we obtain $x_{min} = {l_g}^+$.
- Case $x_0 > u_g$. Then for all $x \in \mathcal{F}$ such that $x \geqslant x_0$, we get $x > u_g$, so from Lemma 4, $\{y \in \mathcal{F} \mid x \oplus y \in \mathbf{b}\} = \emptyset$. This means that E is empty, so $x_{min} = +\infty$ by convention.

Before handling the remaining cases, let $y_0 \stackrel{\text{def}}{=} \min\{y \in \mathcal{F} \mid x_0 \oplus y \geqslant \underline{\mathbf{b}}\}$ and let us show that $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}} \ \Leftrightarrow \ \exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b}$. On the one hand, $x_0 \oplus y_0 \geqslant \underline{\mathbf{b}}$ by definition of y_0 so: if $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}}$ then $x_0 \oplus y_0 \in [\underline{\mathbf{b}}, \overline{\mathbf{b}}] = \mathbf{b}$. On the other hand, assume there exists $y \in \mathcal{F}$ such that $x_0 \oplus y \in \mathbf{b}$, then $x_0 \oplus y \geqslant \underline{\mathbf{b}}$ so $y \geqslant y_0$ by definition of y_0, then by monotony $x_0 \oplus y_0 \leqslant x_0 \oplus y \leqslant \overline{\mathbf{b}}$.

- Case $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 \leqslant \overline{\mathbf{b}}$. As we have just seen, the second condition means that $\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b}$. Then $x_0 \in E$, and by definition of E, $x \geqslant x_0$ for any element $x \in E$, so $x_{min} = x_0$. (Note that we do not need the first condition $l_g < x_0 \leqslant u_g$, which is there to highlight the completeness of the cases.)
- Case $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 > \overline{\mathbf{b}}$ and $\underline{\mathbf{b}} < \overline{\mathbf{b}}$. As explained above and adding negations, the second condition means $\neg(\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b})$, so $x_0 \notin E$. Furthermore, using Lemma 7 (as allowed by the first and third conditions) we get: either $\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b}$ or $\exists y \in \mathcal{F}.\ {x_0}^+ \oplus y \in \mathbf{b}$, but we know the first statement to be false so the second one is true. Since ${x_0}^+ \geqslant x_0$, this means that ${x_0}^+ \in E$. Moreover, by definition of E, for any $x \in E$, we have $x \geqslant x_0$, but $x_0 \notin E$ so actually $x > x_0$ which means $x \geqslant {x_0}^+$. Therefore, $x_{min} = {x_0}^+$.
- Case $l_g < x_0 \leqslant u_g$ and $x_0 \oplus y_0 > \overline{\mathbf{b}}$ and $\underline{\mathbf{b}} = \overline{\mathbf{b}}$. Once again, the second condition means $\neg(\exists y \in \mathcal{F}.\ x_0 \oplus y \in \mathbf{b})$ so $x_0 \notin E$. Furthermore, the first and third conditions let us use Lemma 8 to obtain that x_{min} is either x_0 or $\min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(\underline{\mathbf{b}} - x_0) \geqslant \underline{\mathbf{b}}\}$ (indeed, for $b = \underline{\mathbf{b}} = \overline{\mathbf{b}}$ and any $x \in \mathcal{F}$,

$Y_{x,b} \neq \emptyset \Leftrightarrow \{y \in \mathcal{F} \mid x \oplus y = b\} \neq \emptyset \Leftrightarrow \exists y \in \mathcal{F}.\ x \oplus y \in \mathbf{b}$ so the definition of x_{min} in the lemma is the same as here). But $x_0 \notin E$ so $x_{min} \neq x_0$, therefore $x_{min} = \min\{x \in \mathcal{F} \mid x \oplus \bigtriangledown(\underline{\mathbf{b}} - x_0) \geqslant \underline{\mathbf{b}}\}$.

□

4.4 Refining B

We still denote $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$, $\mathbf{a} = [\underline{\mathbf{a}}, \overline{\mathbf{a}}]$, $\mathbf{b} = [\underline{\mathbf{b}}, \overline{\mathbf{b}}]$ nonempty floating point intervals such that $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, $\underline{\mathbf{x}} * \overline{\mathbf{x}} > 0$, $\underline{\mathbf{a}} * \overline{\mathbf{a}} > 0$. We now want to compute the minimum and maximum of the set B below, denoted b_{min} and b_{max} respectively.

$$B \stackrel{\text{def}}{=} \{b \in \mathbf{b} \mid \exists x \in \mathbf{x}.\ \exists a \in \mathbf{a}.\ x \oplus a = b\}.$$

As we already know how to compute the respective minima (x_{min} and a_{min}) and maxima (x_{max} and a_{max}) of the sets X and A defined below, we will use them in this section.

$$X \stackrel{\text{def}}{=} \{x \in \mathbf{x} \mid \exists a \in \mathbf{a}.\ x \oplus a \in \mathbf{b}\}$$
$$A \stackrel{\text{def}}{=} \{a \in \mathbf{a} \mid \exists x \in \mathbf{x}.\ x \oplus a \in \mathbf{b}\}$$

First of all, note that an element b belonging to B is based on witnesses x and a that are actually, by definition, in X and in A respectively. And since $X \subset [x_{min}, x_{max}]$ and $A \subset [a_{min}, a_{max}]$, we can rewrite B as below. We can do the same for X and A then forget $\underline{\mathbf{x}}$, $\overline{\mathbf{x}}$, $\underline{\mathbf{a}}$ and $\overline{\mathbf{a}}$.

$$B = \{b \in \mathbf{b} \mid \exists x \in [x_{min}, x_{max}].\ \exists a \in [a_{min}, a_{max}].\ x \oplus a = b\}$$
$$X = \{x \in [x_{min}, x_{max}] \mid \exists a \in [a_{min}, a_{max}].\ x \oplus a \in \mathbf{b}\}$$

Theorem 4 *Let $\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$, $\mathbf{a} = [\underline{\mathbf{a}}, \overline{\mathbf{a}}]$, $\mathbf{b} = [\underline{\mathbf{b}}, \overline{\mathbf{b}}]$ nonempty floating point intervals such that $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, and $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are non zero and have the same sign, and so are $\underline{\mathbf{a}}$ and $\overline{\mathbf{a}}$. Let X, A and B the sets defined as above, and let x_{min}, x_{max}, a_{min}, a_{max}, b_{min} and b_{max} their respective minimum and maximum. Assume that $x_{min} \leqslant x_{max}$, which means X is non empty, so neither are A and B because of their definitions.*

- *If $x_{min} \oplus a_{min} \geqslant \underline{\mathbf{b}}$ then $b_{min} = x_{min} \oplus a_{min}$.*
- *If $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $x_{min} < 0$ then $b_{min} = a_{min} \oplus \min\{x \in \mathcal{F} \mid x \oplus a_{min} \geqslant \underline{\mathbf{b}}\}$.*
- *If $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $a_{min} < 0$ then $b_{min} = x_{min} \oplus \min\{a \in \mathcal{F} \mid x_{min} \oplus a \geqslant \underline{\mathbf{b}}\}$.*
- *If $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $x_{min} \geqslant 0$ and $a_{min} \geqslant 0$ then $b_{min} = \underline{\mathbf{b}}$.*

Proof Same notations and assumptions. The assumption that $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are non zero and have the same sign means that all the elements of $\mathbf{x}$ (which include all the elements of $[x_{min}, x_{max}]$) are non zero and share the same sign. Similarly, all the elements of $[a_{min}, a_{max}]$ are non zero and share the same sign.

We will use the alternative forms for X, A and B explained above:

$$X = \{x \in [x_{min}, x_{max}] \mid \exists a \in [a_{min}, a_{max}].\ x \oplus a \in \mathbf{b}\}$$
$$A = \{a \in [a_{min}, a_{max}] \mid \exists x \in [x_{min}, x_{max}].\ x \oplus a \in \mathbf{b}\}$$
$$B = \{b \in \mathbf{b} \mid \exists x \in [x_{min}, x_{max}].\ \exists a \in [a_{min}, a_{max}].\ x \oplus a = b\}.$$

From $b_{min} \in B$, let $x' \in [x_{min}, x_{max}]$ and $a' \in [a_{min}, a_{max}]$ such that $x' \oplus a' = b_{min}$.

- Case $x_{min} \oplus a_{min} \geqslant \underline{\mathbf{b}}$. Then, combining this with $x_{min} \oplus a_{min} \leqslant x' \oplus a' = b_{min} \leqslant \overline{\mathbf{b}}$, we get $x_{min} \oplus a_{min} \in \mathbf{b}$ with $x_{min} \in [x_{min}, x_{max}]$ and $a_{min} \in [a_{min}, a_{max}]$. This means that $x_{min} \oplus a_{min} \in B$, but we also have $x_{min} \oplus a_{min} \leqslant b_{min} \stackrel{\text{def}}{=} \min B$, therefore $b_{min} = x_{min} \oplus a_{min}$.
- Case $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $x_{min} < 0$. Then all the elements of $[x_{min}, x_{max}]$ are negative. Moreover, $x' \oplus a' = b_{min} \geqslant \underline{\mathbf{b}} > 0$ so $x' + a' > 0$ with $x' < 0$, therefore $a' > 0$ and all the elements of $[a_{min}, a_{max}]$ are positive.
  Let $x_1 \stackrel{\text{def}}{=} \min\{x \in \mathcal{F} \mid x \oplus a_{min} \geqslant \underline{\mathbf{b}}\}$.
  We want to prove that $x_1 \oplus a_{min} = b_{min}$.
  The definition of x_1 means that $x_1 \oplus a_{min} \geqslant \underline{\mathbf{b}}$ while $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$, so $x_1 > x_{min}$.
  By definition of a_{min}, let $x \in [x_{min}, x_{max}]$ such that $x \oplus a_{min} \in \mathbf{b}$, then $x_1 \leqslant x$ by definition of x_1, so $x_1 \oplus a_{min} \leqslant x \oplus a_{min} \leqslant \overline{\mathbf{b}}$ and also $x_1 \leqslant x \leqslant x_{max}$.
  We have proven $x_{min} < x_1 \leqslant x_{max}$, and $\underline{\mathbf{b}} \leqslant x_1 \oplus a_{min} \leqslant \overline{\mathbf{b}}$ (using the definition of x_1 for the inequality on the left hand side), and trivially $a_{min} \in [a_{min}, a_{max}]$, therefore $x_1 \oplus a_{min} \in B$ and $x_1 \oplus a_{min} \geqslant b_{min}$.
  What we wanted to prove is $x_1 \oplus a_{min} = b_{min}$, so there remains to prove that $x_1 \oplus a_{min} \leqslant b_{min}$. **By contradiction**, assume $x_1 \oplus a_{min} > b_{min}$.
  We already know that $x_1 + a_{min} \geqslant \underline{\mathbf{b}} > 0$ and $x_1 \leqslant x_{max} < 0$, so $|x_1| < a_{min}$.
  From our assumption $x_1 \oplus a_{min} > b_{min} = x' \oplus a'$, and from $a_{min} \leqslant a'$ by definition of a', we obtain $x_1 > x'$.
  We have shown that $x' < x_1 < 0$ and $0 \leqslant |x_1| < a_{min} \leqslant a'$. This means that x', x_1, a_{min} and a' are all 0 modulo $ulp^-(x_1)$ from (0.2). So ${x_{min}}^- + a_{min} = x_{min} - ulp^-(x_{min}) + a_{min} \equiv 0[ulp^-(x_1)]$, and $x' + a' \equiv 0[ulp^-(x_1)]$ which gives $b_{min} = \circ(x' + a') \equiv 0[ulp^-(x_1)]$ using (0.6).
  By definition of x_1, ${x_1}^- \oplus a_{min} < \underline{\mathbf{b}} \leqslant b_{min}$ so ${x_1}^- + a_{min} < b_{min}$ with ${x_1}^- + a_{min} \equiv b_{min}[ulp^-(x_1)]$, so $b_{min} - ({x_1}^- + a_{min}) \geqslant ulp^-(x_1)$ so $x_1 + a_{min} = {x_1}^- + a_{min} + ulp^-(x_1) \leqslant b_{min}$. Then $x_1 \oplus a_{min} \leqslant \circ(b_{min}) = b_{min}$, which contradicts $x_1 \oplus a_{min} > b_{min}$.
  Therefore, $x_1 \oplus a_{min} \leqslant b_{min}$. But we have also proven that $b_{min} \leqslant x_1 \oplus a_{min}$, so $b_{min} = x_1 \oplus a_{min} = a_{min} \oplus \min\{x \in \mathcal{F} \mid x \oplus a_{min} \geqslant \underline{\mathbf{b}}\}$.
- Case $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $a_{min} < 0$. This is the same as the case $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $x_{min} < 0$ by swapping $\mathbf{x}$ and $\mathbf{a}$ since $\oplus$ is commutative.
- Case $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$ and $x_{min} \geqslant 0$ and $a_{min} \geqslant 0$. In particular, all the elements of $[x_{min}, x_{max}]$ and of $[a_{min}, a_{max}]$ are positive.
  Without loss of generality, we can assume that $x_{min} \leqslant a_{min}$ (otherwise we swap everything about $\mathbf{x}$ and $\mathbf{a}$ by commutativity of $\oplus$).
  Let $a_1 \stackrel{\text{def}}{=} \min\{a \in \mathcal{F} \mid x_{min} \oplus a \geqslant \underline{\mathbf{b}}\}$.

To show that $b_{min} = \underline{\mathbf{b}}$, we will show that $\underline{\mathbf{b}} \in B$ by proving that either $\underline{\mathbf{b}} = x_{min} \oplus a_1$ or $\underline{\mathbf{b}} = {x_{min}}^+ \oplus {a_1}^-$ with x_{min}, ${x_{min}}^+ \in [x_{min}, x_{max}]$ and a_1, ${a_1}^- \in [a_{min}, a_{max}]$.
From $a_{min} \in A$ there exists $x \in [x_{min}, x_{max}]$ such that $x \oplus a_{min} \geqslant \underline{\mathbf{b}}$. But $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$, so $x_{min} < x \leqslant x_{max}$, which means ${x_{min}}^+ \leqslant x_{max}$ so ${x_{min}}^+ \in [x_{min}, x_{max}]$.
By definition of a_1 we get $x_{min} \oplus a_1 \geqslant \underline{\mathbf{b}}$, but $x_{min} \oplus a_{min} < \underline{\mathbf{b}}$, so $a_1 > a_{min}$. Moreover, from $x_{min} \in X$ there exists $a \in [a_{min}, a_{max}]$ such that $x_{min} \oplus a \geqslant \underline{\mathbf{b}}$, so $a_1 \leqslant a \leqslant a_{max}$. We obtain $a_1 \in [a_{min}, a_{max}]$ and ${a_1}^- \in [a_{min}, a_{max}]$.
If $\underline{\mathbf{b}} = x_{min} \oplus a_1$, then $\underline{\mathbf{b}} \in B$ so $b_{min} = \underline{\mathbf{b}}$ as we wanted to prove.
We now assume that $\underline{\mathbf{b}} \neq x_{min} \oplus a_1$. As $x_{min} \oplus a_1 \geqslant \underline{\mathbf{b}}$ from the definition of a_1, this means that $x_{min} \oplus a_1 > \underline{\mathbf{b}}$, so $x_{min} + a_1 \geqslant \underline{\mathbf{b}} + \frac{1}{2} ulp^+(\underline{\mathbf{b}})$ using (0.7). Then ${x_{min}}^+ + {a_1}^- > x_{min} + {a_1}^- = x_{min} + a_1 - ulp^+({a_1}^-) \geqslant \underline{\mathbf{b}} + \frac{1}{2} ulp^+(\underline{\mathbf{b}}) - ulp^+({a_1}^-)$. Moreover the definition of a_1 gives $x_{min} \oplus {a_1}^- < \underline{\mathbf{b}}$ so $x_{min} + {a_1}^- < \underline{\mathbf{b}}$ with $x_{min} > 0$ so $0 \leqslant {a_1}^- < \underline{\mathbf{b}}$ so $ulp^+({a_1}^-) \leqslant ulp^-(\underline{\mathbf{b}}) \leqslant ulp^+(\overline{\mathbf{b}})$ using (0.1). We obtain ${x_{min}}^+ + {a_1}^- > \underline{\mathbf{b}} + \frac{1}{2} ulp^+(\underline{\mathbf{b}}) - ulp^+({a_1}^-) \geqslant \underline{\mathbf{b}} - \frac{1}{2} ulp^-(\underline{\mathbf{b}})$, therefore ${x_{min}}^+ + {a_1}^- \geqslant \underline{\mathbf{b}}$ using (0.7).
On the other hand, the definition of a_1 gives $x_{min} \oplus {a_1}^- < \underline{\mathbf{b}}$ so $x_{min} + {a_1}^- \leqslant \underline{\mathbf{b}} - \frac{1}{2} ulp^-(\underline{\mathbf{b}})$ using (0.7). As we have assumed that $x_{min} \leqslant a_{min}$ and shown that $a_{min} < a_1$, we get $0 < x_{min} \leqslant {a_1}^-$. Moreover $x_{min} + {a_1}^- < \underline{\mathbf{b}}$ so $x_{min} < \frac{1}{2} \underline{\mathbf{b}}$ so $ulp^+(x_{min}) \leqslant ulp^-(\frac{1}{2} \underline{\mathbf{b}})$. If $\underline{\mathbf{b}}$ is subnormal, then x_{min} and ${a_1}^-$ are subnormal as well, so all of them are multiples of the smallest ulp $2^{e_{min}-p+1}$, so $x_{min} + {a_1}^- < \underline{\mathbf{b}}$ means $x_{min} + {a_1}^- \leqslant \underline{\mathbf{b}} - 2^{e_{min}-p+1}$ therefore ${x_{min}}^+ + {a_1}^- = x_{min} + 2^{e_{min}-p+1} + {a_1}^- \leqslant \underline{\mathbf{b}}$. If $\underline{\mathbf{b}}$ is not subnormal, then $ulp^+(x_{min}) \leqslant ulp^-(\frac{1}{2} \underline{\mathbf{b}}) = \frac{1}{2} ulp^-(\underline{\mathbf{b}})$, so ${x_{min}}^+ + {a_1}^- = x_{min} + {a_1}^- + ulp^+(x_{min}) \leqslant \underline{\mathbf{b}} - \frac{1}{2} ulp^-(\underline{\mathbf{b}}) + ulp^+(x_{min}) \leqslant \underline{\mathbf{b}}$, therefore ${x_{min}}^+ + {a_1}^- \leqslant \underline{\mathbf{b}}$.
Finally, we have shown that ${x_{min}}^+ + {a_1}^- \geqslant \underline{\mathbf{b}}$ and ${x_{min}}^+ + {a_1}^- \leqslant \underline{\mathbf{b}}$ with ${x_{min}}^+ \in [x_{min}, x_{max}]$ and ${a_1}^- \in [a_{min}, a_{max}]$, so $\underline{\mathbf{b}} \in B$ therefore $b_{min} = \underline{\mathbf{b}}$.
□

The case where $x_{min} \oplus a_{min} \geqslant \underline{\mathbf{b}}$ is obvious. To illustrate that we need as many other cases, consider the following examples in *binary64*.

If $\mathbf{x} = [-2^{100}, -2^{99}]$, $\mathbf{a} = [2^{100}, 2^{101}]$ and $\mathbf{b} = [1, 2^{80}]$, then $b_{min} = 2^{47}$. Moreover $x_{min} = -2^{100}$ as $-2^{100} \oplus (2^{100})^+ = 2^{48} \in \mathbf{b}$, and $a_{min} = 2^{100}$ as $(-2^{100})^+ \oplus 2^{100} = 2^{47} \in \mathbf{b}$. Then we have indeed $a_{min} \oplus \min\{x \in \mathcal{F} \mid x \oplus a_{min} \geqslant \underline{\mathbf{b}}\} = 2^{47} = b_{min}$. It was obvious that b_{min} would not be $x_{min} \oplus a_{min} = 0$ which is smaller than $\underline{\mathbf{b}}$. But b_{min} is not equal to $\underline{\mathbf{b}} = 1$ either, nor $x_{min} \oplus \min\{a \in \mathcal{F} \mid x_{min} \oplus a \geqslant \underline{\mathbf{b}}\} = 2^{48}$.

If $\mathbf{x} = [1, 2^{100}]$, $\mathbf{a} = [1, 2^{100}]$ and $\mathbf{b} = [2^{53} + 2, 2^{100}]$, then $b_{min} = \underline{\mathbf{b}}$, as for example $\underline{\mathbf{b}} = 2^{53} \oplus 2$ with $2^{53} \in \mathbf{x}$ and $2 \in \mathbf{a}$. However, $x_{min} = 1$ and $a_{min} = 1$ as $1 \oplus 2^{55} \in \mathbf{b}$, and $1 \oplus 2^{53} = 2^{53}$ and $1 \oplus (2^{53})^+ = \circ(1 + 2^{53} + 2) = 2^{53} + 4$, so $a_{min} \oplus \min\{x \in \mathcal{F} \mid x \oplus a_{min} \geqslant \underline{\mathbf{b}}\} = x_{min} \oplus \min\{a \in \mathcal{F} \mid x_{min} \oplus a \geqslant \underline{\mathbf{b}}\} = 2^{53} + 4 \neq b_{min}$.

5 Algorithms

Here are algorithms to compute the values defined in §4. Note that the theorems allow several possible algorithms. We are describing one of the possibility, haven

taken mostly efficiency into account. Note also that the predecessor may be computed using the `nextafter` function recommended by the IEEE-754 standard or the algorithm from [14]. Functions return $+\infty$ for the minimum of an empty set, in accordance with our convention.

As explained in §2.4, we assume that $\underline{\mathbf{b}}$ and $\overline{\mathbf{b}}$ are positive and finite, $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are finite, non zero and have the same sign, and $\underline{\mathbf{a}}$ and $\overline{\mathbf{a}}$ are also finite, non zero and have the same sign.

The function XminPT is based on Theorem 2. For $a, b \in \mathcal{F}$ with $b > 0$, it returns the minimum of $\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$.

```
1  function XminPT(a, b)
2  // returns min{x ∈ F | x ⊕ a ⩾ b}, assumes b > 0
3      if b > 2^(e_min+1) ∧ |b ⊖ a| < b ⊘ 2 then
4          x ← (b ⊖ a) ⊖ (ulp⁻(b) ⊘ 2)
5          if x ⊕ a ⩾ b then return x
6          else return x⁺
7          end if
8      else
9          x ← b ⊖ a
10         x1 ← x⁻
11         if x1 ⊕ a ⩾ b then return x1
12         elseif x ⊕ a ⩾ b then return x
13         else return x⁺
14         end if
15     end if
16 end function
```

The function XminITV is based on Theorem 3. For $x_0 \in \mathcal{F}$, $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, it returns the minimum of $\{x \in \mathcal{F} \mid x \geqslant x_0 \wedge (\exists y \in \mathcal{F},\, x \oplus y \in \mathbf{b})\}$.

It uses a function ExpAndWitness which is not detailed in this section. For $\mathbf{b} \in \mathcal{I}(\mathcal{F})$ with $0 < \underline{\mathbf{b}} \leqslant \overline{\mathbf{b}}$, ExpAndWitness$(\underline{\mathbf{b}}, \overline{\mathbf{b}})$ returns the integer g and the floating point number b_g given by *ExpAndWitness*$(\mathbf{b})$ in Definition 1. As explained there, they may be easily computed by dichotomy over the elements in $\mathbf{b}$.

```
1  function XminITV(x0, b_, b¯)
2  // returns min{x ∈ F | x ⩾ x0 ∧ (∃y ∈ F, x ⊕ y ∈ [b_, b¯])}
3  // assumes 0 < b_ ⩽ b¯
4      (g, b_g) ← ExpAndWitness(b_, b¯)
5      u_g ← (2^(g+p) ⊖ 2^g) ⊕ b_g          // 2^(g+p) ⊖ 2^g is (2^(g+p))⁻
6      l_g ← −2^(g+p)
7      if x0 ⩽ l_g then
8          // assert (l_g ⊕ 2^g) ⊕ (b_g ⊖ (l_g ⊕ 2^g)) = b_g
9          return l_g ⊕ 2^g                 // l_g ⊕ 2^g is l_g⁺
10     elseif x0 > u_g then
11         return +∞                        // the set is empty
12     elseif x0 ⊕ XminPT(x0, b_) ⩽ b¯ then
13         return x0
```

```
14      elseif b = b̄ then
15          y1 ← ▽(b − x0)
16          // assert XminPT(y1, b) ⊕ y1 = b
17          return XminPT(y1, b)
18      else
19          // assert x0+ ⊕ XminPT(x0+, b) ⩽ b̄
20          return x0+
21      end if
22  end function
```

The function RefineFirstArgument combines the previous functions as described in Corollary 1, so that it gives the optimal bounds of $X = \{x \in [\underline{\mathbf{x}}, \overline{\mathbf{x}}] \mid \exists a \in [\underline{\mathbf{a}}, \overline{\mathbf{a}}]. \exists b \in [\underline{\mathbf{b}}, \overline{\mathbf{b}}]. x \oplus a = b\}$. It assumes that we also have functions XmaxPT and XmaxITV, which are almost symmetric to XminPT and XminITV but the assumption $0 < \underline{\mathbf{b}}$ is not symmetrized (which prevents us from actually symmetrizing everything to obtain them).

From Corollary 1 we would need to test that $x_{min} \leqslant \overline{\mathbf{x}}$ and $x_{min} \oplus \underline{\mathbf{a}} \leqslant \overline{\mathbf{b}}$ to ensure that X is non empty, in which case x_{min} is really its minimum and x_{max} is really its maximum with no additional test. By construction, $x_{min} \geqslant x_0 \geqslant \underline{\mathbf{x}}$ and $x_0 \geqslant \text{XminPT}(\overline{\mathbf{a}}, \underline{\mathbf{b}})$ so $x_{min} \oplus \overline{\mathbf{a}} \geqslant x_0 \oplus \overline{\mathbf{a}} \geqslant \underline{\mathbf{b}}$, and similarly $x_{max} \leqslant x'_0 \leqslant \overline{\mathbf{x}}$ and $x_{max} \oplus \underline{\mathbf{a}} \leqslant \overline{\mathbf{b}}$. Then $x_{min} \leqslant x_{max}$ implies both $x_{min} \leqslant \overline{\mathbf{x}}$ and $x_{min} \oplus \underline{\mathbf{a}} \leqslant \overline{\mathbf{b}}$ so x_{min} and x_{max} are the extrema of X; whereas $x_{min} > x_{max}$ implies that X is empty (otherwise its extrema would be x_{min} and x_{max}, then $x_{min} > x_{max}$ would be a contradiction). Therefore we only need to test whether $x_{min} \leqslant x_{max}$.

```
1   function RefineFirstArgument(x, x̄, a, ā, b, b̄)
2   // returns the min and max of X = {x ∈ [x, x̄] | ∃a ∈ [a, ā]. ∃b ∈ [b, b̄]. x ⊕ a = b}
3   // returns (+∞, −∞) if X is empty
4   // assumes 0 ⩽ b ⩽ b̄ and no argument is infinite
5       x0 ← max(x, XminPT(ā, b))
6       xmin ← XminITV(x0, b, b̄)
7       x'0 ← min(x̄, XmaxPT(a, b̄))
8       xmax ← XmaxITV(x'0, b, b̄)
9       if xmin ⩽ xmax then
10          return (xmin, xmax)
11      else
12          return (+∞, −∞) // X is empty
13      end if
14  end function
```

The function RefineB is based on Theorem 4. With the usual notations, it assumes that x_{min}, x_{max}, a_{min} and a_{max} are indeed the optimal bounds of X and A, and returns the optimal bounds of B.

Once again, it assumes that we can compute b_{max}, which should be done almost symmetrically to the computation of b_{min}, but the assumption $0 < \underline{\mathbf{b}}$ prevents us from simply taking the exact symmetric of this computation.

```
1   function RefineB(xmin, xmax, amin, amax, b, b̄)
```

```
2  // Returns (b_min, b_max) the bounds of B.
3  // Assumes x_min * x_max > 0, a_min * a_max > 0, b_lo * b_hi > 0, b_lo > 0
4  // and assumes that x_min ≠ +∞
5      if x_min ⊕ a_min ⩾ b_lo || x_min = x_max || a_min = a_max then
6          b_min ← x_min ⊕ a_min
7      elseif x_min < 0 then
8          b_min ← a_min ⊕ XminPT(a_min, b_lo)
9      elseif a_min < 0 then
10         b_min ← x_min ⊕ XminPT(x_min, b_lo)
11     else
12         b_min ← b_lo
13     end if
14     ... // compute b_max
15     return (b_min, b_max)
16 end function
```

Finally, the function RefineAll takes the bounds of the input intervals $\mathbf{x}$, $\mathbf{a}$ and $\mathbf{b}$, and it returns the optimal bounds of $X = \{x \in \mathbf{x} \mid \exists a \in \mathbf{a}.\ \exists b \in \mathbf{b}.\ x \oplus a = b\}$, $A = \{a \in \mathbf{a} \mid \exists x \in \mathbf{x}.\ \exists b \in \mathbf{b}.\ x \oplus a = b\}$, and $B = \{b \in \mathbf{b} \mid \exists x \in \mathbf{x}.\ \exists a \in \mathbf{a}.\ x \oplus a = b\}$. As discussed in §2.4 and §2.1, it is useful to split variable domains into positives values, negative values, zeroes, infinities and NaNs; moreover cases where at least one of the input intervals $\mathbf{x}$, $\mathbf{a}$ and $\mathbf{b}$ is either empty or the singleton corresponding to one of the special values are usually rather easy. Therefore, we assume here that each of these input intervals is nonempty with either only positive or only negative values. However, we do not assume that $\underline{\mathbf{b}} \geqslant 0$ anymore, but we handle this case differently so that this assumption holds for every call to the previous functions.

```
1  function RefineAll(x_lo, x_hi, a_lo, a_hi, b_lo, b_hi)
2  // Returns (x_min, x_max, a_min, a_max, b_min, b_max)
3  // the respective bounds of X, A, B.
4  // Assumes x_lo * x_hi > 0, a_lo * a_hi > 0, b_lo * b_hi > 0
5  // and no argument is infinite
6      if x_lo > x_hi || a_lo > a_hi || b_lo > b_hi then
7          return (+∞, −∞, +∞, −∞, +∞, −∞)
8      elseif b_lo < 0 then
9          (x'_min, x'_max, a'_min, a'_max, b'_min, b'_max) ← RefineAll(−x_hi, −x_lo, −a_hi, −a_lo, −b_hi, −b_lo)
10         return (−x'_max, −x'_min, −a'_max, −a'_min, −b'_max, −b'_min)
11     else
12         (x_min, x_max) ← RefineFirstArgument(x_lo, x_hi, a_lo, a_hi, b_lo, b_hi)
13         if (x_min, x_max) = (+∞, −∞) then
14             return (+∞, −∞, +∞, −∞, +∞, −∞)
15             else // if X ≠ ∅ then also A ≠ ∅ and B ≠ ∅
16                  // we do not need to test this anymore
17             (a_min, a_max) ← RefineFirstArgument(a_lo, a_hi, x_min, x_max, b_lo, b_hi)
18             (b_min, b_max) ← RefineB(x_min, x_max, a_min, a_max, b_lo, b_hi)
19             return (x_min, x_max, a_min, a_max, b_min, b_max)
20         end if
```

```
21      end if
22  end function
```

6 Conclusion and Perspectives

We have shown theorems and proofs about the inverse projection of the FP addition. All this has lead to several actual algorithms shown in Section 5. The existing algorithms of the literature are either not optimal or require an unknown number of iterations (few in practice) to provide the optimal value, while we provide it immediately. Note that our algorithm may also help in bounding this number of iteration.

Furthermore, these algorithms have been implemented in a prototype. This has shown to be quite fast and efficient in practice. It makes sense to be concerned about the speed of the implementation: this article is an exercise in doing proofs at design-time in order to minimize the work done at program-analysis-time. For a more thorough speed evaluation, the following question remains: what is a representative input interval? Indeed, should the working intervals of a realistic static analyzer be expected to small (a few ulps), medium (a 5% difference between bounds), or large (half the FP range)? There are trivial example programs producing all these, and it seems difficult to predict an average width or order of magnitude for realistic programs. The next step is to incorporate this prototype abstract domain in the industrial static analyzer TrustInSoft Analyzer. In addition to statistics about the ranges manipulated during the analysis of real programs, this will doubtless reveal additional directions of exploration.

Now that we have computed the minimum of $\{x \in \mathcal{F} \mid x \oplus a \geqslant b\}$, there remains to compute its maximum. This is not that obvious as we have assumed $b > 0$. Nevertheless, we are convinced the proofs will be easy to generalize.

About the perspectives, we have given the optimal inverse projection of a single FP operation. We have left to study all the other FP operations. Multiplication is expected to be much easier than addition, especially as the results of Section 4.1 also applies to the multiplication. Square root is also expected to be easy as it involves a single argument and is monotone [8], therefore dichotomy would work fine [7]. There remains division, which we expect to be much more challenging. In the case of the equation $b = x \oslash a$ where b is a constant and for some initial intervals for a and x, many values can be shaved from a and x because they rebound[2] from b^- to b^+.

Another perspective concerns the other rounding modes. Given a fixed rounding-mode, we expect to develop rather similar results, probably even simpler. This is an interesting development as it covers both the common directed roundings (towards $\pm\infty$ or towards zero) but also double rounding (with 80-bits extended registers) that can be seen as a directed rounding.

[2] One example is $x \in$[0x1.00068db8bac72p+0,2.0], $a \in$[0x1.000d1b71758e2p+0,2.0] and $b =$0x1.fff2e53a4e1dbp−1. The operation 0x1.00068db8bac72p+0 / 0x1.000d1b71758e2p+0 evaluates to 0x1.fff2e53a4e1dcp−1, whereas 0x1.00068db8bac72p+0 / 0x1.000d1b71758e3p+0 evaluates to 0x1.fff2e53a4e1dap−1, and 0x1.00068db8bac73p+0 / 0x1.000d1b71758e3p+0 evaluates to 0x1.fff2e53a4e1dcp−1 again, and so on.

Another perspective is a clean handling of special values (infinities and NaNs). The given algorithms have been tested intensively with exceptional values, but there is no clear study to ensure the correct behavior in all possible cases.

Acknowledgments

The authors are indebted to Bruno Marre, Tor G. J. Myklebust and Stephen Canon for discussions that lead up to the abstract domain presented in the article and to the question of inverse functions for this abstract domain. Runhang Li implemented the abstract domain; his implementation allowed an experimental approach towards understanding the difficulties.

References

1. Botella, B., Gotlieb, A., Michel, C.: Symbolic execution of floating-point computations. Software Testing, Verification and Reliability **16**(2), 97–121 (2006). DOI 10.1002/stvr.333
2. Chen, L., Miné, A., Cousot, P.: A sound floating-point polyhedra abstract domain. In: G. Ramalingam (ed.) Programming Languages and Systems, pp. 3–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
3. Cousot, P., Cousot, R.: Static determination of dynamic properties of programs. In: Proceedings of the Second International Symposium on Programming, pp. 106–130. Dunod, Paris, France (1976)
4. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: R. Sethi (ed.) Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL), pp. 238–252. ACM, Los Angeles, CA, USA (1977). DOI 10.1145/512950.512973
5. IEEE Computer Society: IEEE standard for interval arithmetic. Tech. Rep. 754-2008 (2008). DOI 10.1109/IEEESTD.2008.4610935
6. ISO: International standard ISO/IEC 9899:2011, Programming languages – C (2011)
7. Marre, B., Michel, C.: Improving the floating point addition and subtraction constraints. In: D. Cohen (ed.) Principles and Practice of Constraint Programming – CP 2010, pp. 360–367. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
8. Michel, C.: Exact projection functions for floating point number constraints. In: International Symposium on Artificial Intelligence and Mathematics. Fort Lauderdale, Florida, USA (2002). URL `http://rutcor.rutgers.edu/~amai/aimath02/PAPERS/21.ps`. (AI&M 2002)
9. Michel, C., Rueher, M., Lebbah, Y.: Solving constraints over floating-point numbers. In: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, CP '01, pp. 524–538. Springer-Verlag, London, UK (2001). URL `http://dl.acm.org/citation.cfm?id=647488.726803`
10. Miné, A.: Relational abstract domains for the detection of floating-point run-time errors. In: D. Schmidt (ed.) Programming Languages and Systems, pp. 3–17. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
11. Monniaux, D.: The pitfalls of verifying floating-point computations. ACM Transactions on Programming Languages and Systems **30**(3), 1–41 (2008). DOI 10.1145/1353445.1353446
12. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs, NJ (1963)
13. Muller, J.M.: On the definition of ulp(x). Tech. Rep. RR-5504, INRIA (2005). URL `https://hal.inria.fr/inria-00070503`
14. Rump, S.M., Zimmermann, P., Boldo, S., Melquiond, G.: Computing predecessor and successor in rounding to nearest. BIT **49**(2), 419–431 (2009). DOI 10.1007/s10543-009-0218-z. URL `http://hal.inria.fr/inria-00337537/fr/`
15. Sterbenz, P.H.: Floating Point Computation. Prentice Hall (1974)