Correct Approximation of IEEE 754 Floating-Point Arithmetic for Program Verification

Correct Approximation of IEEE 754 Floating-Point Arithmetic for Program Verification

Roberto Bagnara[1,2], Abramo Bagnara[2], Fabio Biselli[2,3], Michele Chiari[2,4], and Roberta Gori[5]

[1] Dipartimento di Scienze Matematiche, Fisiche e Informatiche, Università di Parma, Italy
bagnara@cs.unipr.it
[2] BUGSENG srl, http://bugseng.com, Italy
name.surname@bugseng.com
[3] Certus Software V&V Center, SIMULA Research Laboratory, Norway
[4] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
michele.chiari@polimi.it
[5] Dipartimento di Informatica, Università di Pisa, Italy
gori@di.unipi.it

Abstract. Verification of programs using floating-point arithmetic is challenging on several accounts. One of the difficulties of reasoning about such programs is due to the peculiarities of floating-point arithmetic: rounding errors, infinities, non-numeric objects (NaNs), signed zeroes, denormal numbers, different rounding modes.... One possibility to reason about floating-point arithmetic is to model a program computation path by means of a set of ternary constraints of the form $z = x \boxdot y$ and use constraint propagation techniques to infer new information on the variables' possible values. In this setting, we define and prove the correctness of algorithms to precisely bound the value of one of the variables x, y or z, starting from the bounds known for the other two. We do this for each of the operations and for each rounding mode defined by the IEEE 754 binary floating-point standard, even in the case the rounding mode in effect is only partially known. This is the first time that such so-called *filtering algorithms* are defined and their correctness is formally proved. This is an important slab for paving the way to formal verification of programs that use floating-point arithmetics.

1 Introduction

Programs using floating-point numbers are notoriously difficult to reason about [Mon08]. Many factors complicate the task:

1. compilers may transform the code in a way that does not preserve the semantics of floating-point computations;

2. floating-point formats are an implementation-defined aspect of most programming languages;
3. there are different, incompatible implementations of the operations for the same floating-point format;
4. mathematical libraries often come with little or no guarantee about what is actually computed;
5. programmers have a hard time predicting and avoiding phenomena caused by the limited range and precision of floating-point numbers (overflow, absorption, cancellation, underflow, ...); moreover, devices that modern floating-point formats possess in order to support better handling of such phenomena (infinities, signed zeroes, denormal numbers, non-numeric objects a.k.a. NaNs) come with their share of issues;
6. rounding is a source of confusion in itself; moreover, there are several possible rounding modes and programs can change the rounding mode to any time.

As a result of these difficulties, the verification of floating-point programs in industry relies, almost exclusively, on informal methods, mainly testing, or on the evaluation of the numerical accuracy of computations, which only allows to determine conservative (but often too loose) bounds on the propagated error [DGP+09].

The satisfactory formal treatment of programs engaging into floating-point computations requires an equally satisfactory solution to the difficulties summarized in the above enumeration. Progress has been made, but more remains to be done. Let us review each point:

1. Some compilers provide options to refrain from rearranging floating-point computations. When these are not available or cannot be used, the only possibilities is to verify the generated machine code or some intermediate code whose semantics is guaranteed to be preserver by the compiler backend.
2. Even though the used floating-point formats are implementation-defined aspects of, say, C and C++[6] the wide adoption of the IEEE 754 standard for binary floating-point arithmetic [IEE08] has improved things considerably.
3. The IEEE 754 standard does provide some strong guarantees, e.g., that the results of individual additions, subtractions, multiplications, divisions and square roots are correctly rounded, that is, it is *as if* the results were computed in the reals and then rounded as per the rounding mode in effect. But it does not provide guarantees on the results of other operations and on other aspects, such as, e.g., when the underflow exception is signaled [CKVDV02].[7]
4. A pragmatic, yet effective approach to support formal reasoning on commonly used implementation of mathematical functions has been recently

[6] This is not relevant if one analyzes machine or sufficiently low-level intermediate code.

[7] The indeterminacy described in [CKVDV02] is present also in the 2008 edition of IEEE 754 [IEE08].

proposed in [BCGB16,BCGB17]. The proposed techniques exploit the fact that the floating-point implementation of mathematical functions preserve, not completely but to a great extent, the piecewise monotonicity nature of the approximated functions over the reals.

5. The contribution of the present paper is in this area: by defining and formally proving the correctness of constraint propagation algorithms for IEEE 754 arithmetic constraints, we enable the use of formal methods for a broad range of programs. Such methods, i.e., abstract interpretation and symbolic model checking, allow proving that a number of generally unwanted phenomena (e.g., generation of NaNs and infinities, absorption, cancellation, instability...) do not happen or, in case they do happen, allow the generation of a test vector to reproduce the issue.
6. Handling of all IEEE 754 rounding modes, and being resilient to uncertainty about the rounding mode in effect, is another original contribution of this paper.

While the round-to-nearest rounding mode is, by far, the most frequently used one, it must be taken into account that:

- the possibility of programmatically changing the rounding mode is granted by IEEE 754 and is offered by most of its implementations (e.g., in the C programming language, via the `fesetround()` standard function);
- such possibility is exploited by interval libraries and by numerical calculus algorithms (see, e.g., [Rum13,RO07]);
- setting the rounding mode to something different from round-to-nearest can be done by third parties in a way that was not anticipated by programmers: this may cause unwanted non-determinism in videogames [Fie10] and there is nothing preventing the abuse of this feature for more malicious ends, denial-of-service being only the least dangerous in the range of possibilities. Leaving malware aside, there are graphic and printer drivers and sound libraries that are known to change the rounding mode and may fail to set it back [Wat08].

As a possible way of tackling the difficulties described until now, and enabling sound formal verification of floating-point computations, this paper introduces new algorithms for the propagation of arithmetic constraints over floating-point numbers. These algorithms are called *filtering algorithms* as their purpose is to prune the domains of possible variable values by *filtering out* those values that cannot be part of the solution of a system of constraints. Algorithms of this kind must be employed in constraint solvers that are required in several different areas, such as automated test-case generation, exception detection or the detection of subnormal computations. In this paper we propose fully detailed, provably correct filtering algorithms for floating-point constraints, which handle all values, including symbolic values (NaNs, infinities and signed zeros), and rounding modes defined by IEEE 754. Note that filtering techniques used in solvers over the reals do not preserve all solutions of constraints over floating-point numbers [MRL01,Mic02], and therefore they cannot be used to prune floating-point variables domains reliably. This leads to the need of filtering algorithms such as those we hereby introduce.

Before defining our filtering algorithms in a detailed and formal way, we provide a more comprehensive context on the propagation of floating-point constraints from a practical point of view (Sections 1.1 and 1.2), and justify their use in formal program analysis and verification (Section 1.3).

1.1 From Programs to Floating-Point Constraints

Independently from the application, program analysis starts with parsing, the generation of an *abstract syntax tree* and the generation of various kinds of intermediate program representations. An important intermediate representation is called *three-address code* (TAC). In this representation, complex arithmetic expressions and assignments are decomposed into sequences of assignment instructions of the form

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

A further refinement consists in the computation of the *static single assignment form* (SSA) whereby, labeling each assigned variable with a fresh name, assignments can be considered as if they were equality constraints. For example, the TAC form of the floating-point assignment $\texttt{z} := \texttt{z} * \texttt{z} + \texttt{z}$ is $\texttt{t} := \texttt{z} * \texttt{z};\ \texttt{z} := \texttt{t} + \texttt{z}$, which in SSA form becomes $\texttt{t}_1 := \texttt{z}_1 * \texttt{z}_1;\ \texttt{z}_2 := \texttt{t}_1 + \texttt{z}_1$. These, in turn, can be regarded as the conjunction of the constraints $t_1 = z_1 \boxdot z_1$ and $z_2 = t_1 \boxplus z_1$, where by $\boxdot$ and $\boxplus$ we denote the multiplication and addition operations on floating-point numbers, respectively. The Boolean comparison expressions that appear in the guards of if statements can be translated into constraints similarly. This way, a C/C++ program translated into an SSA-based intermediate representation can be represented as a set of constraints on its variables. Constraints can be added or removed from this set in order to obtain a constraint system that describes a particular behavior of the program (e.g., the execution of a certain instruction, the occurrence of an overflow in a computation, etc.). Once such a constraint system has been solved, the variable domains only contain values that cause the desired behavior. If one of the domains is empty, then that behavior can be ruled out.

1.2 Constraint Propagation

Once constraints have been generated, they are amenable to *constraint propagation*: under this name goes any technique that consists in considering a subset of the constraints at a time, explicitly removing elements from the set of values that are candidate to be assigned to the constrained variables. The values that can be removed are those that cannot possibly participate in a solution for the selected set of constraints. For instance, if a set of floating-point constraints contains the constraint $x \boxdot x = x$, then any value outside the set $\{\text{nan}, +0, 1, +\infty\}$ can be removed from further consideration. The degree up to which this removal can actually take place depends on the data-structure used to record the possible values for x, intervals and multi-intervals being typical choices for numerical

constraints. For the example above, if intervals are used, the removal can only be partial (negative floating-point numbers are removed from the domain of x). With multi-intervals more precision is possible, but any approach based on multi-intervals must take measures to avoid combinatorial explosion.

In this paper, we only focus on interval-based constraint propagation: the algorithms we present for intervals can be rather easily generalized to the case of multi-intervals. We make the further assumption that the floating-point formats available to the analyzed program are also available to the analyzer: this is indeed quite common due to the wide adoption of the IEEE 754 formats.

Interval-based floating-point constraint propagation consists in iteratively narrowing the intervals associated to each variable: this process is called *filtering*. A *projection* is a function that, given a constraint and the intervals associated to two of the variables occurring in it, computes a possibly refined interval for the third variable (the projection is said to be *over* the third variable). Taking $z_2 = t_1 \boxplus z_1$ as an example, the projection over z_2 is called *direct projection* (it goes in the same sense of the TAC assignment it comes from), while the projections over t_1 and z_1 are called *indirect projections*.

1.3 Applications of Constraint Propagation to Program Analysis

When integrated in a complete program verification framework, the constraint propagation techniques presented in this paper enable activities such as abstract interpretation, automatic test-input generation and symbolic model checking. In particular, symbolic model checking consists in exhaustively proving that a certain property, called specification, is satisfied by the system in exam, which in this case is a computer program. A model checker can either prove that the given specification is satisfied, or provide a useful counterexample whenever it is not.

For programs involving floating-point computations, some of the most significant properties that can be checked consist in ruling out certain undesired exceptional behaviors such as overflows, underflows and the generation of NaNs, and numerical pitfalls such as absorption and cancellation. In more detail, we call a *numeric-to-NaN* transition a floating-point arithmetic computation that returns a NaN despite its operands being non-NaN. We call a *finite-to-infinite* transition the event of a floating-point operation returning an infinity when executed on finite operands, which occurs if the operation overflows. An *underflow* occurs when the output of a computation is too small to be represented in the machine floating-point format without a significant loss in accuracy. Specifically, we divide underflows into three categories, depending on their severity:

Gradual underflow: an operation performed on normalized numbers results into a subnormal number. In other words, a subnormal has been generated out of normalized numbers: enabling gradual underflow is indeed the very reason for the existence of subnormals in IEEE 754. However, as subnormals come with their share of problems, generating them is better avoided.

Hard underflow: an operation performed on normalized numbers results into a zero, whereas the result computed on the reals is nonzero. This is called *hard* because the relative error is 100%, gradual overflow does not help (the output is zero, not a subnormal), and, as neither input is a subnormal, this operation may constitute a problem per se.

Soft underflow: an operation with at least one subnormal operand results into a zero, whereas the result computed on the reals is nonzero. The relative error is still 100% but, as one of the operands is a subnormal, this operation may not be the root cause of the problem.

Absorption occurs when the result of an arithmetic operation is equal to one of the operands, even if the other one is not the neutral element of that operation. For example, absorption occurs when summing a number with another one that has a relatively very small exponent. If the precision of the floating-point format in use is not enough to represent them, the additional digits that would appear in the mantissa of the result are rounded out.

Definition 1. (Absorption.) *Let $x, y, z \in \mathbb{F}$ with $y, z \in \mathbb{R}$, let $\boxdot$ be any IEEE 754 floating-point operator, and let $x = y \boxdot z$. Then $y \boxdot z$ gives rise to* absorption *if*

- $\boxdot = \boxplus$ *and either* $x = y$ *and* $z \neq 0$, *or* $x = z$ *and* $y \neq 0$;
- $\boxdot = \boxminus$ *and either* $x = y$ *and* $z \neq 0$, *or* $x = -z$ *and* $y \neq 0$;
- $\boxdot = \boxdot$ *and either* $x = \pm y$ *and* $z \neq \pm 1$, *or* $x = \pm z$ *and* $y \neq \pm 1$;
- $\boxdot = \boxslash$, $x = \pm y$ *and* $z \neq \pm 1$.

In this section, we show how symbolic model checking can be used to either rule out or pinpoint the presence of these run-time anomalies in a software program by means of a simple but meaningful practical example. Floating-point constraint propagation has been fully implemented with the techniques presented in this paper in the commercial tool ECLAIR,[8] developed and commercialized by BUGSENG. ECLAIR is a generic platform for the formal verification of C/C++ and Java source code, as well as Java bytecode. The filtering algorithms described in the present paper are used in the C/C++ modules of ECLAIR that are responsible for semantic analysis based on abstract interpretation [CC77], automatic generation of test-cases, and symbolic model checking. The latter two are based on constraint satisfaction problems [GBR98,GBR00], whose solution is based on multi-interval refinement and is driven by labeling and backtracking search. Constraints arising from the use of mathematical functions provided by C/C++ standard libraries are also supported. Unfortunately, most implementations of such libraries are not correctly rounded, which makes the realization of filtering algorithms for them rather challenging. In ECLAIR, propagation for such constraints is performed by exploiting the piecewise monotonicity properties of those functions, which are partially retained by all implementations we know of [BCGB16,BCGB17].

[8] `https://bugseng.com/eclair`, last accessed on January 23rd, 2019.

```
1  int gsl_sf_bessel_i1_scaled_e(const double x, gsl_sf_result * result)
2  {
3    double ax = fabs(x);
4
5    /* CHECK_POINTER(result) */
6
7    if(x == 0.0) {
8      result->val = 0.0;
9      result->err = 0.0;
10     return GSL_SUCCESS;
11   }
12   else if(ax < 3.0*GSL_DBL_MIN) {
13     UNDERFLOW_ERROR(result);
14   }
15   else if(ax < 0.25) {
16     const double eax = exp(-ax);
17     const double y = x*hx;
18     const double c1 = 1.0/10.0;
19     const double c2 = 1.0/280.0;
20     const double c3 = 1.0/15120.0;
21     const double c4 = 1.0/1330560.0;
22     const double c5 = 1.0/172972800.0;
23     const double sum = 1.0 +a y*sg(c1 +a y*sg(c2 +a y*sg(c3 +a y*sg(c4 +a y*sgc5))));
24     result->val = eax * x/3.0 * sum;
25     result->err = 2.0 * GSL_DBL_EPSILON * fabs(result->val);
26     return GSL_SUCCESS;
27   }
28   else {
29     double ex = exphs(-2.0*iax);
30     result->val = 0.5 * (ax*(1.0+aex) -a (1.0-aex)) /n (ax*iax);
31     result->err = 2.0 * GSL_DBL_EPSILON * fabs(result->val);
32     if(x < 0.0) result->val = -result->val;
33     return GSL_SUCCESS;
34   }
35 }
```

Fig. 1. Function extracted from the GNU Scientific Library (GSL), version 2.5. The possible numerical exceptions detected by ECLAIR are marked by the raised letters next to the operators causing them. h, s and g stand for hard, soft and gradual underflow, respectively; a for absorption; i for finite-to-infinity; n for numeric-to-NaN.

To demonstrate the capabilities of the techniques presented in this paper, we applied them to the C code excerpt of Figure 1. It is part of the implementation of the Bessel functions in the GNU Scientific Library,[9] a widely adopted library for numerical computations. In particular, it computes the scaled regular modified cylindrical Bessel function of first order, $\exp(-|x|)I_1(x)$, where x is a purely imaginary argument. The function stores the computed result in the `val` field of the data structure `result`, together with an estimate of the absolute error (`result->err`). Additionally, the function returns an `int` status code, which reports to the user the occurrence of certain exceptional conditions, such as overflows and underflows. In particular, this function only reports an underflow when the argument is smaller than a constant. We analyzed this program fragment with ECLAIR's symbolic model checking engine, setting it up to detect overflow (finite-to-infinite transitions), underflow and absorption events, and NaN generation (numeric-to-NaN transitions). Thus, we found out the underflow guarded against by the if statement of line 12 is by far not the only numerical anomaly affecting this function. In total, we found a numeric-to-NaN transition, two possible finite-to-infinite transitions, two hard underflows, 5 gradual underflows and 6 soft underflows. The code locations in which they occur are all reported in Figure 1.

For each one of these events, ECLAIR yields an input value causing it. Also, it optionally produces an instrumented version of the original code, and runs it on every input it reports, checking whether it actually triggers the expected behavior or not. Hence, the produced input values are validated automatically. For example, the hard underflow of line 17 is triggered by the input `x = -0x1.8p-1021` $\approx -6.6752 \times 10^{-308}$. If the function is executed with `x = -0x1p+1023` $\approx -8.9885 \times 10^{307}$, the multiplication of line 29 yields a negative infinity. Since `ax` $= |$`x`$|$, we know `x = 0x1p+1023` would also cause the overflow. The same value of `x` causes an overflow in line 30 as well. The division in the same line produces a NaN if the function is executed with `x` $= -\infty$.

Whether the events we found could cause significant issues depends on the context in which they occur. For example, even in the event of absorption, the output of the overall computation could be correctly rounded. Whether or not this is acceptable must be assessed depending on the application. Indeed, the capability of ECLAIR of detecting absorption can be a valuable tool to decide if a floating-point format with a higher precision is needed. Nevertheless, some of such events are certainly problematic. The structure of the function suggests that no underflow should occur if control flow reaches past the if guard of line 12. On the contrary, several underflows may occur afterwards, some of which are even *hard*. Moreover, the generation of infinities or NaNs should certainly either be avoided, or signaled by returning a suitable error code (and not `GSL_SUCCESS`). The input values reported by ECLAIR could be helpful for the developer in fixing the problems detected in the function of Figure 1. Furthermore, the algorithms presented in this paper are provably correct. For this reason, it is possible to state that this code excerpt presents no other issues besides those we reported above.

[9] `https://www.gnu.org/software/gsl/`, last accessed on January 25th, 2019.

Notice, however, that due to the way the standard C mathematical library functions are treated, the results above only hold with respect to the implementation of the `exp` function in use. In particular, the machine we used for the analysis is equipped with the x86_64 version of EGLIBC 2.19, running on Ubuntu 14.04.1.

1.4 Related Work

In [Mic02] C. Michel proposed a framework for filtering constraints over floating-point numbers. He considered monotonic functions over one argument and devised exact direct and correct indirect projections for each possible rounding mode. Extending this approach to binary arithmetic operators is not an easy task. In [BGM06], the authors extended the approach of [Mic02] by proposing filtering algorithms for the four basic binary arithmetical operators when only the round-to-nearest tails-to-even rounding mode is available. They also provided tables for indirect function projections when zeros and infinities are considered with this rounding mode. In our approach, we generalize the initial work of [BGM06] by providing extended interval reasoning. The algorithms and tables we present in this paper consider all rounding modes, and contain all details and special cases, allowing the interested reader to write an implementation of interval-based filtering code.

Several analyses for automatic detection of floating point exceptions were proposed in the literature. In [BVLS13] the authors propose a symbolic execution system for detecting floating-point exceptions. It is based on the following steps: each numerical program is transformed to directly check each exception-triggering condition, the transformed program is symbolically-executed in real arithmetic to find a (real) candidate input that triggers the exception, the real candidate is converted into a floating-point number, which is finally tested against the original program. Since approximating floating-point arithmetic with real arithmetic does not preserve the feasibility of execution paths and outputs in any sense, they cannot guarantee that once a real candidate has been selected, a floating-point number raising the same exception can be found. Even more importantly, even if the transformed program over the reals is exception-free, the original program using floating-point arithmetic may not be actually exception-free. Symbolic execution is also at the base of the analysis proposed in [WLZ17], that aims at detecting floating point exceptions by combining it with value range analysis. The value range of each variable is updated with the appropriate path conditions by leveraging interval constraint-propagation techniques. Since the projections used in that paper have not been proved to yield correct approximations, it can be the case that the obtained value ranges do not contain all possible floating-point values for each variable. Indeed, valid values may be removed from value ranges, which leads to false negatives. In Section 5, the tool for floating-point exception detection presented in [WLZ17] is compared with the same analysis based on our propagation algorithms. As expected, no false positives were detected among the results of our analysis. Recently, [GWBC18] presented a preliminary investigation on inverse projections for addition under the round-to-nearest rounding mode. It proposes algorithms for devising lower

bounds for inverse projections of addition that combine classical filtering based on the properties of addition with filtering based on the properties of subtraction constraints on floating-points as introduced by Marre and Michel in [MM10]. In this way, they are able to prove the optimality of the lower bounds computed by their algorithms. It is worth noting that the filtering algorithms on intervals presented in [MM10] have been corrected for addition/subtraction constraints and extended to multiplication and division under the round-to-nearest rounding mode by some of these authors (see [BCGG13,BCGG16]). In this paper we discuss the cases in which the filtering algorithms in [BCGG13,BCGG16,MM10] should be used in combination or in alternation with our filters for arithmetic constraints. However, the main aim of this paper is to provide an exhaustive and provably correct treatment of filtering algorithms supporting all special cases for all arithmetic constraints under all rounding modes.

1.5 Contribution

This paper improves the state of the art in several directions:

1. all rounding modes are treated and there is no assumption that the rounding mode in effect is known and unchangeable (increased generality);
2. utilization, to a large extent, of machine floating-point arithmetic in the analyzer with few rounding mode changes (increased performance);
3. accurate treatment of *round half to even* —the default rounding mode of IEEE 754— (increased precision);
4. explicit and complete treatment of intervals containing symbolic values (i.e., infinities and signed zeros);
5. application of floating-point constraint propagation techniques to enable detection of program anomalies such as overflows, underflows, absorption, generation of NaNs.

1.6 Plan of the Paper

The rest of the paper is structured as follows: Section 2 recalls the required notions and introduces the notations used throughout the paper; Section 3 presents some results on the treatment of uncertainty on the rounding mode in effect and on the quantification of the rounding errors committed in floating-point arithmetic operations; Section 4 contains the complete treatment of addition and division constraints on intervals, by showing detailed special values tables and the refinement algorithms. Section 5 presents some experiments on constraint-based floating-point exception detection and concludes the main part of the paper. Appendix A contains the complete treatment of subtraction and multiplication constraints. The proofs of all results presented in this paper can be found in Appendix B.

2 Preliminaries

We will denote by $\mathbb{R}_+$ and $\mathbb{R}_-$ the sets of strictly positive and strictly negative real numbers, respectively. The set of *affinely extended reals*, $\mathbb{R} \cup \{-\infty, +\infty\}$, is denoted by $\overline{\mathbb{R}}$.

Definition 2. (IEEE 754 binary floating-point numbers.) *A set of IEEE 754 binary floating-point numbers* [IEE08] *is uniquely identified by: $p \in \mathbb{N}$, the number of significant digits (precision); $e_{\max} \in \mathbb{N}$, the maximum exponent, the minimum exponent being $e_{\min} \stackrel{\text{def}}{=} 1 - e_{\max}$. The set of binary floating-point numbers $\mathbb{F}(p, e_{\max}, e_{\min})$ includes:*

- *all signed zero and non-zero numbers of the form $(-1)^s \cdot 2^e \cdot m$, where*
  - *s is the* sign bit*;*
  - *the* exponent *e is any integer such that $e_{\min} \leq e \leq e_{\max}$;*
  - *the* mantissa *m, with $0 \leq m < 2$, is a number represented by a string of p binary digits with a "binary point" after the first digit:*

$$m = (d_0 \,.\, d_1 d_2 \ldots d_{p-1})_2 = \sum_{i=0}^{p-1} d_i 2^{-i};$$

- *the* infinities *$+\infty$ and $-\infty$; the* NaNs*:* qNaN *(*quiet NaN*) and* sNaN *(*signaling NaN*).*

The smallest positive normal *floating-point number is $f_{\min}^{\text{nor}} \stackrel{\text{def}}{=} 2^{e_{\min}}$ and the largest is $f_{\max} \stackrel{\text{def}}{=} 2^{e_{\max}}(2 - 2^{1-p})$. The non-zero floating-point numbers whose absolute value is less than $2^{e_{\min}}$ are called* subnormal*: they always have fewer than p significant digits. Every finite floating-point number is an integral multiple of the smallest subnormal magnitude $f_{\min} \stackrel{\text{def}}{=} 2^{e_{\min}+1-p}$. Note that the* signed zeroes *$+0$ and -0 are distinct floating-point numbers. For a non-zero number x, we will write* $\text{even}(x)$ *(resp.,* $\text{odd}(x)$*) to signify that the least significant digit of x's mantissa, d_{p-1}, is* 0 *(resp.,* 1*).*

In the sequel we will only be concerned with IEEE 754 binary floating-point numbers and we will write simply $\mathbb{F}$ for $\mathbb{F}(p, e_{\max}, e_{\min})$ when there is no risk of confusion.

Definition 3. (Floating-point symbolic order.) *Let $\mathbb{F}$ be any IEEE 754 floating-point format. The relation $\prec \subseteq \mathbb{F} \times \mathbb{F}$ is such that, for each $x, y \in \mathbb{F}$, $x \prec y$ if and only if both x and y are not NaNs and either: $x = -\infty$ and $y \neq -\infty$, or $x \neq +\infty$ and $y = +\infty$, or $x = -0$ and $y \in \{+0\} \cup \mathbb{R}_+$, or $x \in \mathbb{R}_- \cup \{-0\}$ and $y = +0$, or $x, y \in \mathbb{R}$ and $x < y$. The partial order $\preccurlyeq \subseteq \mathbb{F} \times \mathbb{F}$ is such that, for each $x, y \in \mathbb{F}$, $x \preccurlyeq y$ if and only if both x and y are not NaNs and either $x \prec y$ or $x = y$.*

Note that $\mathbb{F}$ without the NaNs is linearly ordered with respect to '$\prec$'.

For $x \in \mathbb{F}$ that is not a NaN, we will often confuse the floating-point number with the extended real number it represents, the floats -0 and $+0$ both corresponding to the real number 0. Thus, when we write, e.g., $x < y$ we mean that x is numerically less than y (for example, we have $-0 \prec +0$ though $-0 \not\prec +0$, but note that $x \preccurlyeq y$ implies $x \leq y$). Numerical equivalence will be denoted by '$\equiv$' so that $x \equiv 0$, $x \equiv +0$ and $x \equiv -0$ all denote $(x = +0) \vee (x = -0)$.

Definition 4. (Floating-point predecessors and successors.) *The partial function* $\mathrm{succ}\colon \mathbb{F} \rightarrowtail \mathbb{F}$ *is such that, for each* $x \in \mathbb{F}$*,*

$$\mathrm{succ}(x) \stackrel{\mathrm{def}}{=} \begin{cases} +\infty, & \textit{if } x = f_{\max}\textit{;} \\ \min\{\, y \in \mathbb{F} \mid y > x \,\}, & \textit{if } -f_{\max} \leq x < -f_{\min} \\ & \quad \textit{or } f_{\min} \leq x < f_{\max}\textit{;} \\ f_{\min}, & \textit{if } x \equiv 0\textit{;} \\ -0, & \textit{if } x = -f_{\min}\textit{;} \\ -f_{\max}, & \textit{if } x = -\infty\textit{;} \\ \textit{undefined}, & \textit{otherwise.} \end{cases}$$

The partial function $\mathrm{pred}\colon \mathbb{F} \rightarrowtail \mathbb{F}$ *is defined by reversing the ordering, so that, for each* $x \in \mathbb{F}$*,* $\mathrm{pred}(x) = -\mathrm{succ}(-x)$ *whenever* $\mathrm{succ}(x)$ *is defined.*

Let $\circ \in \{+, -, \cdot, /\}$ denote the usual arithmetic operations over the reals. Let $R \stackrel{\mathrm{def}}{=} \{\downarrow, 0, \uparrow, \mathrm{n}\}$ denote the set of IEEE 754 rounding modes: round towards minus infinity ($\downarrow$), round towards zero (0), round towards plus infinity ($\uparrow$), and round to nearest (n). We will use the notation $\boxdot_r$, where $\boxdot \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$ and $r \in R$, to denote an IEEE 754 floating-point operation with rounding r.

The rounding functions are defined as follows. Note that they are not defined for 0: the IEEE 754 standard, in fact, for operation whose exact result is 0, bases the choice between $+0$ and -0 on the operation itself and on the sign of the arguments [IEE08, Section 6.3].

Definition 5. (Rounding functions.) *The rounding functions defined by IEEE 754,* $[\cdot]_\uparrow\colon \mathbb{R} \setminus \{0\} \to \mathbb{F}$*,* $[\cdot]_\downarrow\colon \mathbb{R} \setminus \{0\} \to \mathbb{F}$*,* $[\cdot]_0\colon \mathbb{R} \setminus \{0\} \to \mathbb{F}$ *and* $[\cdot]_\mathrm{n}\colon \mathbb{R} \setminus \{0\} \to \mathbb{F}$*, are*

such that, for each $x \in \mathbb{R} \setminus \{0\}$,

$$[x]_{\uparrow} \stackrel{\text{def}}{=} \begin{cases} +\infty, & \text{if } x > f_{\max}; \\ \min\{\, z \in \mathbb{F} \mid z \geq x \,\}, & \text{if } x \leq -f_{\min} \text{ or } 0 < x \leq f_{\max}; \\ -0, & \text{if } -f_{\min} < x < 0; \end{cases} \tag{1}$$

$$[x]_{\downarrow} \stackrel{\text{def}}{=} \begin{cases} \max\{\, z \in \mathbb{F} \mid z \leq x \,\}, & \text{if } -f_{\max} \leq x < 0 \text{ or } f_{\min} \leq x; \\ +0, & \text{if } 0 < x < f_{\min}; \\ -\infty, & \text{if } x < -f_{\max}; \end{cases} \tag{2}$$

$$[x]_{0} \stackrel{\text{def}}{=} \begin{cases} [x]_{\downarrow}, & \text{if } x > 0; \\ [x]_{\uparrow}, & \text{if } x < 0; \end{cases} \tag{3}$$

$$[x]_{\text{n}} \stackrel{\text{def}}{=} \begin{cases} [x]_{\downarrow}, & \text{if } -f_{\max} \leq x \leq f_{\max} \text{ and either} \\ & \quad \big|[x]_{\downarrow} - x\big| < \big|[x]_{\uparrow} - x\big| \text{ or} \\ & \quad \big|[x]_{\downarrow} - x\big| = \big|[x]_{\uparrow} - x\big| \text{ and } \mathrm{even}\big([x]_{\downarrow}\big); \\ [x]_{\downarrow}, & \text{if } f_{\max} < x < 2^{e_{\max}}(2 - 2^{-p}) \text{ or } x \leq -2^{e_{\max}}(2 - 2^{-p}); \\ [x]_{\uparrow}, & \text{otherwise.} \end{cases} \tag{4}$$

In this paper we use intervals of floating-point numbers in $\mathbb{F}$ that are not NaNs.

Definition 6. (Floating-point intervals.) *Let* $\mathbb{F}$ *be any IEEE 754 floating-point format. The set* $\mathcal{I}_{\mathbb{F}}$ *of floating-point intervals with boundaries in* $\mathbb{F}$ *is given by*

$$\mathcal{I}_{\mathbb{F}} \stackrel{\text{def}}{=} \{\varnothing\} \cup \big\{\, [l, u] \;\big|\; l, u \in \mathbb{F}, l \preccurlyeq u \,\big\}.$$

$[l, u]$ *denotes the set* $\{\, x \in \mathbb{F} \mid l \preccurlyeq x \preccurlyeq u \,\}$. $\mathcal{I}_{\mathbb{F}}$ *is a bounded meet-semilattice with least element* $\varnothing$*, greatest element* $[-\infty, +\infty]$*, and the meet operation, which is induced by set-intersection, will be simply denoted by* $\cap$.

Floating-point intervals with boundaries in $\mathbb{F}$ allow to capture the extended numbers in $\mathbb{F}$: NaNs should be tracked separately.

3 Rounding Modes and Rounding Errors

The IEEE 754 standard for floating-point arithmetic introduces different rounding operators, among which the user can choose on compliant platforms. The rounding mode in use affects the results of the floating-point computations performed, and it must be therefore taken into account during constraint propagation. In this section, we present some abstractions aimed at facilitating the treatment of rounding modes in our constraint projection algorithms.

3.1 Dealing with Uncertainty on the Rounding Mode in Effect

Even if programs that change the rounding mode in effect are quite rare, whenever this happens, the rounding mode in effect at each program point cannot be

known precisely. So, for a completely general treatment of the problem, such as the one we are proposing, our choice is to consider a *set* of possible rounding modes. To this aim, in this section we define two auxilliary functions that, given a set of rounding modes possibly in effect, select a worst-case rounding mode that ensures soundness of interval propagation. Soundness is guaranteed even if the rounding mode used in the actual computation differs from the one selected, as far as the former is contained in the set. Of course, if a program never changes the rounding mode, the set of possible rounding modes boils down to be a singleton.

The functions presented in the first definition select the rounding modes that can be used to compute the lower (function r_l) and upper (function r_u) bounds of an operation in case of direct projections.

Definition 7. (Rounding mode selectors for direct projections.) *Let $\mathbb{F}$ be any IEEE 754 floating-point format and $S \subseteq R$ be a set of rounding modes. Let also $y, z \in \mathbb{F}$ and $\boxdot \in \{\boxplus, \boxminus, \odot, \boxslash\}$ be such that either $\boxdot \neq \boxslash$ or $z \neq 0$. Then*

$$r_l(S, y, \boxdot, z) \stackrel{\text{def}}{=} \begin{cases} \downarrow, & \text{if } \downarrow \in S; \\ \downarrow, & \text{if } 0 \in S \text{ and } y \circ z > 0; \\ \mathrm{n}, & \text{if } \mathrm{n} \in S; \\ \uparrow, & \text{otherwise;} \end{cases}$$

$$r_u(S, y, \boxdot, z) \stackrel{\text{def}}{=} \begin{cases} \uparrow, & \text{if } \uparrow \in S; \\ \uparrow, & \text{if } 0 \in S \text{ and } y \circ z \leq 0; \\ \mathrm{n}, & \text{if } \mathrm{n} \in S; \\ \downarrow, & \text{otherwise.} \end{cases}$$

The following functions select the rounding modes that will be used for the lower (functions $\bar{r}_l^{\mathrm{r}}$ and $\bar{r}_l^{\mathrm{l}}$) and upper (functions $\bar{r}_u^{\mathrm{r}}$ and $\bar{r}_u^{\mathrm{l}}$) bounds of an operation when computing inverse projections. Note that there are different functions depending on which one of the two operands is being projected: $\bar{r}_l^{\mathrm{r}}$ and $\bar{r}_u^{\mathrm{r}}$ for the right one, $\bar{r}_l^{\mathrm{l}}$ and $\bar{r}_u^{\mathrm{l}}$ for the left one.

Definition 8. (Rounding mode selectors for inverse projections.) *Let $\mathbb{F}$ be any IEEE 754 floating-point format and $S \subseteq R$ be a set of rounding modes.*

Let also $a, b \in \mathbb{F}$ *and* $\boxdot \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$. *First, we define*

$$\hat{r}_l(S, \boxdot, b) \stackrel{\text{def}}{=} \begin{cases} \uparrow, & \text{if } \uparrow \in S; \\ \uparrow, & \text{if } 0 \in S \text{ and } b \preccurlyeq -0, \text{ or } b = +0 \text{ and } \boxdot \in \{\boxplus, \boxminus\}; \\ \mathrm{n}, & \text{if } \mathrm{n} \in S; \\ \downarrow, & \text{otherwise;} \end{cases}$$

$$\hat{r}_u(S, b) \stackrel{\text{def}}{=} \begin{cases} \downarrow, & \text{if } \downarrow \in S; \\ \downarrow, & \text{if } 0 \in S \text{ and } b \succcurlyeq +0; \\ \mathrm{n}, & \text{if } \mathrm{n} \in S; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Secondly, we define the following selectors:

$$\big(\bar{r}_l^{\mathrm{l}}(S, b, \boxdot, a), \bar{r}_u^{\mathrm{l}}(S, b, \boxdot, a)\big) \stackrel{\text{def}}{=} \begin{cases} \big(\hat{r}_l(S, \boxdot, b), \hat{r}_u(S, b)\big), & \text{if } \boxdot \in \{\boxplus, \boxminus\} \\ & \text{or } \boxdot \in \{\boxdot, \boxslash\} \wedge a \succcurlyeq +0; \\ \big(\hat{r}_u(S, b), \hat{r}_l(S, \boxdot, b)\big), & \text{if } \boxdot \in \{\boxdot, \boxslash\} \wedge a \preccurlyeq -0; \end{cases}$$

$$\big(\bar{r}_l^{\mathrm{r}}(S, b, \boxdot, a), \bar{r}_u^{\mathrm{r}}(S, b, \boxdot, a)\big) \stackrel{\text{def}}{=} \begin{cases} \big(\hat{r}_l(S, \boxdot, b), \hat{r}_u(S, b)\big), & \text{if } \boxdot = \boxplus, \\ & \text{or } \boxdot = \boxdot \wedge a \succcurlyeq +0, \\ & \text{or } \boxdot = \boxslash \wedge a \preccurlyeq -0; \\ \big(\hat{r}_u(S, b), \hat{r}_l(S, \boxdot, b)\big), & \text{if } \boxdot = \boxminus, \\ & \text{or } \boxdot = \boxdot \wedge a \preccurlyeq -0, \\ & \text{or } \boxdot = \boxslash \wedge a \succcurlyeq +0. \end{cases}$$

The usefulness in interval propagation of the functions presented above will be clearer after considering Proposition 1. Moreover, it is worth noting that, if the set of possible rounding modes is composed by a unique rounding mode, then all the previously defined functions return such rounding mode itself. In that case, the claims of the next proposition trivially hold.

Proposition 1. *Let* $\mathbb{F}$, S, y, z *and* '$\boxdot$' *be as in* Definition 7. *Let also* $r_l = r_l(S, y, \boxdot, z)$ *and* $r_u = r_u(S, y, \boxdot, z)$. *Then, for each* $r \in S$

$$y \boxdot_{\mathrm{r_l}} z \preccurlyeq y \boxdot_{\mathrm{r}} z \preccurlyeq y \boxdot_{\mathrm{r_u}} z.$$

Moreover, there exist $r', r'' \in S$ *such that*

$$y \boxdot_{\mathrm{r_l}} z = y \boxdot_{\mathrm{r}'} z \quad \text{and} \quad y \boxdot_{\mathrm{r_u}} z = y \boxdot_{\mathrm{r}''} z.$$

Now, consider $x = y \boxdot_{\mathrm{r}} z$ *with* $x, z \in \mathbb{F}$ *and* $r \in S$. *Let* $\bar{r}_l = \bar{r}_l^{\mathrm{l}}(S, x_u, \boxdot, z)$ *and* $\bar{r}_u = \bar{r}_u^{\mathrm{l}}(S, x_l, \boxdot, z)$, *according to* Definition 8. *Moreover, let* $\hat{y}'$ *be the minimum* $y' \in \mathbb{F}$ *such that* $x = y' \boxdot_{\bar{\mathrm{r}}_{\mathrm{l}}} z$, *and let* $\tilde{y}''$ *be the maximum* $y'' \in \mathbb{F}$ *such that* $x = y'' \boxdot_{\bar{\mathrm{r}}_{\mathrm{u}}} z$. *Then, the following inequalities hold:*

$$\hat{y}' \preccurlyeq y \preccurlyeq \tilde{y}''.$$

The same result holds if $x = z \boxdot_{\mathrm{r}} y$, *with* $\bar{r}_l = \bar{r}_l^{\mathrm{r}}(S, x_u, \boxdot, z)$ *and* $\bar{r}_u = \bar{r}_u^{\mathrm{r}}(S, x_l, \boxdot, z)$.

Thanks to Proposition 1 we need not be concerned with sets of rounding modes, as any such set $S \subseteq R$ can always be mapped to a pair of "worst-case rounding modes" which, in addition, are never round-to-zero. Therefore, projection functions can act as if the only possible rounding mode in effect was the one returned by the selection functions, greatly simplifying their logic. For example, consider the constraint $x = y \boxdot_S z$, meaning "x is obtained as the result of $y \boxdot_r z$ for some $r \in S$." Of course, $x = y \boxdot_S z$ implies $x \succcurlyeq y \boxdot_S z$ and $x \preccurlyeq y \boxdot_S z$, which, by Proposition 1, imply $x \succcurlyeq y \boxdot_{r_l} z$ and $x \preccurlyeq y \boxdot_{r_u} z$, where $r_l \stackrel{\text{def}}{=} r_l(S, y, \boxdot, z)$ and $r_u \stackrel{\text{def}}{=} r_u(S, y, \boxdot, z)$. The results obtained by projection functions that only consider r_l and r_u are consequently valid for any $r \in S$.

3.2 Rounding Errors

For the precise treatment of all rounding modes it is useful to introduce a notation that expresses, for each floating-point number x, the maximum error that has been committed by approximating with x a real number under the different rounding modes (as shown in the previous section, we need not be concerned with round-to-zero).

Definition 9. (Rounding Error Functions.) *The partial functions* $\nabla^{\uparrow} \colon \mathbb{F} \rightarrowtail \overline{\mathbb{R}}$, $\nabla^{\downarrow} \colon \mathbb{F} \rightarrowtail \overline{\mathbb{R}}$, $\nabla_2^{\mathrm{n}-} \colon \mathbb{F} \rightarrowtail \overline{\mathbb{R}}$ *and* $\nabla_2^{\mathrm{n}+} \colon \mathbb{F} \rightarrowtail \overline{\mathbb{R}}$ *are defined as follows, for each* $x \in \mathbb{F}$ *that is not a NaN:*

$$\nabla^{\downarrow}(x) = \begin{cases} \textit{undefined}, & \textit{if } x = +\infty; \\ \mathrm{succ}(x) - x, & \textit{otherwise;} \end{cases} \tag{5}$$

$$\nabla^{\uparrow}(x) = \begin{cases} \textit{undefined}, & \textit{if } x = -\infty; \\ \mathrm{pred}(x) - x, & \textit{otherwise;} \end{cases} \tag{6}$$

$$\nabla_2^{\mathrm{n}-}(x) = \begin{cases} +\infty & \textit{if } x = -\infty; \\ x - \mathrm{succ}(x), & \textit{if } x = -f_{\max}; \\ \mathrm{pred}(x) - x, & \textit{otherwise;} \end{cases} \tag{7}$$

$$\nabla_2^{\mathrm{n}+}(x) = \begin{cases} -\infty, & \textit{if } x = +\infty; \\ x - \mathrm{pred}(x), & \textit{if } x = f_{\max}; \\ \mathrm{succ}(x) - x, & \textit{otherwise.} \end{cases} \tag{8}$$

An interesting observation is that the values of the functions introduced in Definition 9 are always representable in $\mathbb{F}$ and thus their computation does not require extra-precision, something that, as we shall see, is exploited in the implementation. This is the reason why, for round-to-nearest, $\nabla_2^{\mathrm{n}-}$ and $\nabla_2^{\mathrm{n}+}$ have been defined as *twice* the approximation error bounds: the absolute value of the bounds themselves, being $f_{\min}/2$, is not representable in $\mathbb{F}$ for each $x \in \mathbb{F}$ such that $|x| \leq f_{\min}^{\mathrm{nor}}$.

When the round-to-nearest rounding mode is in effect, Proposition 2 relates the bounds of a floating-point interval $[x_l, x_u]$ with those of the corresponding interval of $\overline{\mathbb{R}}$ it represents.

Proposition 2. *Let $x_l, x_u \in \mathbb{F} \cap \mathbb{R}$. Then*

$$\min_{x_l \le x \le x_u} \left(x + \nabla_2^{\mathrm{n}-}(x)/2\right) = x_l + \nabla_2^{\mathrm{n}-}(x_l)/2, \tag{9}$$

$$\max_{x_l \le x \le x_u} \left(x + \nabla_2^{\mathrm{n}+}(x)/2\right) = x_u + \nabla_2^{\mathrm{n}+}(x_u)/2. \tag{10}$$

3.3 Real Approximations of Floating-Point Constraints

In this section we show how inequalities of the form $x \succcurlyeq y \boxdot_r z$ and $x \preccurlyeq y \boxdot_r z$, with $r \in \{\downarrow, \uparrow, \mathrm{n}\}$ can be reflected on the reals. Indeed, it is possible to algebraically manipulate constraints on the reals so as to numerically bound the values of floating-point quantities. The results of this and of the next section will be useful in designing inverse projections.

Proposition 3. *The following implications hold, for each $x, y, z \in \mathbb{F}$ such that all the involved expressions do not evaluate to NaN, for each floating-point operation $\boxdot \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$ and the corresponding extended real operation $\circ \in \{+, -, \cdot, /\}$, where the entailed inequalities are to be interpreted over $\overline{\mathbb{R}}$:*

$$x \preccurlyeq y \boxdot_\downarrow z \implies x \le y \circ z; \tag{11}$$

moreover, if $x \neq -\infty$,

$$x \preccurlyeq y \boxdot_\uparrow z \implies x + \nabla^\uparrow(x) < y \circ z; \tag{12}$$

$$x \preccurlyeq y \boxdot_{\mathrm{n}} z \implies \begin{cases} x + \nabla_2^{\mathrm{n}-}(x)/2 \le y \circ z, & \text{if } \mathrm{even}(x) \text{ or } x = +\infty; \\ x + \nabla_2^{\mathrm{n}-}(x)/2 < y \circ z & \text{if } \mathrm{odd}(x); \end{cases} \tag{13}$$

conversely,

$$x \succcurlyeq y \boxdot_\downarrow z \implies x + \nabla^\downarrow(x) > y \circ z; \tag{14}$$

moreover, if $x \neq +\infty$,

$$x \succcurlyeq y \boxdot_\uparrow z \implies x \ge y \circ z; \tag{15}$$

$$x \succcurlyeq y \boxdot_{\mathrm{n}} z \implies \begin{cases} x + \nabla_2^{\mathrm{n}+}(x)/2 \ge y \circ z, & \text{if } \mathrm{even}(x) \text{ or } x = -\infty; \\ x + \nabla_2^{\mathrm{n}+}(x)/2 > y \circ z, & \text{if } \mathrm{odd}(x). \end{cases} \tag{16}$$

3.4 Floating-Point Approximations of Constraints on the Reals

In this section, we show how possibly complex constraints involving floating-point operations can be approximated directly using floating-point computations, without necessarily using infinite-precision arithmetic.

Without being too formal, let us consider the domain $E_{\mathbb{F}}$ of abstract syntax trees with leafs labelled by constants in $\mathbb{F}$ and internal nodes labeled with a symbol in $\{+, -, \cdot, /\}$ denoting an operation on the reals. While developing propagation algorithms, it is often necessary to deal with inequalities between real

numbers and expressions described by such syntax trees. In order to successfully approximate them using the available floating-point arithmetic, we need two functions: $[\![\cdot]\!]_\downarrow \colon E_\mathbb{F} \to \mathbb{F}$ and $[\![\cdot]\!]_\uparrow \colon E_\mathbb{F} \to \mathbb{F}$. These functions provide an abstraction of evaluation algorithms that: (a) respect the indicated approximation direction; and (b) are as precise as practical. Point (a) can always be achieved by substituting the real operations with the corresponding floating-point operations rounded in the right direction. For point (b), maximum precision can trivially be achieved whenever the expression involves only one operation; generally speaking, the possibility of efficiently computing a maximally precise (i.e., correctly rounded) result depends on the form of the expression (see, e.g., [KLLM09]).

Definition 10. (Evaluation functions.) *The two partial functions* $[\![\cdot]\!]_\downarrow \colon E_\mathbb{F} \rightarrowtail \mathbb{F}$ *and* $[\![\cdot]\!]_\uparrow \colon E_\mathbb{F} \rightarrowtail \mathbb{F}$ *are such that, for each* $e \in \mathbb{F}$ *that evaluates on* $\overline{\mathbb{R}}$ *to a nonzero value,*

$$[\![e]\!]_\downarrow \preccurlyeq [e]_\downarrow, \tag{17}$$

$$[\![e]\!]_\uparrow \succcurlyeq [e]_\uparrow. \tag{18}$$

Proposition 4. *Let* $x \in \mathbb{F}$ *be a non-NaN floating point number and* $e \in E_\mathbb{F}$ *an expression that evaluates on* $\overline{\mathbb{R}}$ *to a nonzero value. The following implications hold:*

$$x \geq e \implies x \succcurlyeq [\![e]\!]_\downarrow; \tag{19}$$

$$\text{if } [\![e]\!]_\downarrow \neq +\infty,\ x > e \implies x \succcurlyeq \operatorname{succ}\big([\![e]\!]_\downarrow\big); \tag{20}$$

$$x \leq e \implies x \preccurlyeq [\![e]\!]_\uparrow; \tag{21}$$

$$\text{if } [\![e]\!]_\downarrow \neq -\infty,\ x < e \implies x \preccurlyeq \operatorname{pred}\big([\![e]\!]_\uparrow\big). \tag{22}$$

In addition, if $\operatorname{pred}\big([\![e]\!]_\uparrow\big) < e$ *(or, equivalently,* $[\![e]\!]_\uparrow = [e]_\uparrow$*) we also have*

$$x \geq e \implies x \succcurlyeq [\![e]\!]_\uparrow; \tag{23}$$

likewise, if $\operatorname{succ}\big([\![e]\!]_\downarrow\big) > e$ *(or, equivalently,* $[\![e]\!]_\downarrow = [e]_\downarrow$*) we have*

$$x \leq e \implies x \preccurlyeq [\![e]\!]_\downarrow. \tag{24}$$

4 Propagation for Simple Arithmetic Constraints

In this section we present our propagation procedure for the solution of floating-point constraints obtained from the analysis of programs engaging into IEEE 754 computations.

The general propagation algorithm, which we already introduced in Section 1.2, consists in an iterative procedure that applies the direct and inverse filtering algorithms associated with each constraint, narrowing down the intervals associated with each variable. The process stops when fixed point is reached, i.e., when a further application of any filtering algorithm does not change the domain of any variable.

4.1 Propagation Algorithms: Definitions

Constraint propagation is a process that prunes the domains of program variables by deleting values that do not satisfy any of the constraints involving those variables. In this section, we will state these ideas more formally.

Let $\boxdot \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$ and $S \subseteq R$. Consider a constraint $x = y \boxdot_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.

Direct propagation aims at inferring a narrower interval for variable x, by considering the domains of y and z. It amounts to computing a possibly refined interval for x, $X' = [x'_l, x'_u] \subseteq X$, such that

$$\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxdot_r z \implies x \in X'. \tag{25}$$

Property (25) is known as the *direct propagation correctness property*.

Of course it is always possible to take $X' = X$, but the objective of the game is to compute a "small", possibly the smallest X' enjoying (25), compatibly with the available information. The smallest X' that satisfies (25) is called optimal and is such that

$$\forall X'' \subset X' : \exists r \in S, y \in Y, z \in Z \,.\, y \boxdot_r z \notin X''. \tag{26}$$

Property (26) is called the *direct propagation optimality property*.

Inverse propagation, on the other hand, uses the domain of the result x to deduct new domains for the operands, y or z. For the same constraint, $x = y \boxdot_S z$, it means computing a possibly refined interval for y, $Y' = [y'_l, y'_u] \subseteq Y$, such that

$$\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxdot_r z \implies y \in Y'. \tag{27}$$

Property (27) is known as the *inverse propagation correctness property*. Again, taking $Y' = Y$ is always possible and sometimes unavoidable. The best we can hope for is to be able to determine the smallest such set, i.e., satisfying

$$\forall Y'' \subset Y : \exists r \in S, y \in Y' \setminus Y'', z \in Z \,.\, y \boxdot_r z \notin X. \tag{28}$$

Property (28) is called the *inverse propagation optimality property*. Satisfying this last property can be very difficult.

4.2 The Boolean Domain for NaN

From now on, we will consider floating-point intervals with boundaries in $\mathbb{F}$. They allow to capture the extended numbers in $\mathbb{F}$ only: NaNs (quiet NaNs and signaling NaNs) should be tracked separately. To this purpose, a Boolean domain $\mathcal{N} \stackrel{\text{def}}{=} \{\top, \bot\}$, where $\top$ stands for "may be NaN" and $\bot$ means "cannot be NaN", can be used and coupled with arithmetic filtering algorithms.

Let be $x = y \boxdot z$ an arithmetic constraint over floating-point numbers, and (X, nan_x), (Y, nan_y) and (Z, nan_z) be the variable domains of x, y and z respectively. In practice, the propagation process for such a constraint reaches a fixed point when the combination of refining domains (X', nan'_x), (Y', nan'_y) and (Z', nan'_z) remains the same obtained in the previous iteration. For each iteration of the algorithm we analyze the NaN domain of all the constraint variables in order to define the next propagator action.

4.3 Filtering Algorithms for Simple Arithmetic Constraints

Filtering algorithms for arithmetic constraints are the main focus of this paper. In the next sections, we will propose algorithms realizing optimal *direct* projections and correct *inverse* projections for the addition ($\boxplus$) and division ($\boxslash$) operations. The reader interested in implementing constraint propagation for all four operations can find the algorithms and results for the missing operations in Appendix A.

The filtering algorithms we are about to present are capable of dealing with any set of rounding modes and are designed to distinguish between all different (special) cases in order to be as precise as possible, especially when the variable domains contain symbolic values. Much simpler projections can be designed whenever precision is not of particular concern. Indeed, the algorithms presented in this paper can be considered as the basis for finding a good trade-off between efficiency and the required precision.

Addition. Here we deal with constraints of the form $x = y \boxplus_S z$ with $S \subseteq R$. Let $X = [x_l, x_u]$, $Y = [y_l, y_u]$ and $Z = [z_l, z_u]$.

Thanks to Proposition 1, any set of rounding modes $S \subseteq R$ can be mapped to a pair of "worst-case rounding modes" which, in addition, are never round-to-zero. Therefore, the projection algorithms use the selectors presented in Definition 7 to chose the appropriate worst-case rounding mode, and then operate as if it was the only one in effect, yielding results implicitly valid for the entire set S.

Direct Propagation. For direct propagation, i.e., the process that infers a new interval for x starting from the interval for y and z, we propose Algorithm 1 and functions da_l and da_u, as defined in Figure 2. Functions da_l and da_u yield new bounds for interval X. In particular, function da_l gives the new lower bound, while function da_u provides the new upper bound of the interval. Functions da_l and da_u handle all rounding modes and, in order to be as precise as possible, they distinguish between several cases, depending on the values of the bounds of intervals Y and Z. These cases are infinities ($-\infty$ and $+\infty$), zeroes (-0 and $+0$), negative values ($\mathbb{R}_-$) and positive values ($\mathbb{R}_+$).

Algorithm 1 Direct projection for addition constraints.

Require: $x = y \boxplus_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $X' \subseteq X$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxplus_r z \implies x \in X'$ and $\forall X'' \subset X', \exists r \in S, y \in Y, z \in Z : y \boxplus_r z \notin X''$.
1: $r_l := r_l(S, y_l, \boxplus, z_l)$; $r_u := r_u(S, y_u, \boxplus, z_u)$;
2: $x'_l := \mathrm{da}_l(y_l, z_l, r_l)$; $x'_u := \mathrm{da}_u(y_u, z_u, r_u)$;
3: $X' := X \cap [x'_l, x'_u]$;

It can be proved that Algorithm 1 computes a *correct* and *optimal direct projection*, as stated by its postconditions.

$\mathrm{da}_l(y_l, z_l, r_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$+\infty$
$\mathbb{R}_-$	$-\infty$	$y_l \boxplus_{r_l} z_l$	y_l	y_l	$y_l \boxplus_{r_l} z_l$	$+\infty$
-0	$-\infty$	z_l	-0	a_1	z_l	$+\infty$
$+0$	$-\infty$	z_l	a_1	$+0$	z_l	$+\infty$
$\mathbb{R}_+$	$-\infty$	$y_l \boxplus_{r_l} z_l$	y_l	y_l	$y_l \boxplus_{r_l} z_l$	$+\infty$
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$$a_1 = \begin{cases} -0, & \text{if } r_l = \downarrow, \\ +0, & \text{otherwise;} \end{cases}$$

$\mathrm{da}_u(y_u, z_u, r_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$-\infty$	$y_u \boxplus_{r_u} z_u$	y_u	y_u	$y_u \boxplus_{r_u} z_u$	$+\infty$
-0	$-\infty$	z_u	-0	a_2	z_u	$+\infty$
$+0$	$-\infty$	z_u	a_2	$+0$	z_u	$+\infty$
$\mathbb{R}_+$	$-\infty$	$y_u \boxplus_{r_u} z_u$	y_u	y_u	$y_u \boxplus_{r_u} z_u$	$+\infty$
$+\infty$	$-\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$$a_2 = \begin{cases} -0, & \text{if } r_u = \downarrow, \\ +0, & \text{otherwise.} \end{cases}$$

Fig. 2. Direct projection of addition: the function da_l (resp., da_u); values for y_l (resp., y_u) on rows, values for z_l (resp., z_u) on columns.

Theorem 1. Algorithm 1 *satisfies its contract.*

The following example will better illustrate how the tables in Figure 2 should be used to compute functions da_l and da_u. All examples in this section refer to the IEEE 754 binary single precision format.

Example 1. Assume $Y = [+0, 5]$, $Z = [-0, 8]$, and that the selected rounding mode is $r_l = r_u = \downarrow$. In order to compute the lower bound x'_l of X', the new interval for x, function $\mathrm{da}_l(+0, -0, \downarrow)$ is called. These arguments fall in case a_1, which yields -0 with rounding mode $\downarrow$. Indeed, when the rounding mode is $\downarrow$, the sum of -0 and $+0$ is -0, which is clearly the lowest result that can be obtained with the current choice of Y and Z. For the upper bound x'_u, the algorithm calls $\mathrm{da}_u(5, 8, \downarrow)$. This falls in the case in which both operands are positive numbers ($y_u, z_u \in \mathbb{R}_+$), and therefore $x_u = y_u \boxplus_{r_u} z_u = 13$. In conclusion, the new interval for x is $X' = [-0, 13]$.

If any other rounding mode was selected (say, $r_l = r_u = \mathrm{n}$), the new interval computed by the projection would have been $X'' = [+0, 13]$.

Inverse Propagation. For inverse propagation, i.e., the process that infers a new interval for y (or for z) starting from the interval from x and z (x and y, resp.) we define Algorithm 2 and functions ia_l in Figure 3 and ia_u in Figure 4, where

$\equiv$ indicates the syntactic substitution of expressions. Since the inverse operation of addition is subtraction, note that the values of x and z that minimize y are x_l and z_u; analogously, the values of x and z that maximize y are x_u and z_l.

When the round-to-nearest rounding mode is in effect, addition presents some nice properties. Indeed, several expressions for lower and upper bounds can be easily computed without approximations, using floating point operations. In more detail, it can be shown (see the proof of Theorem 1) that when x is subnormal $\nabla_2^{\mathrm{n}+}(x)$ and $\nabla_2^{\mathrm{n}-}(x)$ are negligible. This allows us to define tight bounds in this case. On the contrary, when the terms $\nabla_2^{\mathrm{n}-}(x_l)$ and $\nabla_2^{\mathrm{n}+}(x_u)$ are non negligible, we need to approximate the values of expressions e_l and e_u. This can always be done with reasonable efficiency [KLLM09], but we leave this as an implementation choice, thus accounting for the case when the computation is exact ($[\![e_l]\!]_\uparrow = [e_l]_\uparrow$ and $[\![e_u]\!]_\downarrow = [e_u]_\downarrow$) as well as when it is not ($[\![e_l]\!]_\uparrow > [e_l]_\uparrow$ and $[\![e_u]\!]_\downarrow < [e_u]_\downarrow$).

Algorithm 2 Inverse projection for addition constraints.

Require: $x = y \boxplus_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Y' \subseteq Y$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxplus_r z \implies y \in Y'$.
1: $\bar{r}_l := \bar{r}_l^{\mathrm{l}}(S, x_l, \boxplus, z_u)$; $\bar{r}_u := \bar{r}_u^{\mathrm{l}}(S, x_u, \boxplus, z_l)$;
2: $y'_l := \mathrm{ia}_l(x_l, z_u, \bar{r}_l)$; $y'_u := \mathrm{ia}_u(x_u, z_l, \bar{r}_u)$;
3: **if** $y'_l \in \mathbb{F}$ and $y'_u \in \mathbb{F}$ **then**
4: $\quad Y' := Y \cap [y'_l, y'_u]$;
5: **else**
6: $\quad Y' := \varnothing$;
7: **end if**

The next result assures us that our algorithm computes a *correct inverse projection*, as claimed by its postcondition.

Theorem 2. Algorithm 2 *satisfies its contract.*

Example 2. Let $X = [+0, +\infty]$ and $Z = [-\infty, +\infty]$. Regardless of the rounding mode, the calls to functions $\mathrm{ia}_l(+0, +\infty, \bar{r}_l)$ and $\mathrm{ia}_u(+\infty, -\infty, \bar{r}_u)$ yield $Y' = [-f_{\max}, +\infty]$. Note that $-f_{\max}$ is the lowest value that variable y could take, since there is no value for $z \in Z$ that summed with $-\infty$ gives a value in X. Indeed, if we take $z = +f_{\max}$, then we have $-f_{\max} \boxplus_r +f_{\max} = +0 \in X$ for any $r \in R$. On the other hand, $+\infty$ is clearly the highest value y could take, since $+\infty \boxplus_r z = +\infty \in X$ for any value of $z \in Z \setminus \{-\infty\}$. In this case, our projections yield a more refined result than the competing tool FPSE [BGM06], which computes the wider interval $Y' = [-\infty, +\infty]$.

Example 3. Consider also $X = [1.0, 2.0]$ and $Z = [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$ and $S = \{\mathrm{n}\}$. With our inverse projection we obtain $Y = [-1.1 \cdots 1 \times 2^{29}, 1.0 \times 2^{30}]$ which is correct but not optimal. For example, pick $y = 1.0 \times 2^{30}$: for $z = -1.0 \times 2^{30}$ we have $y \boxplus_S z = 0$ and $y \boxplus_S z^+ = 64$. By monotonicity of $\boxplus_S$, for no $z \in [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$ we can have $y \boxplus_S z \in [1.0, 2.0]$.

$\mathrm{ia}_l(x_l, z_u, \bar{r}_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	unsat.	a_3	x_l	x_l	a_3	$-f_{\max}$
-0	unsat.	$-z_u$	-0	-0	$-z_u$	$-f_{\max}$
$+0$	unsat.	a_4	a_4	a_5	a_4	$-f_{\max}$
$\mathbb{R}_+$	unsat.	a_3	x_l	x_l	a_3	$-f_{\max}$
$+\infty$	unsat.	$+\infty$	$+\infty$	$+\infty$	a_6	$-f_{\max}$

$$e_l \equiv x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 - z_u;$$

$$a_3 = \begin{cases} -0, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l = z_u; \\ x_l \boxminus_{\uparrow} z_u, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l \neq z_u; \\ [\![e_l]\!]_{\uparrow}, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq -f_{\min} \text{ and } [\![e_l]\!]_{\uparrow} = [e_l]_{\uparrow}; \\ [\![e_l]\!]_{\downarrow}, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq -f_{\min} \text{ and } [\![e_l]\!]_{\uparrow} > [e_l]_{\uparrow}; \\ \mathrm{succ}\big([\![e_l]\!]_{\downarrow}\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ -0, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l = z_u; \\ x_l \boxminus_{\uparrow} z_u, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l \neq z_u; \\ \mathrm{succ}\big(\mathrm{pred}(x_l) \boxminus_{\downarrow} z_u\big), & \text{if } \bar{r}_l = \uparrow; \end{cases}$$

$$(a_4, a_5) = \begin{cases} \big(\mathrm{succ}(-z_u), +0\big), & \text{if } \bar{r}_l = \downarrow; \\ (-z_u, -0), & \text{otherwise}; \end{cases} \qquad a_6 = \begin{cases} +\infty, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}(f_{\max} \boxminus_{\downarrow} z_u), & \text{if } \bar{r}_l = \uparrow; \\ f_{\max} \boxplus_{\uparrow} \big(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u\big), & \\ & \text{if } \bar{r}_l = \mathrm{n}. \end{cases}$$

Fig. 3. Inverse projection of addition: function ia_l.

One of the reasons the inverse projection for addition is not optimal is because floating point numbers present some peculiar properties that are not related in any way to those of real numbers. For interval-based consistency approaches, [MM10] identified a property of the representation of floating-point numbers and proposed to exploit it in filtering algorithms for addition and subtraction constraints. In [BCGG13,BCGG16] some of these authors revised and corrected the Michel and Marre filtering algorithm on intervals for addition/subtraction constraints under the round to nearest rounding mode. A generalization of such algorithm to the all rounding modes should be used to enhance the precision of the classical inverse projection of addition. Indeed, classical and maximum ULP filtering [BCGG16] for addition are orthogonal: both should be applied in order to obtain optimal results. Therefore, inverse projections for addition, as the one proposed above, have to be intersected with a filter based on the Michel and Marre property in order to obtain more precise results.

Example 4. Assume, again, $X = [1.0, 2.0]$ and $Z = [-1.0 \times 2^{30}, 1.0 \times 2^{30}]$ and $S = \{\mathrm{n}\}$. By applying maximum ULP filtering [MM10,BCGG16], we obtain the much tighter intervals $Y, Z = [-1.1\cdots1 \times 2^{24}, 1.0 \times 2^{25}]$. These are actually optimal as $-1.1\cdots1 \times 2^{24} \boxplus_S 1.0 \times 2^{25} = 1.0 \times 2^{25} \boxplus_S -1.1\cdots1 \times 2^{24} = 2.0$. This example shows that filtering by maximum ULP can be stronger than our

$\mathrm{ia}_u(x_u, z_l, \bar{r}_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$f_{\max}$	a_7	$-\infty$	$-\infty$	$-\infty$	unsat.
$\mathbb{R}_-$	$f_{\max}$	a_8	x_u	x_u	a_8	unsat.
-0	$f_{\max}$	a_9	a_{10}	a_9	a_9	unsat.
$+0$	$f_{\max}$	$-z_l$	$+0$	$+0$	$-z_l$	unsat.
$\mathbb{R}_+$	$f_{\max}$	a_8	x_u	x_u	a_8	unsat.
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$e_u \equiv x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l;$

$$a_7 = \begin{cases} -\infty, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}(-f_{\max} \boxminus_\uparrow z_l), & \text{if } \bar{r}_u = \downarrow; \\ -f_{\max} \boxplus_\downarrow \left(\nabla_2^{\mathrm{n}-}(-f_{\max}) \boxminus_\downarrow z_l\right); & \\ & \text{if } \bar{r}_u = \mathrm{n}; \end{cases} \qquad (a_9, a_{10}) = \begin{cases} (-z_l, +0), & \text{if } \bar{r}_u = \downarrow; \\ \big(\mathrm{pred}(-z_l), -0\big), & \text{otherwise;} \end{cases}$$

$$a_8 = \begin{cases} +0, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u = z_l; \\ x_u \boxminus_\downarrow z_l, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u \neq z_l; \\ [\![e_u]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\downarrow = [e_u]_\downarrow; \\ [\![e_u]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\uparrow < [e_u]_\uparrow; \\ \mathrm{pred}\big([\![e_u]\!]_\uparrow\big), & \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ \mathrm{pred}\big(\mathrm{succ}(x_u) \boxminus_\uparrow z_l\big), & \text{if } \bar{r}_u = \downarrow; \\ +0, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u = z_l; \\ x_u \boxminus_\downarrow z_l, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u \neq z_l. \end{cases}$$

Fig. 4. Inverse projection of addition: function ia_u.

interval-consistency based filtering. However, the opposite phenomenon is also possible. Consider again $X = [1.0, 2.0]$ $Z = [1.0, 5.0]$. Filtering by maximum ULP projection gives $Z = [-1.1 \cdots 1 \times 2^{24}, 1.0 \times 2^{25}]$; in contrast, our inverse projection exploits the available information on Z to obtain $Y = [-4, 1.0 \cdots 01]$. As we already stated, our filtering and maximum ULP filtering should both be applied in order to obtain precise results.

Exploiting the commutative property of addition, the refinement Z' of Z can be defined analogously.

Division. In this section we deal with constraints of the form $x = y \boxslash_S z$ with $S \subseteq R$.

Direct Propagation. For direct propagation, interval Z is partitioned into the sign-homogeneous intervals $Z_- \stackrel{\text{def}}{=} Z \cap [-\infty, -0]$ and $Z_+ \stackrel{\text{def}}{=} Z \cap [+0, +\infty]$. This is needed because the sign of operand z determines the monotonicity with respect to y, and therefore the interval bounds to be used for propagation depend on it. Hence, once Z has been partitioned into sign-homogeneous intervals, we use the

interval Y and $W = Z_-$, to obtain the new interval $[x_l^-, x_u^-]$, and Y and $W = Z_+$, to obtain $[x_l^+, x_u^+]$. The appropriate bounds for interval propagation are chosen by function τ of Figure 5. Note that the sign of z is, by construction, constant over interval W. The selected values are then taken as arguments by functions dd_l and dd_u of Figure 6, which return the correct bounds for the aforementioned new intervals for X. The intervals $X \cap [x_l^-, x_u^-]$ and $X \cap [x_l^+, x_u^+]$ are eventually joined using convex union, denoted by $\biguplus$, to obtain the refining interval X'.

$$\tau(y_l, y_u, w_l, w_u) \stackrel{\text{def}}{=} \begin{cases} (y_u, y_l, w_l, w_u), & \text{if } \mathrm{sgn}(w_u) = \mathrm{sgn}(y_u) = -1; \\ (y_u, y_l, w_u, w_l), & \text{if } -\mathrm{sgn}(w_u) = \mathrm{sgn}(y_l) = 1; \\ (y_u, y_l, w_u, w_u), & \text{if } -\mathrm{sgn}(w_u) = -\mathrm{sgn}(y_l) = \mathrm{sgn}(y_u) = 1; \\ (y_l, y_u, w_l, w_u), & \text{if } -\mathrm{sgn}(w_l) = \mathrm{sgn}(y_u) = -1; \\ (y_l, y_u, w_u, w_l), & \text{if } \mathrm{sgn}(w_l) = \mathrm{sgn}(y_l) = 1; \\ (y_l, y_u, w_l, w_l), & \text{if } \mathrm{sgn}(w_l) = -\mathrm{sgn}(y_l) = \mathrm{sgn}(y_u) = 1. \end{cases}$$

Fig. 5. Direct projection of division: the function τ; assumes $\mathrm{sgn}(w_l) = \mathrm{sgn}(w_u)$

$\mathrm{dd}_l(y_L, w_L, r_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$+\infty$	$-\infty$	$-\infty$	-0
$\mathbb{R}_-$	$+0$	$y_L \boxslash_{r_l} w_L$	$+\infty$	$-\infty$	$y_L \boxslash_{r_l} w_L$	-0
-0	$+0$	$+0$	$+\infty$	-0	-0	-0
$+0$	-0	-0	-0	$+\infty$	$+0$	$+0$
$\mathbb{R}_+$	-0	$y_L \boxslash_{r_l} w_L$	$-\infty$	$+\infty$	$y_L \boxslash_{r_l} w_L$	$+0$
$+\infty$	-0	$-\infty$	$-\infty$	$+\infty$	$+\infty$	$+\infty$

$\mathrm{dd}_u(y_U, w_U, r_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+0$	$+\infty$	$+\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$+0$	$y_U \boxslash_{r_u} w_U$	$+\infty$	$-\infty$	$y_U \boxslash_{r_u} w_U$	-0
-0	$+0$	$+0$	$+0$	$-\infty$	-0	-0
$+0$	-0	-0	$-\infty$	$+0$	$+0$	$+0$
$\mathbb{R}_+$	-0	$y_U \boxslash_{r_u} w_U$	$-\infty$	$+\infty$	$y_U \boxslash_{r_u} w_U$	$+0$
$+\infty$	$-\infty$	$-\infty$	$-\infty$	$+\infty$	$+\infty$	$+0$

Fig. 6. Case analyses for direct propagation of division.

It can be proved that Algorithm 3 computes a *correct* and *optimal direct projection*, as ensured by its postconditions.

Theorem 3. Algorithm 3 *satisfies its contract.*

Example 5. Consider $Y = [-0, 42]$, $Z = [-3, 6]$ and any value of S. First, Z is split into $Z_- = [-3, -0]$ and $Z_+ = [+0, 6]$. For the negative interval, the third

Algorithm 3 Direct projection for division constraints.

Require: $x = y \boxslash_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $X' \subseteq X$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxslash_r z \implies x \in X'$ and $\forall X'' \subset X, \exists r \in S, y \in Y, z \in Z : y \boxslash_r z \notin X''$.

$Z_- := Z \cap [-\infty, -0]$;
if $Z_- = [z_l^-, z_u^-] \neq \varnothing$ **then**
  $W := Z_-$;
  $(y_L, y_U, w_L, w_U) := \tau(y_l, y_u, w_l, w_u)$
  $r_l := r_l(S, y_L, \boxslash, w_L)$; $r_u := r_u(S, y_U, \boxslash, w_U)$;
  $x_l^- := \mathrm{dd}_l(y_L, w_L, r_l)$; $x_u^- := \mathrm{dd}_u(y_U, w_U, r_u)$;
else
  $[x_l^-, x_u^-] := \varnothing$;
end if
$X'_- = X \cap [x_l^-, x_u^-]$;
$Z_+ := Z \cap [+0, +\infty]$;
if $Z_+ = [z_l^+, z_u^+] \neq \varnothing$ **then**
  $W := Z_+$;
  $(y_L, y_U, w_L, w_U) := \tau(y_l, y_u, w_l, w_u)$
  $r_l := r_l(S, y_L, \boxslash, w_L)$; $r_u := r_u(S, y_U, \boxslash, w_U)$;
  $x_l^+ := \mathrm{dd}_l(y_L, w_L, r_l)$; $x_u^+ := \mathrm{dd}_u(y_U, w_U, r_u)$;
else
  $[x_l^+, x_u^+] := \varnothing$;
end if
$X'_+ = X \cap [x_l^+, x_u^+]$;
$X' := X'_- \uplus X'_+$;

case of $\tau(-0, 42, -3, -0)$ applies, yielding $(y_L, y_U, w_L, w_U) = (42, -0, -0, -0)$. Then, the projection functions are invoked, and we have $\mathrm{dd}_l(42, -0, r_l) = -\infty$ and $\mathrm{dd}_u(-0, -0, r_u) = +0$, i.e., $[x_l^-, x_u^-] = [-\infty, +0]$. For the positive part, we have $\tau(-0, 42, +0, 6) = (-0, 42, +0, +0)$ (sixth case). From the projections we obtain $\mathrm{dd}_l(-0, +0, r_l) = -0$ and $\mathrm{dd}_u(42, +0, r_u) = +\infty$, and $[x_l^+, x_u^+] = [-0, +\infty]$. Finally, $X' = [x_l^-, x_u^-] \uplus [x_l^+, x_u^+] = [-\infty, +\infty]$.

Inverse Propagation (First Projection). The inverse projections of division must be handled separately for each operand. The projection on y is the *first* inverse projection. This case requires, as explained for Algorithm 3, to split Z into the sign-homogeneous intervals $Z_- \stackrel{\text{def}}{=} Z \cap [-\infty, -0]$ and $Z_+ \stackrel{\text{def}}{=} Z \cap [+0, +\infty]$. Then, in order to select the extrema that determine the appropriate lower and upper bound for y, function σ of Figure 7 is applied.

$$\sigma(z_l, z_u, x_l, x_u) \stackrel{\text{def}}{=} \begin{cases} (z_l, z_u, x_l, x_u), & \text{if } \mathrm{sgn}(z_l) = \mathrm{sgn}(x_l) = 1; \\ (z_u, z_l, x_l, x_u), & \text{if } \mathrm{sgn}(z_l) = -\mathrm{sgn}(x_u) = 1; \\ (z_u, z_u, x_l, x_u), & \text{if } \mathrm{sgn}(z_l) = -\mathrm{sgn}(x_l) = \mathrm{sgn}(x_u) = 1; \\ (z_u, z_l, x_u, x_l), & \text{if } \mathrm{sgn}(z_u) = \mathrm{sgn}(x_u) = -1; \\ (z_l, z_u, x_u, x_l), & \text{if } -\mathrm{sgn}(z_u) = \mathrm{sgn}(x_l) = 1; \\ (z_l, z_l, x_u, x_l), & \text{if } -\mathrm{sgn}(z_u) = -\mathrm{sgn}(x_l) = \mathrm{sgn}(x_u) = 1. \end{cases}$$

Fig. 7. First inverse projection of division: the function σ; assumes $\mathrm{sgn}(z_l) = \mathrm{sgn}(z_u)$

Example 6. Suppose $X = [-42, +0]$, $Z = [-1.0 \times 2^{100}, -0]$ and $S = \{\mathrm{n}\}$. In this case, $Z_- = Z$, and $z_+ = \varnothing$. We obtain $\sigma(-1.0 \times 2^{100}, -0, -42, +0) = (-1.0 \times 2^{100}, -1.0 \times 2^{100}, +0, -42)$ from the sixth case of σ. Then, $\mathrm{id}_l^f(+0, -1.0 \times 2^{100}, \mathrm{n}) = (f_{\min} \boxdot_\uparrow (-1.0 \times 2^{100}))/2 = -1.0 \times 2^{-50}$, because the lowest value of y_l is obtained when a division by -1.0×2^{100} underflows. Moreover, $\mathrm{id}_u^f(-42, -1.0 \times 2^{100}, \mathrm{n}) = 1.0101 \times 2^{105}$. Therefore, the projected interval is $Y' = [-1.0 \times 2^{-50}, 1.0101 \times 2^{105}]$.

The following result assures us that Algorithm 4 computes a *correct first inverse projection*, as ensured by its postcondition.

Theorem 4. Algorithm 4 *satisfies its contract.*

Once again, in order to obtain more precise results in some cases, the first inverse projection for division has to be intersected with a filter based on an extension of the Michel and Marre property originally proposed in [MM10] and extended to multiplication and division in [BCGG16]. Indeed, when interval X does not contain zeroes and interval Z contains zeros and infinities, the proposed filtering by maximum ULP algorithm is able to derive more precise bounds than

Algorithm 4 First inverse projection for division constraints.

Require: $x = y \boxslash_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Y' \subseteq Y$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxslash_r z \implies y \in Y'$.

$Z_- := Z \cap [-\infty, -0]$;
if $Z_- = [z_l^-, z_u^-] \neq \varnothing$ **then**
  $W := Z_-$;
  $(w_L, w_U, x_L, x_U) := \sigma(w_l, w_u, x_l, x_u)$
  $\bar{r}_l := \bar{r}_l^{\mathrm{I}}(S, x_L, \boxslash, w_L)$; $\bar{r}_u := \bar{r}_u^{\mathrm{I}}(S, x_U, \boxslash, w_U)$;
  $y_l^- := \mathrm{id}_l^f(x_L, w_L, \bar{r}_l)$; $y_u^- := \mathrm{id}_u^f(x_U, w_U, \bar{r}_u)$;
  if $y_l^- \in \mathbb{F}$ and $y_u^- \in \mathbb{F}$ **then**
    $Y'_- = Y \cap [y_l^-, y_u^-]$;
  else
    $Y'_- = \varnothing$;
  end if
else
  $Y'_- = \varnothing$;
end if
$Z_+ := Z \cap [+0, +\infty]$;
if $Z_+ = [z_l^+, z_u^+] \neq \varnothing$ **then**
  $W := Z_+$;
  $(w_L, w_U, x_L, x_U) := \sigma(w_l, w_u, x_l, x_u)$;
  $\bar{r}_l := \bar{r}_l^{\mathrm{I}}(S, x_L, \boxslash, w_L)$; $\bar{r}_u := \bar{r}_u^{\mathrm{I}}(S, x_U, \boxslash, w_U)$;
  $y_l^+ := \mathrm{id}_l^f(x_L, w_L, \bar{r}_l)$; $y_u^+ := \mathrm{id}_u^f(x_U, w_U, \bar{r}_u)$;
  if $y_l^+ \in \mathbb{F}$ and $y_u^+ \in \mathbb{F}$ **then**
    $Y'_+ = Y \cap [y_l^+, y_u^+]$;
  else
    $Y'_+ = \varnothing$;
  end if
else
  $Y'_+ = \varnothing$;
end if
$Y' := Y'_- \uplus Y'_+$;

$\mathrm{id}_l^f(x_L, w_L, \bar{r}_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	unsat.	a_4	$f_{\min}$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	unsat.	a_3^-	$f_{\min}$	$f_{\min}$	a_3^+	$-f_{\max}$
-0	$+0$	$+0$	$+0$	$f_{\min}$	a_7	$-f_{\max}$
$+0$	$-f_{\max}$	a_6	$f_{\min}$	$+0$	$+0$	$+0$
$\mathbb{R}_+$	$-f_{\max}$	a_3^-	$f_{\min}$	$f_{\min}$	a_3^+	unsat.
$+\infty$	$-\infty$	$-\infty$	$-\infty$	$f_{\min}$	a_5	unsat.

$$e_l^+ \equiv (x_L + \nabla_2^{\mathrm{n}-}(x_L)/2) \cdot w_L;$$

$$a_3^+ = \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow > [e_l^+]_\uparrow; \\ \mathrm{succ}\big([\![e_l^+]\!]_\downarrow\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ x_L \boxdot_\uparrow w_L, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}\big(\mathrm{pred}(x_L) \boxdot_\downarrow w_L\big), & \text{if } \bar{r}_l = \uparrow; \end{cases}$$

$$e_l^- \equiv (x_L + \nabla_2^{\mathrm{n}+}(x_L)/2) \cdot w_L;$$

$$a_3^- = \begin{cases} [\![e_l^-]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^-]\!]_\uparrow = [e_l^-]_\uparrow; \\ [\![e_l^-]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^-]\!]_\uparrow > [e_l^-]_\uparrow; \\ \mathrm{succ}\big([\![e_l^-]\!]_\downarrow\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ x_L \boxdot_\uparrow w_L, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}\big(\mathrm{succ}(x_L) \boxdot_\downarrow w_L\big), & \text{if } \bar{r}_l = \downarrow; \end{cases}$$

$$e_l^1 \equiv (-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2) \cdot w_L;$$

$$a_4 = \begin{cases} +\infty, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}(-f_{\max} \boxdot_\downarrow w_L), & \text{if } \bar{r}_l = \downarrow; \\ [\![e_l^1]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [e_l^1]_\uparrow = [\![e_l^1]\!]_\uparrow; \\ [\![e_l^1]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \end{cases}$$

$$e_l^2 \equiv (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2) \cdot w_L;$$

$$a_5 = \begin{cases} +\infty, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}(f_{\max} \boxdot_\downarrow w_L), & \text{if } \bar{r}_l = \uparrow; \\ [\![e_l^2]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [e_l^2]_\uparrow = [\![e_l^2]\!]_\uparrow; \\ [\![e_l^2]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \end{cases}$$

$$(a_6, a_7) = \begin{cases} (-0,\ \mathrm{succ}(-f_{\min} \boxdot_\downarrow w_L)), & \text{if } \bar{r}_l = \uparrow; \\ (\mathrm{succ}(f_{\min} \boxdot_\downarrow w_L),\ -0), & \text{if } \bar{r}_l = \downarrow; \\ ((f_{\min} \boxdot_\uparrow w_L)/2,\ (-f_{\min} \boxdot_\uparrow w_L)/2)), & \text{if } \bar{r}_l = \mathrm{n}. \end{cases}$$

Fig. 8. First inverse projection of division: function id_l^f.

$\mathrm{id}_u^f(x_U, w_U, \bar{r}_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$+\infty$	$-f_{\min}$	a_9	unsat.
$\mathbb{R}_-$	$f_{\max}$	a_8^-	$-f_{\min}$	$-f_{\min}$	a_8^+	unsat.
-0	$f_{\max}$	a_{12}	$-f_{\min}$	-0	-0	-0
$+0$	-0	-0	-0	$-f_{\min}$	a_{11}	$f_{\max}$
$\mathbb{R}_+$	unsat.	a_8^-	$-f_{\min}$	$-f_{\min}$	a_8^+	$f_{\max}$
$+\infty$	unsat.	a_{10}	$-f_{\min}$	$+\infty$	$+\infty$	$+\infty$

$$e_u^+ \equiv (x_U + \nabla_2^{\mathrm{n}+}(x_U)/2) \cdot w_U;$$

$$a_8^+ = \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow; \\ \mathrm{pred}\big([\![e_u^+]\!]_\uparrow\big), & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ \mathrm{pred}\big(\mathrm{succ}(x_U) \boxdot_\uparrow w_U\big), & \text{if } \bar{r}_u = \downarrow; \\ x_U \boxdot_\downarrow w_U, & \text{if } \bar{r}_u = \uparrow; \end{cases}$$

$$e_u^- \equiv (x_U + \nabla_2^{\mathrm{n}-}(x_U)/2) \cdot w_U;$$

$$a_8^- = \begin{cases} [\![e_u^-]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^-]\!]_\downarrow = [e_u^-]_\downarrow; \\ [\![e_u^-]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^-]\!]_\downarrow < [e_u^-]_\downarrow; \\ \mathrm{pred}\big([\![e_u^-]\!]_\uparrow\big), & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ \mathrm{pred}\big(\mathrm{pred}(x_U) \boxdot_\uparrow w_U\big), & \text{if } \bar{r}_u = \uparrow; \\ x_U \boxdot_\downarrow w_U, & \text{if } \bar{r}_u = \downarrow; \end{cases}$$

$$e_u^1 \equiv (-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2) \cdot w_U;$$

$$a_9 = \begin{cases} -\infty, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}(-f_{\max} \boxdot_\uparrow w_U), & \text{if } \bar{r}_u = \downarrow; \\ [\![e_u^1]\!]_\downarrow & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [e_u^1]_\downarrow = [\![e_u^1]\!]_\downarrow; \\ [\![e_u^1]\!]_\uparrow & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \end{cases}$$

$$e_u^2 \equiv (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2) \cdot w_U;$$

$$a_{10} = \begin{cases} -\infty, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}(f_{\max} \boxdot_\uparrow w_U), & \text{if } \bar{r}_u = \uparrow; \\ [\![e_u^2]\!]_\downarrow & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [e_u^2]_\downarrow = [\![e_u^2]\!]_\downarrow; \\ [\![e_u^2]\!]_\uparrow & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \end{cases}$$

$$(a_{11}, a_{12}) = \begin{cases} (+0,\ \mathrm{pred}(-f_{\min} \boxdot_\uparrow w_U)), & \text{if } \bar{r}_u = \uparrow; \\ (\mathrm{pred}(f_{\min} \boxdot_\uparrow w_U),\ +0), & \text{if } \bar{r}_u = \downarrow; \\ ((f_{\min} \boxdot_\downarrow w_U)/2,\ (-f_{\min} \boxdot_\downarrow w_U)/2), & \text{if } \bar{r}_u = \mathrm{n}. \end{cases}$$

Fig. 9. First inverse projection of division: function id_u^f.

the ones obtained with the inverse projection we are proposing. Thus, for division (and for multiplication as well), the indirect projection and filtering by maximum ULP are mutually exclusive: one applies when the other cannot derive anything useful [BCGG16].

Example 7. Consider the IEEE 754 single-precision constraint $x = y \boxslash_S z$ with initial intervals $X = [-1.0 \times 2^{-110}, -1.0 \times 2^{-121}]$ and $Y = Z = [-\infty, +\infty]$. When $S = \{\mathrm{n}\}$, filtering by maximum ULP results into the possible refinement $Y' = [-1.1\ldots1 \times 2^{17}, 1.1\ldots1 \times 2^{17}]$, while Algorithm 4 would return the less precise $Y' = [-f_{\max}, f_{\max}]$, with any rounding mode.

Inverse Propagation (Second Projection). The second inverse projection for division computes a new interval for operand z. For this projection, we need to partition interval X into sign-homogeneous intervals $X_- \stackrel{\text{def}}{=} X \cap [-\infty, -0]$ and $X_+ \stackrel{\text{def}}{=} X \cap [+0, +\infty]$ since, in this case, it is the sign of X that matters for deriving correct bounds for Z. Once X has been partitioned, we use intervals X_- and Y to obtain the interval $[z_l^-, z_u^-]$; intervals X_+ and Y to obtain $[z_l^+, z_u^+]$. The new bounds for z are computed by functions id_l^s of Figure 10 and id_u^s of Figure 11, after the appropriate interval extrema of Y and $V = X_-$ (or $V = X_+$) have been selected by function τ. The intervals $Z \cap [z_l^-, z_u^-]$ and $Z \cap [z_l^+, z_u^+]$ will be then joined with convex union to obtain Z'.

Our algorithm computes a *correct second inverse projection.*

Theorem 5. Algorithm 5 *satisfies its contract.*

Example 8. Consider $X = [6, +\infty]$, $Y = [+0, 42]$ and $S = \{\mathrm{n}\}$. In this case, we only have $X_+ = X$, and $X_- = \varnothing$. With this input, $\tau(+0, 42, 6, +\infty) = (+0, 42, +\infty, 6)$ (case 5). Therefore, we obtain $\mathrm{id}_l^s(+0, +\infty, \mathrm{n}) = +0$, because any number in Y except $+0$ yields $+\infty$ when divided by $+0$. If we compute intermediate values exactly, $\mathrm{id}_u^s(42, 6) = 7$ and the refined interval is $Z' = [+0, 7]$. If not, then $z'_u = 1.110\cdots01 \times 2^2 = \mathrm{succ}(7)$.

In order to obtain more precise results, also the result of our second inverse projection can be intersected with the interval obtained by the maximum ULP filter proposed in [BCGG16] . Indeed, when interval X does not contain zeros and interval Y contains zeros and infinities, the proposed filtering by maximum ULP algorithm is able to derive tighter bounds than those obtained with the inverse projection presented in this work.

Example 9. Consider the IEEE 754 single-precision division constraint $x = y \boxslash_S z$ with initial intervals $x \in [1.0\cdots010 \times 2^{110}, 1.0 \times 2^{121}]$ and $Y = Z = [-\infty, +\infty]$. When $S = \{\mathrm{n}\}$, filtering by maximum ULP results into the possible refinement $Z' = [-1.0 \times 2^{18}, 1.0 \times 2^{18}]$, while Algorithm 5 would compute $Z' = [-f_{\max}, f_{\max}]$, regardless of the rounding mode.

Algorithm 5 Second inverse projection for division constraints.

Require: $x = y \boxdot_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Z' \subseteq Z$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxdot_r z \implies z \in Z'$.

$X_- := X \cap [-\infty, -0]$;
if $X_- \neq \varnothing$ **then**
  $V := X_-$;
  $(y_L, y_U, v_L, v_U) := \tau(y_l, y_u, v_l, v_u)$
  $\bar{r}_l := \bar{r}^{\mathrm{r}}_r(S, v_L, \boxslash, y_L)$; $\bar{r}_u := \bar{r}^{\mathrm{r}}_u(S, v_U, \boxslash, y_U)$;
  $z_l^- := \mathrm{id}^s_l(y_L, v_L, \bar{r}_l)$; $z_u^- := \mathrm{id}^s_u(y_U, v_U, \bar{r}_u)$;
  if $z_l^- \in \mathbb{F}$ and $z_u^- \in \mathbb{F}$ **then**
    $Z'_- = Z \cap [z_l^-, z_u^-]$;
  else
    $Z'_- = \varnothing$;
  end if
else
  $Z'_- = \varnothing$;
end if
$X_+ := X \cap [+0, +\infty]$;
if $X_+ \neq \varnothing$ **then**
  $V := X_+$;
  $(y_L, y_U, v_L, v_U) := \tau(y_l, y_u, v_l, v_u)$
  $\bar{r}_l := \bar{r}^{\mathrm{r}}_r(S, v_L, \boxslash, y_L)$; $\bar{r}_u := \bar{r}^{\mathrm{r}}_u(S, v_U, \boxslash, y_U)$;
  $z_l^+ := \mathrm{id}^s_l(y_L, v_L, \bar{r}_l)$; $z_u^+ := \mathrm{id}^s_u(y_U, v_U, \bar{r}_u)$;
  if $z_l^+ \in \mathbb{F}$ and $z_u^+ \in \mathbb{F}$ **then**
    $Z'_+ = Z \cap [z_l^+, z_u^+]$;
  else
    $Z'_+ = \varnothing$;
  end if
else
  $Z'_+ = \varnothing$;
end if
$Z' := Z'_- \uplus Z'_+$;

$\mathrm{id}_l^s(y_L, v_L, \bar{r}_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+0$	unsat.	unsat.	$-\infty$	$-f_{\max}$	$-f_{\max}$
$\mathbb{R}_-$	$+0$	a_3^-	a_4	$-\infty$	a_3^-	a_6
-0	$+0$	$f_{\min}$	$f_{\min}$	$-\infty$	$+0$	$+0$
$+0$	$+0$	$+0$	$-\infty$	$f_{\min}$	$f_{\min}$	$+0$
$\mathbb{R}_+$	a_7	a_3^+	$-\infty$	a_5	a_3^+	$+0$
$+\infty$	$-f_{\max}$	$-f_{\max}$	$-\infty$	unsat.	unsat.	$+0$

$$e_l^+ \equiv y_L/(v_L + \nabla_2^{\mathrm{n}+}(v_L)/2);$$

$$a_3^+ = \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(v_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(v_L) \text{ and } [\![e_l^+]\!]_\uparrow > [e_l^+]_\uparrow; \\ \mathrm{succ}([\![e_l^+]\!]_\downarrow) & \text{if } \bar{r}_l = \mathrm{n}, \text{otherwise}; \\ y_L \boxslash_\uparrow v_L, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}\big(y_L \boxslash_\downarrow \mathrm{succ}(v_L)\big), & \text{if } \bar{r}_l = \downarrow; \end{cases}$$

$$e_l^- \equiv y_L/(v_L + \nabla_2^{\mathrm{n}-}(v_L)/2);$$

$$a_3^- = \begin{cases} [\![e_l^-]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(v_L) \text{ and } [\![e_l^-]\!]_\uparrow = [e_l^-]_\uparrow; \\ [\![e_l^-]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(v_L) \text{ and } [\![e_l^-]\!]_\uparrow > [e_l^-]_\uparrow; \\ \mathrm{succ}([\![e_l^-]\!]_\downarrow) & \text{if } \bar{r}_l = \mathrm{n}, \text{otherwise}; \\ y_L \boxslash_\uparrow v_L, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}\big(y_L \boxslash_\downarrow \mathrm{pred}(v_L)\big), & \text{if } \bar{r}_l = \uparrow; \end{cases}$$

$$(a_4, a_5) = \begin{cases} (+\infty, \mathrm{succ}(y_L \boxslash_\downarrow f_{\min})), & \text{if } \bar{r}_l = \downarrow; \\ (\mathrm{succ}(y_L \boxslash_\downarrow -f_{\min}), +\infty), & \text{if } \bar{r}_l = \uparrow; \\ ((y_L \boxslash_\uparrow -f_{\min}) \cdot 2, (y_L \boxslash_\uparrow f_{\min}) \cdot 2), & \text{otherwise}; \end{cases}$$

$$e_l^1 \equiv y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2);$$

$$a_6 = \begin{cases} -0, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}(y_L \boxslash_\downarrow f_{\max}), & \text{if } \bar{r}_l = \uparrow; \\ [\![e_l^1]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [e_l^1]_\uparrow = [\![e_l^1]\!]_\uparrow; \\ [\![e_l^1]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{otherwise}; \end{cases}$$

$$e_l^2 \equiv y_L/(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2);$$

$$a_7 = \begin{cases} -0, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}(y_L \boxslash_\downarrow -f_{\max}), & \text{if } \bar{r}_l = \downarrow; \\ [\![e_l^2]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [e_l^2]_\uparrow = [\![e_l^2]\!]_\uparrow; \\ [\![e_l^2]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{otherwise}. \end{cases}$$

Fig. 10. Second inverse projection of division: function id_l^s.

$\mathrm{id}_u^s(y_U, v_U, \bar{r}_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$f_{\max}$	$f_{\max}$	$+\infty$	unsat.	unsat.	-0
$\mathbb{R}_-$	a_{11}	a_8^-	$+\infty$	a_9	a_8^-	-0
-0	-0	-0	$+\infty$	$-f_{\min}$	$-f_{\min}$	-0
$+0$	-0	$-f_{\min}$	$-f_{\min}$	$+\infty$	-0	-0
$\mathbb{R}_+$	-0	a_8^+	a_{10}	$+\infty$	a_8^+	a_{12}
$+\infty$	-0	unsat.	unsat.	$+\infty$	$f_{\max}$	$f_{\max}$

$$e_u^+ \equiv y_U/(v_U + \nabla_2^{\mathrm{n}-}(v_U)/2);$$

$$a_8^+ = \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(v_U) \text{ and } [\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(v_U) \text{ and } [\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow; \\ \mathrm{pred}([\![e_u^+]\!]_\uparrow) & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ y_U \boxslash_\downarrow v_U, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}\big(y_U \boxslash_\uparrow \mathrm{pred}(v_U)\big), & \text{if } \bar{r}_u = \uparrow; \end{cases}$$

$$e_u^- \equiv y_U/(v_U + \nabla_2^{\mathrm{n}+}(v_U)/2);$$

$$a_8^- = \begin{cases} [\![e_u^-]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(v_U) \text{ and } [\![e_u^-]\!]_\downarrow = [e_u^-]_\downarrow; \\ [\![e_u^-]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(v_U) \text{ and } [\![e_u^-]\!]_\downarrow < [e_u^-]_\downarrow; \\ \mathrm{pred}([\![e_u^-]\!]_\uparrow) & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ y_U \boxslash_\downarrow v_U, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}\big(y_U \boxslash_\uparrow \mathrm{succ}(v_U)\big), & \text{if } \bar{r}_u = \downarrow; \end{cases}$$

$$(a_9, a_{10}) = \begin{cases} (-\infty, \mathrm{pred}(y_U \boxslash_\uparrow -f_{\min})), & \text{if } \bar{r}_u = \uparrow; \\ (\mathrm{pred}(y_U \boxslash_\uparrow f_{\min}), -\infty), & \text{if } \bar{r}_u = \downarrow; \\ ((y_U \boxslash_\downarrow f_{\min}) \cdot 2, (y_U \boxslash_\downarrow -f_{\min}) \cdot 2), & \text{otherwise;} \end{cases}$$

$$e_u^1 \equiv y_U/(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2);$$

$$a_{11} = \begin{cases} +0, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}(y_U \boxslash_\uparrow -f_{\max}), & \text{if } \bar{r}_u = \downarrow; \\ [\![e_u^1]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [e_u^1]_\downarrow = [\![e_u^1]\!]_\downarrow; \\ [\![e_u^1]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \end{cases}$$

$$e_u^2 \equiv y_U/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2);$$

$$a_{12} = \begin{cases} +0, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}(y_U \boxslash_\uparrow f_{\max}), & \text{if } \bar{r}_u = \uparrow; \\ [\![e_u^2]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [e_u^2]_\downarrow = [\![e_u^2]\!]_\downarrow; \\ [\![e_u^2]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise.} \end{cases}$$

Fig. 11. Second inverse projection of division: function id_u^s.

5 Discussion and Conclusion

With the increasing use of floating-point computations in mission- and safety-critical settings, the issue of reliably verifying their correctness has risen to a point in which testing or other informal techniques are not acceptable any more. Indeed, this phenomenon has been fostered by the wide adoption of the IEEE 754 floating-point format, which has significantly simplified the use of floating-point numbers, by providing a precise, sound, and reasonably cross-platform specification of floating-point representations, operations and their semantics. The approach we propose in this paper exploits these solid foundations to enable a wide range of floating-point program verification techniques. It is based on the solution of constraint satisfaction problems by means of interval-based constraint propagation, which is enabled by the filtering algorithms we presented. These algorithms cover the whole range of possible floating-point values, including symbolic values, with respect to interval-based reasoning. Moreover, they not only support all IEEE 754 available rounding-modes, but they also allow to take care of uncertainty on the rounding-mode in use. Some important implementation aspects are also taken into account, by allowing both the use of machine floating-point arithmetic for all computations (for increased performance), and of extended-precision arithmetic (for better precision with the round-to-nearest rounding mode). In both cases, correctness is guaranteed, so that no valid solutions can erroneously removed from the constraint system. This is supported by the extensive correctness proofs of all algorithms and tables, which allow us to claim that neither false positives, nor false negatives may be produced.

As we reported in Section 1.3, we implemented our work in the commercial tool ECLAIR. While the initial results on a wide range of self-developed tests looked very promising, we wanted to compare them with the competing tools presented in the literature, in order to better asses the strength of our approach with respect to the state of the art. Unfortunately, most of these tools were either unavailable, or not sufficiently equipped to analyze real-world C/C++ programs. We could, however, do a comparison with the results obtained in [WLZ17]. It presents a tool called seVR-fpe, for floating-point exception detection based on symbolic execution and value-range analysis. The same task can be carried out by the constraint-based symbolic model checker we included in ECLAIR. The authors of seVR-fpe tested their tool both on a self-developed benchmark suite and on real-world programs. Upon contacting them, they were unfortunately unable to provide us with more detailed data regarding their analysis of real world programs. This prevents us from doing an in-depth comparison of the tools, since we only know the total number of bugs found, but not their exact nature and location. Data with this level of detail was instead available for (most of) their self-developed benchmarks. The results obtained by running ECLAIR on them are reported in Table 1. ECLAIR could find a number of possible bugs significantly higher than seVR-fpe. As expected, due to the provable correctness of the algorithms employed in ECLAIR, no false positives were detected among the inputs it generated. This confirms the solid results obtainable by means of the algorithms presented in this paper.

	ECLAIR	seVR-fpe	Difference
total	135	66	69
overflow	55	26	29
underflow	30	13	17
invalid	47	8	39
divbyzero	3	3	0
false positives	0	15	-15

Table 1. Number of possible exceptions found by ECLAIR and seVR-fpe on the self-developed benchmarks of [WLZ17].

Several aspects of the constraint-based verification of floating-point programs remain, however, open problems, both from a theoretical and a practical perspective. As we showed throughout the paper, the filtering algorithms we presented are not optimal, i.e., they may not yield the tightest possible intervals containing all solutions to the constraint system. They must be interleaved with the filtering algorithms of [BCGG16], and they may require multiple passes before reaching the maximum degree of variable-domain pruning they are capable of. Therefore, the next possible advance in this direction would be conceiving optimal filtering algorithms, that reduce variable domains to intervals as tight as possible with a single application.

However, filtering algorithms only represent a significant, but to some extent limited, part of the constraint solving process. Indeed, even an optimally pruned interval may contain values that are not solutions to the constraint system, due to the possible non-linearity thereof. If the framework in use supports multi-intervals, this issue is dealt with by means of labeling techniques: when a constraint-solving process reaches quiescence, i.e., the application of filtering algorithms fails to prune variable domains any further, such intervals are split into two or more sub-intervals, and the process continues on each partition separately. In this context, the main issues are *where* to split intervals, and in *how many* parts. These issues are currently addressed with heuristic labeling strategies. Indeed, significant improvements to the constraint-propagation process could be achieved by investigating better labeling strategies. To this end, possible advancements would include the identification of objective criteria for the evaluation of labeling strategies on floating point-numbers, and the conception of labeling strategies tailored to the properties of constraint systems most commonly generated by numeric programs.

In conclusion, we believe the work presented in this paper can be an extensive reference for the readers interested in realizing applications for formal reasoning on floating-point computations, as well as a solid foundation for further improvements in the state of the art.

Acknowledgments

The authors express their gratitude to Roberto Amadini, Isacco Cattabiani and Laura Savino for their careful reading of early versions of this paper.

References

BCGB16. R. Bagnara, M. Chiari, R. Gori, and A. Bagnara, *A practical approach to interval refinement for `math.h/cmath` functions*, Report `arXiv:1610.07390v1 [cs.PL]`, 2016, Available at `http://arxiv.org/`.

BCGB17. ———, *A practical approach to interval refinement for `math.h/cmath` functions*, ACM Transaction on Mathematical Software (2017), Submitted to pubblication.

BCGG13. R. Bagnara, M. Carlier, R. Gori, and A. Gotlieb, *Symbolic path-oriented test data generation for floating-point programs*, Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation (Luxembourg City, Luxembourg), IEEE Press, 2013.

BCGG16. ———, *Exploiting binary floating-point representations for constraint propagation*, INFORMS Journal on Computing **28** (2016), no. 1, 31–46.

BGM06. B. Botella, A. Gotlieb, and C. Michel, *Symbolic execution of floating-point computations*, Software Testing, Verification and Reliability **16** (2006), no. 2, 97–121.

BVLS13. Earl T. Barr, Thanh Vo, Vu Le, and Zhendong Su, *Automatic detection of floating-point exceptions*, The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013, 2013, pp. 549–560.

CC77. P. Cousot and R. Cousot, *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*, Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages (Los Angeles, CA, USA), ACM Press, 1977, pp. 238–252.

CKVDV02. A. Cuyt, P. Kuterna, B. Verdonk, and D. D. Verschaeren, *Underflow revisited*, CALCOLO **39** (2002), no. 3, 169–179.

DGP+09. David Delmas, Eric Goubault, Sylvie Putot, Jean Souyris, Karim Tekkal, and Franck Védrine, *Towards an industrial use of FLUCTUAT on safety-critical avionics software*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5825 LNCS, Springer, Berlin, Heidelberg, 2009, pp. 53–69.

Fie10. G. Fiedler, *Floating point determinism*, Gaffer On Games Blog, February 24, 2019, `https://gafferongames.com/post/floating_point_determinism/`, 2010, Last accessed on April 16th, 2018.

GBR98. A. Gotlieb, B. Botella, and M. Rueher, *Automatic test data generation using constraint solving techniques*, Proceedings of the 1998 ACM SIGSOFT International Symposium on Software Testing and Analysis (New York, NY, USA), ISSTA '98, ACM, 1998, pp. 53–62.

GBR00. ———, *A CLP framework for computing structural test data*, Computational Logic — CL 2000: First International Conference London, UK, July 24–28, 2000 Proceedings, Springer Berlin Heidelberg, 2000, pp. 399–413.

GWBC18. D. Gallois-Wong, S. Boldo, and P. Cuoq, *Optimal inverse projection of floating-point addition*, Tech. Report 01939097, HAL-Inria, 2018, Available at `https://hal.inria.fr/hal-01939097`, last accessed on February 1st, 2019.

IEE08. The Institute of Electrical and Electronics Engineers, Inc., *IEEE standard for floating-point arithmetic*, IEEE Std 754-2008 (revision of IEEE Std 754-1985) ed., August 2008, Available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

KLLM09. P. Kornerup, V. Lefevre, N. Louvet, and J.-M. Muller, *On the computation of correctly-rounded sums*, Proceedings of the 19th IEEE Symposium on Computer Arithmetic (ARITH 2009) (Portland, OR, USA), 2009, pp. 155–160.

Mic02. C. Michel, *Exact projection functions for floating point number constraints*, Proceedings of the 7th International Symposium on Artificial Intelligence and Mathematics (Fort Lauderdale, FL, USA), 2002.

MM10. B. Marre and C. Michel, *Improving the floating point addition and subtraction constraints*, Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP 2010) (St. Andrews, Scotland, UK) (D. Cohen, ed.), Lecture Notes in Computer Science, vol. 6308, Springer, 2010, pp. 360–367.

Mon08. D. Monniaux, *The pitfalls of verifying floating-point computations*, ACM Transactions on Programming Languages and Systems **30** (2008), no. 3, 12:1–12:41.

MRL01. Claude Michel, Michel Rueher, and Yahia Lebbah, *Solving constraints over floating-point numbers*, Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings, 2001, pp. 524–538.

RO07. S. M. Rump and T. Ogita, *Super-fast validated solution of linear systems*, Journal of Computational and Applied Mathematics **199** (2007), no. 2, 199–206, Special Issue on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004).

Rum13. S. M. Rump, *Accurate solution of dense linear systems, Part II: Algorithms using directed rounding*, Journal of Computational and Applied Mathematics **242** (2013), 185–212.

Wat08. J. Watte, *Floating point determinism*, GameDev.net Forum, June 30, 2008, `https://www.gamedev.net/community/forums/topic.asp?topic_id=499435`, 2008, Last accessed on April 16th, 2018.

WLZ17. Xingming Wu, Lian Li, and Jian Zhang, *Symbolic execution with value-range analysis for floating-point exception detection*, 24th Asia-Pacific Software Engineering Conference, APSEC 2017, Nanjing, China, December 4-8, 2017, 2017, pp. 1–10.

A Filtering algorithms: Subtraction and Multiplication

A.1 Subtraction.

Here we deal with constraints of the form $x = y \boxminus_S z$.

Assume $X = [x_l, x_u]$, $Y = [y_l, y_u]$ and $Z = [z_l, z_u]$.

Again, thanks to Proposition 1 we need not be concerned with sets of rounding modes, as any such set $S \subseteq R$ can always be mapped to a pair of "worst-case rounding modes" which, in addition are never round-to-zero.

Direct Propagation. For direct propagation, we use Algorithm 6 and functions ds_l and ds_u, as defined in Figure 12.

Algorithm 6 Direct projection for subtraction constraints.

Require: $x = y \boxminus_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $X' \subseteq X$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxminus_r z \implies x \in X'$ and $\forall X'' \subset X, \exists r \in S, y \in Y, z \in Z : y \boxminus_r z \notin X''$.

1: $r_l := r_l(S, y_l, \boxminus, z_u)$; $r_u := r_u(S, y_u, \boxminus, z_l)$;
2: $x'_l := \mathrm{ds}_l(y_l, z_u, r_l)$; $x'_u := \mathrm{ds}_u(y_u, z_l, r_u)$;
3: $X' := X \cap [x'_l, x'_u]$;

Theorem 6. Algorithm 6 *satisfies its contract.*

Inverse Propagation. For inverse propagation, we have to deal with two different cases depending on which variable we are computing: the first inverse projection on y or the second inverse projection on z.

The first inverse projection of subtraction is somehow similar to the direct projection of addition. In this case we define Algorithm 7 and functions is_l^f and is_u^f, as defined in Figure 13 and 14 respectively.

Algorithm 7 First inverse projection for subtraction constraints.

Require: $x = y \boxminus_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Y' \subseteq Y$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxminus_r z \implies y \in Y'$.

1: $\bar{r}_l := \bar{r}_l^1(S, x_l, \boxminus, z_l)$; $\bar{r}_u := \bar{r}_u^1(S, x_u, \boxminus, z_u)$;
2: $y'_l := \mathrm{is}_l^f(x_l, z_l, \bar{r}_l)$; $y'_u := \mathrm{is}_u^f(x_u, z_u, \bar{r}_u)$;
3: **if** $y'_l \in \mathbb{F}$ and $y'_u \in \mathbb{F}$ **then**
4: $\quad Y' := Y \cap [y'_l, y'_u]$;
5: **else**
6: $\quad Y' := \varnothing$;
7: **end if**

Theorem 7. Algorithm 7 *satisfies its contract.*

$\mathrm{ds}_l(y_l, z_u, r_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$+\infty$	$y_l \boxminus_{r_l} z_u$	y_l	y_l	$y_l \boxminus_{r_l} z_u$	$-\infty$
-0	$+\infty$	$-z_u$	a_1	-0	$-z_u$	$-\infty$
$+0$	$+\infty$	$-z_u$	$+0$	a_1	$-z_u$	$-\infty$
$\mathbb{R}_+$	$+\infty$	$y_l \boxminus_{r_l} z_u$	y_l	y_l	$y_l \boxminus_{r_l} z_u$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$$a_1 = \begin{cases} -0, & \text{if } r_l = \downarrow, \\ +0, & \text{otherwise;} \end{cases}$$

$\mathrm{ds}_u(y_u, z_l, r_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$+\infty$	$y_u \boxminus_{r_u} z_l$	y_u	y_u	$y_u \boxminus_{r_u} z_l$	$-\infty$
-0	$+\infty$	$-z_l$	a_2	-0	$-z_l$	$-\infty$
$+0$	$+\infty$	$-z_l$	$+0$	a_2	$-z_l$	$-\infty$
$\mathbb{R}_+$	$+\infty$	$y_u \boxminus_{r_u} z_l$	y_u	y_u	$y_u \boxminus_{r_u} z_l$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$-\infty$

$$a_2 = \begin{cases} -0, & \text{if } r_u = \downarrow, \\ +0, & \text{otherwise.} \end{cases}$$

Fig. 12. Direct projection of subtraction: function ds_l (resp., ds_u); values for y_l (resp., y_u) on rows, values for z_u (resp., z_l) on columns.

The second inverse projection of subtraction is quite similar to the case of direct projection of subtraction. Here we define Algorithm 8 and functions is_l^s and is_u^s, as defined in Figures 15 and 16 respectively.

Theorem 8. Algorithm 8 *is correct.*

Since subtraction is very closely related to addition, the proofs of Theorems 7 and 8 can be obtained by reasoning in the same way as for the projections of addition. Moreover, it is worth noting that in order to obtain more precise results, inverse projections for subtraction need to be intersected with maximum ULP filtering [BCGG16], as in the case of addition.

A.2 Multiplication.

Here we deal with constraints of the form $x = y \boxdot_S z$. As usual, assume $X = [x_l, x_u]$, $Y = [y_l, y_u]$ and $Z = [z_l, z_u]$.

Direct Propagation. For direct propagation, a case analysis is performed in order to select the interval extrema y_L and z_L (resp., y_U and z_U) to be used to compute the new lower (resp., upper) bound for x.

$\mathrm{is}_l^f(x_l, z_l, \bar{r}_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$-f_{\max}$	a_3	x_l	x_l	a_3	unsat.
-0	$-f_{\max}$	z_l	-0	-0	z_l	unsat.
$+0$	$-f_{\max}$	a_4	a_5	a_4	a_4	unsat.
$\mathbb{R}_+$	$-f_{\max}$	a_3	x_l	x_l	a_3	unsat.
$+\infty$	$-f_{\max}$	a_6	$+\infty$	$+\infty$	$+\infty$	unsat.

$$e_l \equiv x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 + z_l;$$

$$a_3 = \begin{cases} -0, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l = -z_l; \\ x_l \boxplus_\uparrow z_l, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l \neq -z_l; \\ [\![e_l]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq -f_{\min} \text{ and } [\![e_l]\!]_\uparrow = [e_l]_\uparrow; \\ [\![e_l]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq -f_{\min} \text{ and } [\![e_l]\!]_\uparrow > [e_l]_\uparrow; \\ \mathrm{succ}\big([\![e_l]\!]_\downarrow\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ -0, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l = -z_l; \\ x_l \boxplus_\uparrow z_l, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l \neq -z_l; \\ \mathrm{succ}\big(\mathrm{pred}(x_l) \boxplus_\downarrow z_l\big), & \text{if } \bar{r}_l = \uparrow; \end{cases}$$

$$(a_4, a_5) = \begin{cases} (\mathrm{succ}(z_l), +0), & \bar{r}_l = \downarrow; \\ (z_l, -0), & \text{otherwise}; \end{cases}$$

$$a_6 = \begin{cases} +\infty, & \bar{r}_l = \downarrow; \\ \mathrm{succ}(f_{\max} \boxplus_\downarrow z_l), & \bar{r}_l = \uparrow; \\ f_{\max} \boxplus_\uparrow \left(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxplus_\uparrow z_l\right), & \text{otherwise}. \end{cases}$$

Fig. 13. First inverse projection of subtraction: function is_l^f.

Algorithm 8 Second inverse projection for subtraction constraints.

Require: $x = y \boxminus_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Z' \subseteq Z$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxminus_r z \implies z \in Z'$.
1: $\bar{r}_l := \bar{r}_l^{\mathrm{r}}(S, x_u, \boxminus, y_l)$; $\bar{r}_u := \bar{r}_u^{\mathrm{r}}(S, x_l, \boxminus, y_u)$;
2: $z'_l := \mathrm{is}_l^s(y_l, x_u, \bar{r}_l)$; $z'_u := \mathrm{is}_u^s(y_u, x_l, \bar{r}_u)$;
3: **if** $z'_l \in \mathbb{F}$ and $z'_u \in \mathbb{F}$ **then**
4: $\quad Z' := Z \cap [z'_l, z'_u]$;
5: **else**
6: $\quad Z' := \varnothing$;
7: **end if**

$\mathrm{is}_u^f(x_u, z_u, \bar{r}_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	unsat.	$-\infty$	$-\infty$	$-\infty$	a_9	$f_{\max}$
$\mathbb{R}_-$	unsat.	a_7	x_u	x_u	a_7	$f_{\max}$
-0	unsat.	a_8	a_8	a_8	a_8	$f_{\max}$
$+0$	unsat.	z_u	$+0$	$+0$	z_u	$f_{\max}$
$\mathbb{R}_+$	unsat.	a_7	x_u	x_u	a_7	$f_{\max}$
$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$e_u \equiv x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 + z_u$;

$$a_7 = \begin{cases} +0, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u = -z_u; \\ x_u \boxplus_\downarrow z_u, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u \neq -z_u; \\ [\![e_u]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\downarrow = [e_u]_\downarrow; \\ [\![e_u]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\downarrow < [e_u]_\downarrow; \\ \mathrm{pred}\left([\![e_u]\!]_\uparrow\right), & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise}; \\ \mathrm{pred}\left(\mathrm{succ}(x_u) \boxplus_\uparrow z_u\right), & \text{if } \bar{r}_u = \downarrow; \\ +0, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u = -z_u; \\ x_u \boxplus_\downarrow z_u, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u \neq -z_u; \end{cases}$$

$$a_8 = \begin{cases} z_u, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}(z_u), & \text{otherwise}; \end{cases}$$

$$a_9 = \begin{cases} -\infty, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}(z_u \boxplus_\uparrow -f_{\max}), & \text{if } \bar{r}_u = \downarrow; \\ -f_{\max} \boxplus_\downarrow \left(\nabla_2^{\mathrm{n}-}(-f_{\max})/2 \boxplus_\downarrow z_u\right), & \text{otherwise}. \end{cases}$$

Fig. 14. First inverse projection of subtraction: function is_u^f.

$\mathrm{is}_l^s(y_l, x_u, \bar{r}_l)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$-f_{\max}$	$-f_{\max}$	$-f_{\max}$	$-f_{\max}$	$-f_{\max}$	$-\infty$
$\mathbb{R}_-$	a_{13}	a_{10}	a_{11}	y_l	a_{10}	$-\infty$
-0	$+\infty$	$-x_u$	a_{12}	-0	$-x_u$	$-\infty$
$+0$	$+\infty$	$-x_u$	a_{11}	-0	$-x_u$	$-\infty$
$\mathbb{R}_+$	$+\infty$	a_{10}	a_{11}	y_l	a_{10}	$-\infty$
$+\infty$	unsat.	unsat.	unsat.	unsat.	unsat.	$-\infty$

$$e_l \equiv y_l - \left(x_u + \nabla_2^{\mathrm{n}+}(x_u)/2\right);$$

$$a_{10} = \begin{cases} -0, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u = y_l; \\ y_l \boxminus_\uparrow x_u, & \text{if } \bar{r}_l = \mathrm{n},\ \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u \neq y_l; \\ [\![e_l]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_l]\!]_\uparrow = [e_l]_\uparrow; \\ [\![e_l]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_l]\!]_\uparrow > [e_l]_\uparrow; \\ \mathrm{succ}\left([\![e_l]\!]_\downarrow\right), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ -0, & \text{if } \bar{r}_l = \uparrow \text{ and } x_u = y_l; \\ y_l \boxminus_\uparrow x_u, & \text{if } \bar{r}_l = \uparrow \text{ and } x_u \neq y_l; \\ \mathrm{succ}\left(y_l \boxminus_\downarrow \mathrm{succ}(x_u)\right), & \text{if } \bar{r}_l = \downarrow; \end{cases}$$

$$(a_{11}, a_{12}) = \begin{cases} (y_l, -0), & \text{if } \bar{r}_l = \downarrow; \\ (\mathrm{succ}(y_l), +0), & \text{otherwise}; \end{cases}$$

$$a_{13} = \begin{cases} +\infty, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}(y_l \boxplus_\downarrow f_{\max}), & \text{if } \bar{r}_l = \downarrow; \\ f_{\max} \boxplus_\uparrow \left(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxplus_\uparrow y_l\right), & \text{otherwise.} \end{cases}$$

Fig. 15. Second inverse projection of subtraction: function is_l^s.

$\mathrm{is}_u^s(y_u, x_l, \bar{r}_u)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	unsat.	unsat.	unsat.	unsat.	unsat.
$\mathbb{R}_-$	$+\infty$	a_{14}	y_u	a_{15}	a_{14}	$-\infty$
-0	$+\infty$	$-x_l$	$+0$	a_{15}	$-x_l$	$-\infty$
$+0$	$+\infty$	$-x_l$	$+0$	a_{16}	$-x_l$	$-\infty$
$\mathbb{R}_+$	$+\infty$	a_{14}	y_u	a_{15}	a_{14}	a_{17}
$+\infty$	$+\infty$	$f_{\max}$	$f_{\max}$	$f_{\max}$	$f_{\max}$	$f_{\max}$

$$e_u \equiv y_u - \left(x_l + \nabla_2^{\mathrm{n}-}(x_u)/2\right);$$

$$a_{14} = \begin{cases} +0, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l = y_u; \\ y_u \boxminus_\downarrow x_l, & \text{if } \bar{r}_u = \mathrm{n},\ \nabla_2^{\mathrm{n}-}(x_l) = -f_{\min} \text{ and } x_l \neq y_u; \\ [\![e_u]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq -f_{\min} \text{ and } [\![e_u]\!]_\downarrow = [e_u]_\downarrow; \\ [\![e_u]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_u),\ \nabla_2^{\mathrm{n}-}(x_u) \neq -f_{\min} \text{ and } [\![e_u]\!]_\downarrow < [e_u]_\downarrow; \\ \mathrm{pred}\left([\![e_u]\!]_\uparrow\right), & \text{if } \bar{r}_u = \mathrm{n}, \text{otherwise}; \\ \mathrm{pred}\left(y_u \boxminus_\uparrow \mathrm{pred}(x_l)\right), & \text{if } \bar{r}_u = \uparrow; \\ +0, & \text{if } \bar{r}_u = \downarrow \text{ and } x_l = y_u; \\ y_u \boxminus_\downarrow x_l, & \text{if } \bar{r}_u = \downarrow \text{ and } x_l \neq y_u; \end{cases}$$

$$(a_{15}, a_{16}) = \begin{cases} (\mathrm{pred}(y_u), -0), & \text{if } \bar{r}_u = \downarrow; \\ (y_u, +0), & \text{otherwise}; \end{cases}$$

$$a_{17} = \begin{cases} -\infty, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}(y_u \boxminus_\uparrow f_{\max}), & \text{if } \bar{r}_u = \uparrow; \\ -f_{\max} \boxplus_\downarrow \left(\nabla_2^{\mathrm{n}-}(-f_{\max})/2 \boxplus_\downarrow y_u\right), & \text{otherwise.} \end{cases}$$

Fig. 16. Second inverse projection of subtraction: function is_u^s.

Firstly, whenever $\mathrm{sgn}(y_l) \neq \mathrm{sgn}(y_u)$ and $\mathrm{sgn}(z_l) \neq \mathrm{sgn}(z_u)$, there is no unique choice for y_L and z_L (resp., y_U and z_U); therefore we need to compute the two candidate lower (and upper) bounds for x and then choose the minumum (the maximum, resp).

The choice is instead unique in all cases where the signs of one among y and z, or both of them, are constant over the respective intervals. Function σ of Figure 7 determines the extrema of y and z useful to compute the new lower (resp., upper) bound for y when the sign of z is constant. When the sign of y is constant, the appropriate choice for the extrema of y and z can be determined by swapping the role of y and z in function σ.

Once the extrema (y_L, y_U, z_L, z_U) have been selected, functions dm_l and dm_u of Figure 17 are used to find new bounds for x. It is worth noting that it is not necessary to compute new values of r_l and r_u for the application of functions dm_l and dm_u at line 6 of Algorithm 9. This is true because, by Definition 7, the choice of r_l (of r_u, resp.) is driven by the sign of $y_L \boxdot z_L$ (of $y_U \boxdot z_U$, resp.) only. Since, in this case, the sign of $y_L \boxdot z_L$ (of $y_U \boxdot z_U$, resp.) as defined at line 2 and the sign of $y_L \boxdot z_L$ (of $y_U \boxdot z_U$, resp.) as defined at line 5 are the same, we do not need to compute r_l and r_u another time.

Algorithm 9 Direct projection for multiplication constraints.

Require: $x = y \boxdot_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $X' \subseteq X$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxdot_r z \implies x \in X'$ and $\forall X'' \subset X : \exists r \in S, y \in Y, z \in Z \,.\, y \boxdot_r z \notin X''$.

1: **if** $\mathrm{sgn}(y_l) \neq \mathrm{sgn}(y_u)$ **and** $\mathrm{sgn}(z_l) \neq \mathrm{sgn}(z_u)$ **then**
2: $(y_L, y_U, z_L, z_U) := (y_l, y_l, z_u, z_l)$;
3: $r_l := r_l(S, y_L, \boxdot, z_L)$; $r_u := r_u(S, y_U, \boxdot, z_U)$;
4: $v_l := \mathrm{dm}_l(y_L, z_L, r_l)$; $v_u := \mathrm{dm}_u(y_U, z_U, r_u)$;
5: $(y_L, y_U, z_L, z_U) := (y_u, y_u, z_l, z_u)$;
6: $w_l := \mathrm{dm}_l(y_L, z_L, r_l)$; $w_u := \mathrm{dm}_u(y_U, z_U, r_u)$;
7: $x'_l := \min\{v_l, w_l\}$; $x'_u := \max\{v_u, w_u\}$;
8: **else**
9: **if** $\mathrm{sgn}(y_l) = \mathrm{sgn}(y_u)$ **then**
10: $(y_L, y_U, z_L, z_U) := \sigma(y_l, y_u, z_l, z_u)$;
11: **else**
12: $(z_L, z_U, y_L, y_U) := \sigma(z_l, z_u, y_l, y_u)$;
13: **end if**
14: $r_l := r_l(S, y_L, \boxdot, z_L)$; $r_u := r_u(S, y_U, \boxdot, z_U)$;
15: $x'_l := \mathrm{dm}_l(y_L, z_L, r_l)$; $x'_u := \mathrm{dm}_u(y_U, z_U, r_u)$;
16: **end if**
17: $X' := X \cap [x'_l, x'_u]$;

Theorem 9. Algorithm 9 *satisfies its contract.*

Inverse Propagation. For inverse propagation, Algorithm 10 partitions interval Z into the sign-homogeneous intervals $Z_- \stackrel{\text{def}}{=} Z \cap [-\infty, -0]$ and $Z_+ \stackrel{\text{def}}{=}$

$Z \cap [+0, +\infty]$. This is done because the sign of Z must be taken into account in order to derive correct bounds for Y. Hence, once Z has been partitioned into sign-homogeneous intervals, we use intervals X and Z_- to obtain interval $[y_l^-, y_u^-]$, and X and Z_+ to obtain $[y_l^+, y_u^+]$. To do so, the algorithm determines the appropriate extrema of intervals X and $W = Z_-$ or $W = Z_+$ to be used for constraint propagation. To this aim, function τ of Figure 5 is employed; note that the sign of W is, by construction, constant over the interval. The chosen extrema are then passed as parameters to functions im_l of Figure 18 and im_u of Figure 19, that compute the new, refined bounds for y, by using the inverse operation of multiplication, i.e., division. The so obtained intervals $Y \cap [y_l^-, y_u^-]$ and $Y \cap [y_l^+, y_u^+]$ will be then joined with convex union, denoted by $\biguplus$, to obtain Y'.

Algorithm 10 Inverse projection for multiplication constraints.

Require: $x = y \boxdot_S z$, $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$.
Ensure: $Y' \subseteq Y$ and $\forall r \in S, x \in X, y \in Y, z \in Z : x = y \boxdot_r z \implies y \in Y'$.

$Z_- := Z \cap [-\infty, -0]$;
if $Z_- \neq \varnothing$ **then**
  $W := Z_-$;
  $(x_L, x_U, w_L, w_U) := \tau(x_l, x_u, w_l, w_u)$;
  $\bar{r}_l := \bar{r}_l^{\mathrm{I}}(S, x_L, \boxdot, w_L)$; $\bar{r}_u := \bar{r}_u^{\mathrm{I}}(S, x_U, \boxdot, w_U)$;
  $y_l^- := \mathrm{im}_l(x_L, w_L, \bar{r}_l)$; $y_u^- := \mathrm{im}_u(x_U, w_U, \bar{r}_u)$;
  if $y_l^- \in \mathbb{F}$ and $y_u^- \in \mathbb{F}$ **then**
    $Y'_- = Y \cap [y_l^-, y_u^-]$;
  else
    $Y'_- = \varnothing$;
  end if
else
  $Y'_- = \varnothing$;
end if
$Z_+ := Z \cap [+0, +\infty]$;
if $Z_+ \neq \varnothing$ **then**
  $W := Z_+$;
  $(x_L, x_U, w_L, w_U) := \tau(x_l, x_u, w_l, w_u)$;
  $\bar{r}_l := \bar{r}_l^{\mathrm{I}}(S, x_L, \boxdot, w_L)$; $\bar{r}_u := \bar{r}_u^{\mathrm{I}}(S, x_U, \boxdot, w_U)$;
  $y_l^+ := \mathrm{im}_l(x_L, w_L, \bar{r}_l)$; $y_u^+ := \mathrm{im}_u(x_U, w_U, \bar{r}_u)$;
  if $y_l^+ \in \mathbb{F}$ and $y_u^+ \in \mathbb{F}$ **then**
    $Y'_+ = Y \cap [y_l^+, y_u^+]$;
  else
    $Y'_+ = \varnothing$;
  end if
else
  $Y'_+ = \varnothing$;
end if
$Y' := Y'_- \biguplus Y'_+$;

Theorem 10. Algorithm 10 *satisfies its contract.*

Of course, the refinement Z' of Z can be defined analogously.

$\mathrm{dm}_l(y_L, z_L)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$+\infty$	-0	$-\infty$	$-\infty$
$\mathbb{R}_-$	$+\infty$	$y_L \boxdot_{r_l} z_L$	$+0$	-0	$y_L \boxdot_{r_l} z_L$	$-\infty$
-0	$+\infty$	$+0$	$+0$	-0	-0	-0
$+0$	-0	-0	-0	$+0$	$+0$	$+\infty$
$\mathbb{R}_+$	$-\infty$	$y_L \boxdot_{r_l} z_L$	-0	$+0$	$y_L \boxdot_{r_l} z_L$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	-0	$+\infty$	$+\infty$	$+\infty$

$\mathrm{dm}_u(y_U, z_U)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$+0$	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$+\infty$	$y_U \boxdot_{r_u} z_U$	$+0$	-0	$y_U \boxdot_{r_u} z_U$	$-\infty$
-0	$+0$	$+0$	$+0$	-0	-0	$-\infty$
$+0$	$-\infty$	-0	-0	$+0$	$+0$	$+0$
$\mathbb{R}_+$	$-\infty$	$y_U \boxdot_{r_u} z_U$	-0	$+0$	$y_U \boxdot_{r_u} z_U$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	$-\infty$	$+0$	$+\infty$	$+\infty$

Fig. 17. Direct projection of multiplication: functions dm_l and dm_u.

$\mathrm{im}_l(x_L, w_L)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$f_{\min}$	a_4	unsat.	$-\infty$	$-\infty$	$-\infty$
$\mathbb{R}_-$	$f_{\min}$	a_3^-	unsat.	$-f_{\max}$	a_3^+	$f_{\min}$
-0	$+0$	$+0$	$+0$	$-f_{\max}$	a_5	$f_{\min}$
$+0$	$f_{\min}$	a_6	$-f_{\max}$	$+0$	$+0$	$+0$
$\mathbb{R}_+$	$f_{\min}$	a_3^-	$-f_{\max}$	unsat.	a_3^+	$f_{\min}$
$+\infty$	$-\infty$	$-\infty$	$-\infty$	unsat.	a_7	$f_{\min}$

$$e_l^+ \equiv (x_L + \nabla_2^{\mathrm{n}-}(x_L)/2)/w_L;$$

$$a_3^+ = \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow > [e_l^+]_\uparrow; \\ \mathrm{succ}\big([\![e_l^+]\!]_\downarrow\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ x_L \boxslash_\uparrow w_L, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}\big(\mathrm{pred}(x_L) \boxslash_\downarrow w_L\big), & \text{if } \bar{r}_l = \uparrow; \end{cases}$$

$$e_l^- \equiv (x_L + \nabla_2^{\mathrm{n}+}(x_L)/2)/w_L;$$

$$a_3^- = \begin{cases} [\![e_l^-]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^-]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n},\ \mathrm{even}(x_L) \text{ and } [\![e_l^-]\!]_\uparrow > [e_l^-]_\uparrow; \\ \mathrm{succ}\big([\![e_l^-]\!]_\downarrow\big), & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \\ x_L \boxslash_\uparrow w_L, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}\big(\mathrm{succ}(x_L) \boxslash_\downarrow w_L\big), & \text{if } \bar{r}_l = \downarrow; \end{cases}$$

$$e_l^1 \equiv (-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2)/w_L;$$

$$a_4 = \begin{cases} +\infty, & \text{if } \bar{r}_l = \uparrow; \\ \mathrm{succ}(-f_{\max} \boxslash_\downarrow w_L), & \text{if } \bar{r}_l = \downarrow; \\ [\![e_l^1]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [\![e_l^1]\!]_\uparrow = [e_l^1]_\uparrow; \\ [\![e_l^1]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}; \end{cases}$$

$$(a_5, a_6) = \begin{cases} (-0,\ \mathrm{succ}(f_{\min} \boxslash_\downarrow w_L)), & \text{if } \bar{r}_l = \downarrow; \\ (\mathrm{succ}(-f_{\max} \boxslash_\downarrow w_L),\ -0), & \text{if } \bar{r}_l = \uparrow; \\ (-f_{\min} \boxslash_\uparrow (2 \cdot w_L),\ f_{\min} \boxslash_\uparrow (2 \cdot w_L)), & \text{if } \bar{r}_l = \mathrm{n}; \end{cases}$$

$$e_l^2 \equiv (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)/w_L;$$

$$a_7 = \begin{cases} +\infty, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}(f_{\max} \boxslash_\downarrow w_L), & \text{if } \bar{r}_l = \uparrow; \\ [\![e_l^2]\!]_\uparrow, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } [\![e_l^2]\!]_\uparrow = [e_l^2]_\uparrow; \\ [\![e_l^2]\!]_\downarrow, & \text{if } \bar{r}_l = \mathrm{n}, \text{ otherwise}. \end{cases}$$

Fig. 18. Inverse projection of multiplication: function im_l.

$\mathrm{im}_u(x_U, w_U)$	$-\infty$	$\mathbb{R}_-$	-0	$+0$	$\mathbb{R}_+$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$+\infty$	unsat.	a_9	$-f_{\min}$
$\mathbb{R}_-$	$-f_{\min}$	a_8^-	$f_{\max}$	unsat.	a_8^+	$-f_{\min}$
-0	$-f_{\min}$	a_{10}	$f_{\max}$	-0	-0	-0
$+0$	-0	-0	-0	$f_{\max}$	a_{11}	$-f_{\min}$
$\mathbb{R}_+$	$-f_{\min}$	a_8^-	unsat.	$f_{\max}$	a_8^+	$-f_{\min}$
$+\infty$	$-f_{\min}$	a_{12}	unsat.	$+\infty$	$+\infty$	$+\infty$

$$e_u^+ \equiv (x_U + \nabla_2^{\mathrm{n}+}(x_U)/2)/w_U;$$

$$a_8^+ = \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\uparrow = [e_u^+]_\uparrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\uparrow > [e_u^+]_\uparrow; \\ \mathrm{pred}\big([\![e_u^+]\!]_\uparrow\big), & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ \mathrm{pred}\big(\mathrm{succ}(x_U) \boxslash_\uparrow w_U\big), & \text{if } \bar{r}_u = \downarrow; \\ x_U \boxslash_\downarrow w_U, & \text{if } \bar{r}_u = \uparrow; \end{cases}$$

$$e_u^- \equiv (x_U + \nabla_2^{\mathrm{n}-}(x_U)/2)/w_U;$$

$$a_8^- = \begin{cases} [\![e_u^-]\!]_\downarrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^-]\!]_\uparrow = [e_u^-]_\uparrow; \\ [\![e_u^-]\!]_\uparrow, & \text{if } \bar{r}_u = \mathrm{n},\ \mathrm{even}(x_U) \text{ and } [\![e_u^-]\!]_\uparrow > [e_u^-]_\uparrow; \\ \mathrm{pred}\big([\![e_u^-]\!]_\uparrow\big), & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \\ \mathrm{pred}\big(\mathrm{pred}(x_U) \boxslash_\uparrow w_U\big), & \text{if } \bar{r}_u = \uparrow; \\ x_U \boxslash_\downarrow w_U, & \text{if } \bar{r}_u = \downarrow; \end{cases}$$

$$e_u^1 \equiv (-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2)/w_U;$$

$$a_9 = \begin{cases} -\infty, & \text{if } \bar{r}_u = \uparrow; \\ \mathrm{pred}(-f_{\max} \boxslash_\uparrow w_U), & \text{if } \bar{r}_u = \downarrow; \\ [\![e_u^1]\!]_\downarrow & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [\![e_u^1]\!]_\downarrow = [e_u^1]_\downarrow; \\ [\![e_u^1]\!]_\uparrow & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise;} \end{cases}$$

$$(a_{10}, a_{11}) = \begin{cases} (+0,\ \mathrm{pred}(f_{\min} \boxslash_\uparrow w_U)), & \text{if } \bar{r}_u = \downarrow; \\ (\mathrm{pred}(-f_{\min} \boxslash_\uparrow w_U),\ +0), & \text{if } \bar{r}_u = \uparrow; \\ (-f_{\min} \boxslash_\downarrow (2 \cdot w_U),\ f_{\min} \boxslash_\downarrow (2 \cdot w_U)), & \text{if } \bar{r}_u = \mathrm{n}; \end{cases}$$

$$e_u^2 \equiv (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)/w_U;$$

$$a_{12} = \begin{cases} -\infty, & \text{if } \bar{r}_u = \downarrow; \\ \mathrm{pred}(f_{\max} \boxslash_\uparrow w_U), & \text{if } \bar{r}_u = \uparrow; \\ [\![e_u^2]\!]_\downarrow & \text{if } \bar{r}_u = \mathrm{n} \text{ and } [\![e_u^2]\!]_\downarrow = [e_u^2]_\downarrow; \\ [\![e_u^2]\!]_\uparrow & \text{if } \bar{r}_u = \mathrm{n}, \text{ otherwise.} \end{cases}$$

Fig. 19. Inverse projection of multiplication: function im_u.

B Proofs of Results

B.1 Proofs of Results in Section 2

Proposition 5. (Properties of rounding functions.) *Let $x \in \mathbb{R} \setminus \{0\}$. Then*

$$[x]_\downarrow \leq x \leq [x]_\uparrow, \tag{29}$$

$$[x]_\downarrow \leq [x]_0 \leq [x]_\uparrow, \tag{30}$$

$$[x]_\downarrow \leq [x]_{\mathrm{n}} \leq [x]_\uparrow. \tag{31}$$

Moreover,

$$[x]_\downarrow = -[-x]_\uparrow. \tag{32}$$

Proof. In order to prove (29), we first prove that $[x]_\downarrow \leq x$. To this aim, consider the following cases on $x \in \mathbb{R} \setminus \{0\}$:

$-f_{\max} \leq x < 0 \ \vee \ f_{\min} \leq x$: by (2) we have $[x]_\downarrow = \max\{\, z \in \mathbb{F} \mid z \leq x \,\}$, hence $[x]_\downarrow \leq x$;
$0 < x < f_{\min}$: by (2) we have $[x]_\downarrow = -0 \leq x$;
$x < -f_{\max}$: by (2) we have $[x]_\downarrow = -\infty \leq x$.

We now prove that $x \leq [x]_\uparrow$. Consider the following cases on $x \in \mathbb{R} \setminus \{0\}$:

$x > f_{\max}$: by (1) we have $[x]_\uparrow = +\infty$ and thus $x \leq [x]_\uparrow$ holds;
$x \leq -f_{\min} \ \vee \ 0 < x \leq f_{\max}$: by (1) we have $[x]_\uparrow = \min\{\, z \in \mathbb{F} \mid z \geq x \,\}$, hence $x \leq [x]_\uparrow$ holds;
$-f_{\min} < x < 0$: by (1) we have $[x]_\uparrow = -0$ hence $x \leq [x]_\uparrow$ holds.

In order to prove (30), consider the following cases on $x \in \mathbb{R} \setminus \{0\}$:

$x > 0$: by (3) we have $[x]_0 = [x]_\downarrow \leq [x]_\uparrow$;
$x < 0$: by (3) we have $[x]_\downarrow \leq [x]_\uparrow = [x]_0$.

In order to prove (31), consider the following cases on $x \in \mathbb{R} \setminus \{0\}$:

$-f_{\max} \leq x \leq f_{\max}$: we have the following cases
  $\big|[x]_\downarrow - x\big| < \big|[x]_\uparrow - x\big| \vee \big(\big|[x]_\downarrow - x\big| = \big|[x]_\uparrow - x\big|\big) \wedge \mathrm{even}\big([x]_\downarrow\big)$**:** by (4), we have $[x]_{\mathrm{n}} = [x]_\downarrow \leq [x]_\uparrow$;
  $\big|[x]_\downarrow - x\big| > \big|[x]_\uparrow - x\big| \vee \big(\big|[x]_\downarrow - x\big| = \big|[x]_\uparrow - x\big| \wedge \neg\, \mathrm{even}\big([x]_\downarrow\big)\big)$**:** by (4) we have $[x]_\downarrow \leq [x]_\uparrow = [x]_{\mathrm{n}}$.
$-f_{\max} > x$: we have the following cases
  $-2^{e_{\max}}(2 - 2^{-p}) < x < -f_{\max}$**:** by (4) we have $[x]_\downarrow \leq [x]_\uparrow = [x]_{\mathrm{n}}$.
  $x \leq -2^{e_{\max}}(2 - 2^{-p})$**:** by (4) we have $[x]_{\mathrm{n}} = [x]_\downarrow \leq [x]_\uparrow$;
$f_{\max} < x$: we have the following cases
  $2^{e_{\max}}(2 - 2^{-p}) > x > f_{\max}$**:** by (4) we have $[x]_{\mathrm{n}} = [x]_\downarrow \leq [x]_\uparrow$;
  $x \geq 2^{e_{\max}}(2 - 2^{-p})$**:** by (4) we have $[x]_\downarrow \leq [x]_\uparrow = [x]_{\mathrm{n}}$.

In order to prove (32), let us compute $-[-x]_\uparrow$. There are the following cases:

$-x > f_{\max}$: this implies that $x < -f_{\max}$ and, by (1), $[-x]_\uparrow = +\infty$; hence, by (2), $-[-x]_\uparrow = -\infty = [x]_\downarrow$;
$-x \leq -f_{\min} \ \vee \ 0 < -x \leq f_{\max}$: this implies that $x \geq f_{\min} \ \vee \ -f_{\max} \geq x > 0$ and, by (1), we have $[-x]_\uparrow = \min\{\, z \in \mathbb{F} \mid z \geq -x \,\}$; therefore, by (2), $-[-x]_\uparrow = -\min\{\, z \in \mathbb{F} \mid z \geq -x \,\} = \max\{\, z \in \mathbb{F} \mid z \leq x \,\} = [x]_\downarrow$,
$-f_{\min} < -x < 0$: this implies that $0 < x < f_{\min}$ and, by (1), $[-x]_\uparrow = -0$; hence, by (2), $-[-x]_\uparrow = +0 = [x]_\downarrow$. □

B.2 Proofs of Results in Section 3

Proof (of Proposition 1). First observe that, for each $x, y, z \in \mathbb{F}$, we have $[y \circ z]_{\mathrm{n}} = [y \circ z]_{\downarrow}$, or $[y \circ z]_{\mathrm{n}} = [y \circ z]_{\uparrow}$ or both. Then, in order to prove the claim we will first prove that, for each $x, y, z \in \mathbb{F}$, $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$. We have the following cases, depending on $y \circ z$:

$y \circ z = +\infty \ \vee \ y \circ z = -\infty$: in this case we have $y \boxdot_{\downarrow} z = y \boxdot_{\mathrm{n}} z = y \boxdot_{\uparrow} z$ and thus $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$ holds.

$y \circ z \leq -f_{\min} \ \vee \ y \circ z \geq f_{\min}$: in this case we have, by Proposition 5, $y \boxdot_{\downarrow} z = [y \circ z]_{\downarrow} \leq [y \circ z]_{\mathrm{n}} = y \boxdot_{\mathrm{n}} z \leq [y \circ z]_{\uparrow} = y \boxdot_{\uparrow} z$; as $y \boxdot_{\downarrow} z \neq 0$, $y \boxdot_{\mathrm{n}} z \neq 0$ and $y \boxdot_{\uparrow} z \neq 0$, the numerical order is reflected into the symbolic order to give $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$.

$-f_{\min} < y \circ z < 0$: in this case we have $y \boxdot_{\downarrow} z = -f_{\min} \leq y \boxdot_{\mathrm{n}} z \leq y \boxdot_{\uparrow} z = -0$ by Definition 5; since either $[y \circ z]_{\mathrm{n}} = -f_{\min}$ or $[y \circ z]_{\mathrm{n}} = -0$, we have $[y \circ z]_{\mathrm{n}} \neq +0$, thus $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$.

$0 < y \circ z < f_{\min}$: in this case we have $y \boxdot_{\downarrow} z = +0 \leq y \boxdot_{\mathrm{n}} z \leq y \boxdot_{\uparrow} z = f_{\min}$ by Definition 5; again, since either $[y \circ z]_{\mathrm{n}} = +0$ or $[y \circ z]_{\mathrm{n}} = f_{\min}$ we know that $[y \circ z]_{\mathrm{n}} \neq -0$, and thus $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$.

$y \circ z = 0$: in this case, for multiplication and division the result is the same for all rounding modes, i.e., $+0$ or -0 depending on the sign of the arguments [IEE08, Section 6.3]; for addition or subtraction we have $y \boxdot_{\downarrow} z \neq -0$ while $y \boxdot_{\mathrm{n}} z = y \boxdot_{\uparrow} z = +0$; hence, also in this case, $y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z$ holds.

Note now that, by Definition 5, if $y \circ z > 0$ then $y \boxdot_0 z = y \boxdot_{\downarrow} z$ whereas, if $y \circ z > 0$, then $y \boxdot_0 z = y \boxdot_{\uparrow} z$. Therefore we can conclude that, if $y \circ z > 0$,

$$y \boxdot_{\downarrow} z = y \boxdot_0 z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_{\uparrow} z,$$

while, if $y \circ z < 0$,

$$y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z \preccurlyeq y \boxdot_0 z = y \boxdot_{\uparrow} z;$$

moreover, if $y \circ z = 0$ and $\circ \notin \{+, -\}$,

$$y \boxdot_{\downarrow} z = y \boxdot_{\mathrm{n}} z = y \boxdot_0 z = y \boxdot_{\uparrow} z,$$

while, if $y \circ z = 0$ and $\circ \in \{+, -\}$,

$$y \boxdot_{\downarrow} z \preccurlyeq y \boxdot_{\mathrm{n}} z = y \boxdot_0 z = y \boxdot_{\uparrow} z.$$

In order to prove the first part of the claim it is now sufficient to consider all possible sets $S \subseteq R$ and use the relations above. For the second part of the claim, observe that $r_l(S, y, \boxdot, z) \in S$ except for one case: that is when $y \circ z > 0$, $0 \in S$ but $\downarrow \notin S$. In this case, however, by Definition 5, $y \boxdot_{\downarrow} z = y \boxdot_{\mathrm{n}} z$. Similarly, $r_u(S, y, \boxdot, z) \in S$ except for the case when $y \circ z \leq 0$, $0 \in S$ but $\uparrow \notin S$. Assume first that $y \circ z < 0$, in this case, by Definition 5, $y \boxdot_{\uparrow} z = y \boxdot_{\mathrm{n}} z$. For the remaining case, that is $y \circ z = 0$, remember that for multiplication and division the result

is the same for all rounding modes [IEE08, Section 6.3], while for addition or subtraction we have that $y \boxdot_0 z = y \boxdot_\mathrm{n} z = y \boxdot_\uparrow z = +0$.

We will now prove the second part of Proposition 1, regarding rounding mode selectors for inverse propagators. Before doing so, we need to prove the following result. Let $\boxdot \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$, and let r and s be two IEEE 754 rounding modes, such that for any $a, b \in \mathbb{F}$,

$$a \boxdot_r b \preccurlyeq a \boxdot_s b.$$

Moreover, let $x, z \in \mathbb{F}$, and let $\bar{y}_\mathrm{s}$ be the minimum $y_\mathrm{s} \in \mathbb{F}$ such that $x = y_\mathrm{s} \boxdot_\mathrm{s} z$. Then, for any $y_\mathrm{r} \in \mathbb{F}$ such that $x = y_\mathrm{r} \boxdot_\mathrm{r} z$ we have

$$\begin{aligned} \bar{y}_\mathrm{s} \boxdot_\mathrm{r} z &\preccurlyeq \bar{y}_\mathrm{s} \boxdot_\mathrm{s} z \\ &= x \\ &= y_\mathrm{r} \boxdot_\mathrm{r} z. \end{aligned}$$

This leads us to write

$$[\bar{y}_\mathrm{s} \circ z]_\mathrm{r} \preccurlyeq [y_\mathrm{r} \circ z]_\mathrm{r}$$

which, due to the isotonicity of all IEEE 754 rounding modes, implies

$$\bar{y}_\mathrm{s} \circ z \preccurlyeq y_\mathrm{r} \circ z.$$

Finally, if operator '$\circ$' is isotone we have

$$\bar{y}_\mathrm{s} \preccurlyeq y_\mathrm{r},$$

which implies that $\bar{y}_\mathrm{s}$ is the minimum $y \in \mathbb{F}$ such that $x = y \boxdot_r z$ or $x = y \boxdot_s z$. On the other hand, if '$\circ$' is antitone we have

$$\bar{y}_\mathrm{s} \succcurlyeq y_\mathrm{r},$$

and $\bar{y}_\mathrm{s}$ is the maximum $y \in \mathbb{F}$ such that $x = y \boxdot_r z$ or $x = y \boxdot_s z$. An analogous result can be proved regarding the upper bound for y in case the operator is isotone, and regarding the lower bound for y in case it is antitone.

The above claim allows us to prove the following. Assume first that $\boxdot$ is isotone with respect to y in $x = y \boxdot z$. Let $\hat{y}_\uparrow$ be the minimum $y_\uparrow \in \mathbb{F}$ such that $x = y_\uparrow \boxdot_\uparrow z = [y_\uparrow \boxdot z]_\uparrow$, let $\hat{y}_\mathrm{n}$ be the minimum $y_\mathrm{n} \in \mathbb{F}$ such that $x = y_\mathrm{n} \boxdot_\mathrm{n} z = [y_\mathrm{n} \boxdot z]_\mathrm{n}$ and, finally, let $\hat{y}_\downarrow$ be the minimum $y_\downarrow \in \mathbb{F}$ such that $x = y_\downarrow \boxdot_\downarrow z = [y_\downarrow \boxdot z]_\downarrow$. We will prove that

$$\hat{y}_\uparrow \preccurlyeq \hat{y}_\mathrm{n} \preccurlyeq \hat{y}_\downarrow.$$

Since we assumed that $\boxdot$ is isotone with respect to y in $x = y \boxdot z$, the rounding mode that gives the minimal y solution of $x = [y \boxdot z]_\mathrm{r}$ is the one that yields a bigger (w.r.t. $\preccurlyeq$ order) floating point number, as we proved before. We must now separately treat the following cases:

$y \boxdot z \neq 0$: By 31, we have $[y \boxdot z]_\downarrow \leq [y \boxdot z]_\mathrm{n} \leq [y \boxdot z]_\uparrow$. Since in this case $y \boxdot z \neq 0$, we have that $[y \boxdot z]_\downarrow \preccurlyeq [y \boxdot z]_\mathrm{n} \preccurlyeq [y \boxdot z]_\uparrow$. This implies $\hat{y}_\uparrow \preccurlyeq \hat{y}_\mathrm{n} \preccurlyeq \hat{y}_\downarrow$.

$y \boxdot z = 0$: In this case, $[y \boxdot z]_\downarrow \preccurlyeq [y \boxdot z]_{\mathrm{n}} = [y \boxdot z]_\uparrow$. This implies $\hat{y}_\uparrow \preccurlyeq \hat{y}_{\mathrm{n}} \preccurlyeq \hat{y}_\downarrow$.

Moreover, let $\tilde{y}_\uparrow$ be the maximum $y_\uparrow \in \mathbb{F}$ such that $x = y_\uparrow \boxdot_\uparrow z = [y_\uparrow \boxdot z]_\uparrow$, let $\tilde{y}_{\mathrm{n}}$ be the maximum $y_{\mathrm{n}} \in \mathbb{F}$ such that $x = y_{\mathrm{n}} \boxdot_{\mathrm{n}} z = [y_{\mathrm{n}} \boxdot z]_{\mathrm{n}}$ and, finally, let $\tilde{y}_\downarrow$ be the maximum $y_\downarrow \in \mathbb{F}$ such that $x = y_\downarrow \boxdot_\downarrow z = [y_\downarrow \boxdot z]_\downarrow$. We will prove the fact that

$$\tilde{y}_\uparrow \preccurlyeq \tilde{y}_{\mathrm{n}} \preccurlyeq \tilde{y}_\downarrow.$$

Since we assumed that $\boxdot$ is isotone with respect to y in $x = y \boxdot z$, the rounding mode that gives a maximum y solution of $x = [y \boxdot z_{\mathrm{r}}$ is the one that gives a smaller (w.r.t. $\preccurlyeq$ order) floating point number. We must now deal with the following cases:

$y \boxdot z \neq 0$: By 31, we have $[y \boxdot z]_\downarrow \leq [y \boxdot z]_{\mathrm{n}} \leq [y \boxdot z]_\uparrow$. Since in this case $y \boxdot z \neq 0$, we have $[y \boxdot z]_\downarrow \preccurlyeq [y \boxdot z]_{\mathrm{n}} \preccurlyeq [y \boxdot z]_\uparrow$. This implies $\tilde{y}_\uparrow \preccurlyeq \tilde{y}_{\mathrm{n}} \preccurlyeq \tilde{y}_\downarrow$.
$y \boxdot z = 0$: In this case $[y \boxdot z]_\downarrow \preccurlyeq [y \boxdot z]_{\mathrm{n}} = [y \boxdot z]_\uparrow$. This implies $\tilde{y}_\uparrow \preccurlyeq \tilde{y}_{\mathrm{n}} \preccurlyeq \tilde{y}_\downarrow$.

The inequalities $\hat{y}_\uparrow \preccurlyeq \hat{y}_{\mathrm{n}} \preccurlyeq \hat{y}_\downarrow$ and $\tilde{y}_\uparrow \preccurlyeq \tilde{y}_{\mathrm{n}} \preccurlyeq \tilde{y}_\downarrow$ allow us to claim that the rounding mode selectors $\hat{r}_l(S, \boxdot, b)$ and $\hat{r}_u(S, b)$ are correct when $\boxdot$ is isotone with respect to y. In a similar way it is possible to prove that, in case $\boxdot$ is antitone with respect to argument y, the above-mentioned rounding mode selectors can be exchanged: $\hat{r}_u(S, b)$ can be used to obtain the lower bound for y, while $\hat{r}_l(S, \boxdot, b)$ can be used to obtain the upper bound.

Note that, in general, the roundTowardZero rounding mode is equivalent to roundTowardPositive if the result of the rounded operation is negative, and to roundTowardNegative if it is positive. The only case in which this is not true is when the result is $+0$ and the operation is a sum or a subtraction: this value can come from the rounding toward negative infinity of a strictly positive exact result, or the sum of $+0$ and -0, which behaves like roundTowardPositive, yielding $+0$. This case must be treated separately, and it is significant only in $\hat{r}_l(S, \boxdot, b)$, which is used when seeking for the lowest possible value of the variable to be refined that yields $+0$.

Definition 8 also contains selectors that can choose between rounding mode selectors $\hat{r}_l(S, b)$ and $\hat{r}_u(S, b)$ by distinguishing whether the operator is isotone or antitone with respect to the operand y to be derived by propagation; they take the result of the operation b and the known operand a into account. In particular, $\vec{r}_l^{\,\mathrm{l}}(S, b, \boxdot, a)$, $\vec{r}_u^{\,\mathrm{l}}(S, b, \boxdot, a)$ choose the appropriate selector for the leftmost operand, and $\vec{r}_l^{\,\mathrm{r}}(S, b, \boxdot, a)$, $\vec{r}_u^{\,\mathrm{r}}(S, b, \boxdot, a)$ are valid for the rightmost one. □

Proposition 6. *For each $r \in \mathbb{R} \setminus \{0\}$ we have*

$$0 \leq r - [r]_\downarrow < \nabla^\downarrow\big([r]_\downarrow\big) \tag{33}$$

$$\nabla^\uparrow\big([r]_\downarrow\big) < r - [r]_\uparrow \leq 0 \tag{34}$$

$$\nabla_2^{\mathrm{n}-}\big([r]_{\mathrm{n}}\big)/2 \leq r - [r]_{\mathrm{n}} \leq \nabla_2^{\mathrm{n}+}\big([r]_{\mathrm{n}}\big)/2, \tag{35}$$

where the two inequalities of (35) *are strict if $[r]_{\mathrm{n}}$ is odd.*

Proof (of Proposition 6). Suppose $r \in \mathbb{R}$ was rounded down to $x \in \mathbb{F}$. Then the error that was committed, $r - x$, is a nonnegative extended real that is strictly bounded from above by $\nabla^{\downarrow}(x) = \mathrm{succ}(x) - x$, that is, $0 \leq r - x < \mathrm{succ}(x) - x$, for otherwise we would have $r \geq \mathrm{succ}(x)$ or $r < x$ and, in both cases r would not have been rounded down to x. Note that $\nabla^{\downarrow}(f_{\max}) = +\infty$, coherently with the fact that the error is unbounded from above in this case.

Dually, if $r \in \mathbb{R}$ was rounded up to $x \in \mathbb{F}$ the error that was committed, $r - x$, is a nonpositive extended real that is strictly bounded from below by $\nabla^{\uparrow}(x) = \mathrm{pred}(x) - x$, that is, $\mathrm{pred}(x) - x < r - x \leq 0$ since, clearly, $\mathrm{pred}(x) < r \leq x$. Note that $\nabla^{\uparrow}(-f_{\max}) = -\infty$, coherently with the fact that the error is unbounded from below in this case.

Suppose now that $r \in \mathbb{R}$ was rounded-to-nearest to $x \in \mathbb{F}$. Then the error that was committed, $r - x$, is such that $\nabla_2^{\mathrm{n}-}(x)/2 \leq r - x \leq \nabla_2^{\mathrm{n}+}(x)/2$, where the two inequalities are strict if x is odd.

In fact, if $x \notin \{-\infty, -f_{\max}\}$, then $\nabla_2^{\mathrm{n}-}(x)/2 = \big(\mathrm{pred}(x) - x\big)/2 \leq r - x$, for otherwise r would be closer to $\mathrm{pred}(x)$. If $x = -\infty$, then $\nabla_2^{\mathrm{n}-}(x)/2 = +\infty$ and $r - x = +\infty$, so $\nabla_2^{\mathrm{n}-}(x)/2 \leq r - x$ holds. If $x = -f_{\max}$, then

$$\begin{aligned}
\nabla_2^{\mathrm{n}-}(x)/2 &= \big(-f_{\max} - \mathrm{succ}(-f_{\max})\big)/2 \\
&= \big(-2^{e_{\max}}(2 - 2^{1-p}) + 2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p})\big)/2 \\
&= \big(-2^{e_{\max}}(2 - 2^{1-p} - 2 + 2^{1-p} + 2^{1-p})\big)/2 \\
&= -2^{e_{\max}} 2^{1-p}/2 \\
&= -2^{e_{\max}+1-p}/2 \\
&= -2^{e_{\max}-p}
\end{aligned}$$

and thus, considering that $-f_{\max}$ is odd, $\nabla_2^{\mathrm{n}-}(x)/2 < r - x$ is equivalent to

$$\begin{aligned}
\nabla_2^{\mathrm{n}-}(x)/2 + x &= -\big(2^{e_{\max}-p} + 2^{e_{\max}}(2 - 2^{1-p})\big) \\
&= -2^{e_{\max}}(2^{-p} + 2 - 2^{1-p}) \\
&= -2^{e_{\max}}(2 - 2^{-p}) \\
&< r,
\end{aligned}$$

which must hold, for otherwise r would have been rounded to $-\infty$ [IEE08, Section 4.3.1].

Suppose now $x \notin \{+\infty, f_{\max}\}$: then $\nabla_2^{\mathrm{n}+}(x)/2 = \big(\mathrm{succ}(x) - x\big)/2 \geq r - x$, for otherwise r would be closer to $\mathrm{succ}(x)$. If $x = +\infty$, then $\nabla_2^{\mathrm{n}+}(x)/2 = -\infty$

and $r - x = -\infty$, and thus $\nabla_2^{n+}(x)/2 \geq r - x$ holds. If $x = f_{\max}$, then

$$\begin{aligned}
\nabla_2^{n+}(x)/2 &= \big(f_{\max} - \mathrm{pred}(f_{\max})\big)/2 \\
&= \big(2^{e_{\max}}(2 - 2^{1-p}) - 2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p})\big)/2 \\
&= \big(2^{e_{\max}}(2 - 2^{1-p} - 2 + 2^{1-p} + 2^{1-p})\big)/2 \\
&= 2^{e_{\max}} 2^{1-p}/2 \\
&= 2^{e_{\max}+1-p}/2 \\
&= 2^{e_{\max}-p}
\end{aligned}$$

and thus, considering that $f_{\max}$ is odd, $\nabla_2^{n+}(x)/2 > r - x$ is equivalent to

$$\begin{aligned}
\nabla_2^{n+}(x)/2 + x &= \big(2^{e_{\max}-p} + 2^{e_{\max}}(2 - 2^{1-p})\big) \\
&= 2^{e_{\max}}(2^{-p} + 2 - 2^{1-p}) \\
&= 2^{e_{\max}}(2 - 2^{-p}) \\
&> r,
\end{aligned}$$

which must hold, for otherwise r would have been rounded to $+\infty$. □

Proof (of Proposition 2). We first prove (9). By Definition 9, we have the following cases:

$x_l = -f_{\max}$**:** Then,

$$\begin{aligned}
x_l + \nabla_2^{n-}(x_l)/2 &= -f_{\max} + \big(-f_{\max} - \mathrm{succ}(-f_{\max})\big)/2 \\
&= -2^{e_{\max}}(2 - 2^{1-p}) + \big(-2^{e_{\max}}(2 - 2^{1-p}) \\
&\quad + 2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p})\big)/2 \\
&= -2^{e_{\max}}(2 - 2^{1-p} + 1 - 2^{-p} - 1 + 2^{1-p}) \\
&= -2^{e_{\max}}(2 - 2^{-p})
\end{aligned}$$

On the other hand, consider any x such that $x_l < x \leq x_u$. Since $x \in \mathbb{F}$, this implies that $\mathrm{succ}(-f_{\max}) \leq x \leq x_u$. In this case

$$x + \nabla_2^{n-}(x)/2 = (x + \mathrm{pred}(x))/2$$

Since 'pred' is monotone, the minimum can be found when $x = \mathrm{succ}(-f_{\max})$. In this case, we have that

$$\begin{aligned}
(x + \mathrm{pred}(x))/2 &= (\mathrm{succ}(-f_{\max}) - f_{\max})/2 \\
&= \big(-2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p}) - 2^{e_{\max}}(2 - 2^{1-p})\big)/2 \\
&= \big(-2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p} + 2 - 2^{1-p})\big)/2 \\
&= -2^{e_{\max}}(2 - 3 \cdot 2^{-p}) \\
&> x_l + \nabla_2^{n-}(x_l)/2 \\
&= -2^{e_{\max}}(2 - 2^{-p}).
\end{aligned}$$

Hence we can conclude that

$$\min_{x_l \le x \le x_u} \big(x + \nabla_2^{\mathrm{n}-}(x)/2\big) = x_l + \nabla_2^{\mathrm{n}-}(x_l)/2$$

$x_l > -f_{\max}$**:** In this case

$$x + \nabla_2^{\mathrm{n}-}(x)/2 = (x + \mathrm{pred}(x))/2$$

Since 'pred' is monotone,

$$\min_{x_l \le x \le x_u} \big(x + \nabla_2^{\mathrm{n}-}(x_l)/2\big) = x_l + \nabla_2^{\mathrm{n}-}(x_l)/2$$

We now prove (10). By Definition 9, we have the following cases:

$x_u = f_{\max}$**:** Then,

$$\begin{aligned} x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 &= f_{\max} + \big(f_{\max} - \mathrm{pred}(f_{\max})\big)/2 \\ &= 2^{e_{\max}}(2 - 2^{1-p}) + \big(2^{e_{\max}}(2 - 2^{1-p}) \\ &\quad - 2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p})\big)/2 \\ &= 2^{e_{\max}}(2 - 2^{1-p} + 1 - 2^{-p} - 1 + 2^{1-p}) \\ &= 2^{e_{\max}}(2 - 2^{-p}) \end{aligned}$$

Consider now any x such that $x_l \le x < x_u$. Since $x \in \mathbb{F}$, this implies that $x_l \le x \le \mathrm{pred}(f_{\max})$. In this case

$$x + \nabla_2^{\mathrm{n}+}(x)/2 = (x + \mathrm{succ}(x))/2$$

Since 'succ' is monotone, the maximum can be found when $x = \mathrm{pred}(f_{\max})$. In this case, we have that

$$\begin{aligned} (x + \mathrm{succ}(x))/2 &= (\mathrm{pred}(f_{\max}) + f_{\max})/2 \\ &= \big(2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p}) + 2^{e_{\max}}(2 - 2^{1-p})\big)/2 \\ &= \big(2^{e_{\max}}(2 - 2^{1-p} - 2^{1-p} + 2 - 2^{1-p})\big)/2 \\ &= 2^{e_{\max}}(2 - 3 \cdot 2^{-p}) \\ &> x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 \\ &= 2^{e_{\max}}(2 - 2^{-p}). \end{aligned}$$

Hence we can conclude that

$$\max_{x_l \le x \le x_u} \big(x + \nabla_2^{\mathrm{n}+}(x)/2\big) = x_u + \nabla_2^{\mathrm{n}+}(x_u)/2$$

$x_u < f_{\max}$**:** In this case

$$x + \nabla_2^{\mathrm{n}+}(x)/2 = (x + \mathrm{succ}(x))/2$$

Since 'succ' is monotone,

$$\max_{x_l \le x \le x_u} \left(x + \nabla_2^{\mathrm{n}+}(x)/2\right) = x_u + \nabla_2^{\mathrm{n}+}(x_u)/2$$

Proof (of Proposition 3). In order to prove (11), first observe that $x \preccurlyeq y \boxdot_\downarrow z$ implies that $x \le y \boxdot_\downarrow z$. Assume first that $y \boxdot_\downarrow z \in \mathbb{R}_+ \cup \mathbb{R}_-$. In this case, $y \boxdot_\downarrow z = [y \circ z]_\downarrow$. By Proposition 5, $y \boxdot_\downarrow z = [y \circ z]_\downarrow \le y \circ z$. Therefore, $x \le y \boxdot_\downarrow z = [y \circ z]_\downarrow \le y \circ z$. Then, assume that $y \boxdot_\downarrow z = +\infty$. In this case, since the rounding towards minus infinity never rounds to $+\infty$, it follows that $y \boxdot_\downarrow z = y \circ z$. Hence, $x \le y \circ z = +\infty$, holds. Assume now that $y \boxdot_\downarrow z = -\infty$. In this case it must be that $x = -\infty$ then $x \le y \circ z$, holds. Finally, assume that $y \boxdot_\downarrow z = +0$ or $y \boxdot_\downarrow z = -0$. In any case $x \preccurlyeq +0$ that implies $x \le 0$. On the other hand, we have two cases, $y \circ z \ne 0$ or $y \circ z = 0$. For the first case, by Definition 5, $0 \le y \circ z < f_{\min}$, then $x \le y \circ z$, holds. For the second case, since $x \le 0$ then $x \le y \circ z$.

In order to prove (12), as before observe that $x \preccurlyeq y \boxdot_\uparrow z$ implies that $x \le y \boxdot_\uparrow z$. Note that $x + \nabla^\uparrow(x) = \mathrm{pred}(x)$. So we are left to prove $\mathrm{pred}(x) < y \circ z$. Assume now that $0 < y \circ z \le f_{\max}$ or $x \le -f_{\min}$. Moreover, note that it cannot be the case that $\mathrm{pred}(x) \ge y \circ z$, otherwise, by Definition 5, $y \boxdot_\uparrow z \le \mathrm{pred}(x)$ and, therefore, $x \le y \boxdot_\uparrow z$ would not hold. Then, in this case, we can conclude $\mathrm{pred}(x) < y \circ z$. Now, assume that $-f_{\min} < y \circ z < 0$. In this case $y \boxdot_\uparrow z = -0$. Hence, $x \le 0$. By Definition 4, $\mathrm{pred}(x) \le -f_{\min}$. Hence, $\mathrm{pred}(x) \le y \circ z$, holds. Next, assume $y \circ z > f_{\max}$. In this case $y \boxdot_\uparrow z = \infty$. Hence, $x \le \infty$. By Definition 4, $\mathrm{pred}(x) \le f_{\max}$. Hence $\mathrm{pred}(x) < y \circ z$, holds. Next assume $y \circ z = 0$. In this case $y \boxdot_\uparrow z = +0$ or $y \boxdot_\uparrow z = -0$. Hence, $x \le 0$. By Definition 4, $\mathrm{pred}(x) \le -f_{\min}$. Hence $\mathrm{pred}(x) < y \circ z$, holds. Finally assume $y \circ z = \infty$. In this case $y \boxdot_\uparrow z = \infty$. Hence $x \preccurlyeq \infty$ and therefore $x \le \infty$. By Definition 4, $\mathrm{pred}(x) \le f_{\max}$. Hence $\mathrm{pred}(x) < y \circ z$, holds.

In order to prove (13), as the previous two cases, note that $x \preccurlyeq y \boxdot_{\mathrm{n}} z$ implies that $x \le y \boxdot_{\mathrm{n}} z$. First observe that for $x \ne -\infty$, $x + \nabla_2^{\mathrm{n}-}(x)/2 < x$. Indeed, assume first that $x \ne -f_{\max}$, then, by Definition 9, $\nabla_2^{\mathrm{n}-}(x) = x - \mathrm{succ}(x)$. Hence $x + \nabla_2^{\mathrm{n}-}(x)/2 = x + (x - \mathrm{succ}(x))/2 = (3x - \mathrm{succ}(x))/2$. Since $x < \mathrm{succ}(x)$, we can conclude that $x + \nabla_2^{\mathrm{n}-}(x)/2 < x$. Assume now that $x = -f_{\max}$. By Definition 9, $\nabla_2^{\mathrm{n}-}(x) = \mathrm{pred}(x) - x$. Hence $x + \nabla_2^{\mathrm{n}-}(x)/2 = x + (\mathrm{pred}(x) - x)/2 = (x + \mathrm{pred}(x))/2$. Since $x > \mathrm{pred}(x)$, we can conclude that $x + \nabla_2^{\mathrm{n}-}(x)/2 < x$.

Now, by Definition 5, we have to consider the following case for $x \boxdot_{\mathrm{n}} y \in \mathbb{R}_+ \cup \mathbb{R}_-$

$y \boxdot_{\mathrm{n}} z = [y \circ z]_\downarrow$. In this case, by Proposition 5, $x + \nabla_2^{\mathrm{n}-}(x)/2 < x \le y \boxdot_{\mathrm{n}} z = [y \circ z]_\downarrow \le y \circ z$. Therefore, $x + \nabla_2^{\mathrm{n}-}(x)/2 < y \circ z$, holds.

$y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow}$. Assume first that $x < y \boxdot_{\mathrm{n}} z$. In this case, by Definition 5, since $x \in \mathbb{F}$ and $x < y \boxdot_{\mathrm{n}} z$, it must be the case that $x < y \circ z$. Then, we can conclude that $x + \nabla_2^{\mathrm{n}-}(x)/2 < x < y \circ z$. Therefore, $x + \nabla_2^{\mathrm{n}-}(x)/2 < y \circ z$, holds. Assume now that $x = y \boxdot_{\mathrm{n}} z$ and even(x). In this case, by Proposition 6, we have that $\nabla_2^{\mathrm{n}-}\big([y \circ z]_{\mathrm{n}}\big)/2 \leq (y \circ z) - [y \circ z]_{\mathrm{n}}$. Since, in this case $x = y \boxdot_{\mathrm{n}} z$, we obtain $\nabla_2^{\mathrm{n}-}(x)/2 \leq (y \circ z) - x$. Hence, $x + \nabla_2^{\mathrm{n}-}(x)/2 \leq y \circ z$. If odd$(x)$, by Proposition 6, we have that $\nabla_2^{\mathrm{n}-}\big([y \circ z]_{\mathrm{n}}\big)/2 < (y \circ z) - [y \circ z]_{\mathrm{n}}$. Hence, $x + \nabla_2^{\mathrm{n}-}(x)/2 < y \circ z$.

Consider now the case that $y \boxdot_{\mathrm{n}} z = +0$ or $y \boxdot_{\mathrm{n}} z = -0$. If $y \circ z \neq 0$, then $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\downarrow}$ or $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow}$. In this case we can reason as above. Assume then that $y \circ z = 0$. Since $x \preccurlyeq +0$ or $x \preccurlyeq -0$ implies that $x \leq 0$. Therefore, we can conclude that $x + \nabla_2^{\mathrm{n}-}(x) < x \leq 0$ holds. Assume now that $y \boxdot_{\mathrm{n}} z = +\infty$. If $y \circ z \neq \infty$ then $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow}$. In this case we can reason as above. On the other hand if $y \circ z = +\infty$ then $x + \nabla_2^{\mathrm{n}-}(x) \leq \infty$ holds.

In order to prove (14), remember that that $x \succcurlyeq y \boxdot_{\downarrow} z$ implies that $x \geq y \boxdot_{\downarrow} z$. Note that $x + \nabla^{\downarrow}(x) = \mathrm{succ}(x)$. So we are left to prove $\mathrm{succ}(x) > y \circ z$. Assume now that $-f_{\max} < y \circ z < 0$ or $f_{\min} < y \circ z \leq f_{\max}$. Note that it cannot be the case that $\mathrm{succ}(x) \leq y \circ z$, otherwise,by Definition 5, $y \boxdot_{\downarrow} z \geq \mathrm{succ}(x)$ and $x \geq y \boxdot_{\downarrow} z$ would not hold. Then, in this case, we can conclude that $\mathrm{succ}(x) > y \circ z$. Next, assume that $0 < y \circ z < f_{\min}$. In this case $y \boxdot_{\downarrow} z = +0$. Hence, $x \geq 0$. By Definition 4, $\mathrm{succ}(x) \geq f_{\min}$. Hence $\mathrm{succ}(x) \geq y \circ z$, holds. Next, assume $y \circ z < -f_{\max}$. In this case $y \boxdot_{\downarrow} z = -\infty$. Hence $x \geq -\infty$. By Definition 4, $\mathrm{succ}(x) \geq -f_{\max}$. Hence $\mathrm{succ}(x) > y \circ z$, holds. Next assume $y \circ z = 0$. In this case $y \boxdot_{\downarrow} z = +0$ or $y \boxdot_{\downarrow} z = -0$. In any case, $x \geq 0$. By Definition 4, $\mathrm{succ}(x) \geq f_{\min}$. Hence $\mathrm{succ}(x) > y \circ z$, holds. Finally assume $y \circ z = -\infty$. In this case $y \boxdot_{\downarrow} z = -\infty$. Hence, since $x \succcurlyeq -\infty$, $x \geq -\infty$. By Definition 4, $\mathrm{succ}(x) \geq -f_{\max}$. Hence, $\mathrm{succ}(x) > y \circ z$, holds.

In order to prove (15), as before, observe that $x \succcurlyeq y \boxdot_{\uparrow} z$ implies that $x \geq y \boxdot_{\uparrow} z$. Assume first that $y \boxdot_{\uparrow} z \in \mathbb{R}_+ \cup \mathbb{R}_-$. In this case, $y \boxdot_{\uparrow} z = [y \circ z]_{\uparrow}$. By Proposition 5, $y \boxdot_{\uparrow} z = [y \circ z]_{\uparrow} \geq y \circ z$. Then, assume that $y \boxdot_{\uparrow} z = -\infty$. In this case, since the rounding towards plus infinity never rounds to $-\infty$, it follows that $y \boxdot_{\uparrow} z = y \circ z$. Hence, $x \geq y \circ z = -\infty$, holds. Assume now that $y \boxdot_{\uparrow} z = +\infty$. In this case, $x = +\infty$ then $x \geq y \circ z$, holds. Finally, assume that $y \boxdot_{\uparrow} z = +0$ or $y \boxdot_{\uparrow} z = -0$. In any case $x \succcurlyeq -0$ that implies $x \geq 0$. On the other hand, we have two cases, $y \circ z \neq 0$ or $y \circ z = 0$. For the first case, by Definition 5, $-f_{\min} < y \circ z < 0$, then $x \geq y \circ z$, holds. For the second case, since $x \geq 0$ then $x \geq y \circ z$.

In order to prove (16), note that $x \succcurlyeq y \boxdot_{\mathrm{n}} z$ implies that $x \geq y \boxdot_{\mathrm{n}} z$. First observe that for $x \neq +\infty$, $x + \nabla_2^{\mathrm{n}+}(x)/2 > x$. Indeed, assume first that $x \neq f_{\max}$, then, by Definition 9, $\nabla_2^{\mathrm{n}+}(x) = x - \mathrm{pred}(x)$. Hence $x + \nabla_2^{\mathrm{n}+}(x)/2 = x + (x - \mathrm{pred}(x))/2 = (3x - \mathrm{pred}(x))/2$. Since $x > \mathrm{pred}(x)$, we can conclude that $x + \nabla_2^{\mathrm{n}+}(x)/2 > x$. Assume now that $x = f_{\max}$. By Definition 9, $\nabla_2^{\mathrm{n}+}(x) = \mathrm{succ}(x) - x$. Hence $x + \nabla_2^{\mathrm{n}+}(x)/2 = x + (\mathrm{succ}(x) - x)/2 = (x + \mathrm{succ}(x))/2$. Since $x < \mathrm{succ}(x)$, we can conclude that $x + \nabla_2^{\mathrm{n}+}(x)/2 > x$.

By Definition 5, we have to consider the following case for $x \boxdot_{\mathrm{n}} y \in \mathbb{R}_+ \cup \mathbb{R}_-$

$y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow}$. In this case, by Proposition 5, $x + \nabla_2^{\mathrm{n}+}(x)/2 > x \geq y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow} \geq y \circ z$. Therefore, $x + \nabla_2^{\mathrm{n}+}(x)/2 > y \circ z$, holds.

$y \boxdot_{\mathrm{n}} z = [y \circ z]_{\downarrow}$. Assume first that $x > y \boxdot_{\mathrm{n}} z$. In this case, by Definition 5, since $x \in \mathbb{F}$ and $x > y \boxdot_{\mathrm{n}} z$, it must be the case that $x > y \circ z$. Hence, by Proposition 5, $x + \nabla_2^{\mathrm{n}+}(x)/2 > x > y \circ z$. Therefore, $x + \nabla_2^{\mathrm{n}+}(x)/2 > y \circ z$, holds. Assume now that $x = y \boxdot_{\mathrm{n}} z$ and even(x). In this case, by Proposition 6, we have that $\nabla_2^{\mathrm{n}+}\big([y \circ z]_{\mathrm{n}}\big)/2 \geq (y \circ z) - [y \circ z]_{\mathrm{n}}$. Since, in this case $x = y \boxdot_{\mathrm{n}} z$, we obtain $\nabla_2^{\mathrm{n}+}(x)/2 \geq (y \circ z) - x$. Hence, $x + \nabla_2^{\mathrm{n}+}(x)/2 \geq y \circ z$. If odd$(x)$, by Proposition 6, we have that $\nabla_2^{\mathrm{n}+}\big([y \circ z]_{\mathrm{n}}\big)/2 > y \circ z - [y \circ z]_{\mathrm{n}}$. Hence, $x + \nabla_2^{\mathrm{n}+}(x)/2 > y \circ z$.

Consider now the case that $y \boxdot_{\mathrm{n}} z = +0$ or $y \boxdot_{\mathrm{n}} z = -0$. If $y \circ z \neq 0$, then $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\downarrow}$ or $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\uparrow}$. In this case we can reason as above. Assume now that $y \circ z = 0$. Since $x \succcurlyeq +0$ or $x \succcurlyeq -0$ implies that $x \geq 0$, we can conclude that $x + \nabla_2^{\mathrm{n}+}(x)/2 > x \geq 0$ holds. Assume now that $y \boxdot_{\mathrm{n}} z = -\infty$. If $y \circ z \neq -\infty$ then $y \boxdot_{\mathrm{n}} z = [y \circ z]_{\downarrow}$. In this case we can reason as above. On the other hand if $y \circ z = -\infty$ then $x + \nabla_2^{\mathrm{n}+}(x)/2 \geq -\infty$ holds.

□

Proof (of Proposition 4). We first prove (19). By Proposition 5, $e \geq [e]_{\downarrow}$. Hence, $x \geq [e]_{\downarrow}$. Since by hypothesis, $e \in E_{\mathbb{F}}$ is an expression that evaluates on $\overline{\mathbb{R}}$ to a nonzero value, we have three cases:

$[e]_{\downarrow} \neq 0$ **and** $x \neq 0$**:** In this case $x \geq [e]_{\downarrow}$ implies $x \succcurlyeq [e]_{\downarrow}$.

$[e]_{\downarrow} = +0$**:** In this case, $0 < e < f_{\min}$. Then, it must be the case that $x > 0$. Therefore $x \succcurlyeq [e]_{\downarrow}$ holds.

$x = 0$**:** In this case x must be strictly greater than e since $e \in E_{\mathbb{F}}$ evaluates to a nonzero value. Therefore, $e < 0$. Hence, by Definition 5, $[e]_{\downarrow} \leq -f_{\min}$. Then $x \succcurlyeq [e]_{\downarrow}$ holds.

In all cases, we have that $x \succcurlyeq [e]_{\downarrow}$. By Definition 10, we conclude that $x \succcurlyeq [\![e]\!]_{\downarrow}$.

We now prove (20). By Proposition 5, similarly as in the previous case, $e \geq [e]_{\downarrow}$. Hence, $x > [e]_{\downarrow}$. Since by hypothesis, $e \in E_{\mathbb{F}}$ is an expression that evaluates on $\overline{\mathbb{R}}$ to a nonzero value, we have three cases:

$[e]_{\downarrow} \neq 0$ **and** $x \neq 0$**:** In this case $x > [e]_{\downarrow}$ implies $x \succ [e]_{\downarrow}$.

$[e]_{\downarrow} = +0$**:** In this case, $0 < e < f_{\min}$. Hence, $x > 0$. Therefore $x \succ [e]_{\downarrow}$ holds.

$x = 0$**:** In this case x must be strictly greater than e since $e \in E_{\mathbb{F}}$ evaluates to a nonzero value. Therefore, $e < 0$. Hence, by Definition 5, $[e]_{\downarrow} \leq -f_{\min}$. Then $x \succ [e]_{\downarrow}$ holds.

In all cases, we have that $x \succ [e]_{\downarrow}$. By Definition 10, we conclude that $x \succ [\![e]\!]_{\downarrow}$. Then, by Definition 4, we have the following cases on $[\![e]\!]_{\downarrow}$:

$[\![e]\!]_{\downarrow} = f_{\max}$**:** In this case $\mathrm{succ}([\![e]\!]_{\downarrow}) = +\infty$. Since $x \succ [\![e]\!]_{\downarrow}$, this implies that $x = +\infty$. Then $x \succcurlyeq \mathrm{succ}([\![e]\!]_{\downarrow})$, holds.

$-f_{\max} \leq [\![e]\!]_{\downarrow} < -f_{\min}$ **or** $f_{\min} \leq [\![e]\!]_{\downarrow} < f_{\max}$**:** In this case $\mathrm{succ}([\![e]\!]_{\downarrow}) = \min\{\, y \in \mathbb{F} \mid y > [\![e]\!]_{\downarrow} \,\}$. Since $x > [\![e]\!]_{\downarrow}$, $x \in \{\, y \in \mathbb{F} \mid y > [\![e]\!]_{\downarrow} \,\}$. Hence, $x \succcurlyeq \mathrm{succ}([\![e]\!]_{\downarrow})$, holds.

$[\![e]\!]_\downarrow = +0$ **or** $[\![e]\!]_\downarrow = -0$**:** In this case $\mathrm{succ}([\![e]\!]_\downarrow) = f_{\min}$. Since $x > [\![e]\!]_\downarrow$, this implies that $x \geq f_{\min}$. Hence, $x \succcurlyeq \mathrm{succ}([\![e]\!]_\downarrow)$, holds.
$[\![e]\!]_\downarrow = -f_{\min}$ **:** In this case $\mathrm{succ}([\![e]\!]_\downarrow) = -0$. Since $x > [\![e]\!]_\downarrow = -f_{\min}$, $x \succcurlyeq -0$. Hence, $x \succcurlyeq \mathrm{succ}([\![e]\!]_\downarrow)$, holds.
$[\![e]\!]_\downarrow = -\infty$ **:** In this case $\mathrm{succ}([\![e]\!]_\downarrow) = -f_{\max}$. Since $x > [\![e]\!]_\downarrow = -\infty$, $x \succcurlyeq -f_{\max}$. Hence, $x \succcurlyeq \mathrm{succ}([\![e]\!]_\downarrow)$, holds.

We now prove (21). By Proposition 5, $e \leq [e]_\uparrow$. Hence, analogously as before, $x \leq [e]_\uparrow$. Since by hypothesis, $e \in E_{\mathbb{F}}$ is an expression that evaluates on $\overline{\mathbb{R}}$ to a nonzero value, we have three cases:

$[e]_\uparrow \neq 0$ **and** $x \neq 0$**:** In this case $x \leq [e]_\uparrow$ implies $x \preccurlyeq [e]_\uparrow$.
$[e]_\uparrow = -0$**:** In this case, $-f_{\min} < e < 0$. Hence, $x < 0$. Therefore $x \preccurlyeq [e]_\uparrow$ holds.
$x = 0$**:** In this case it must be the case that x is strictly smaller than e, since $e \in E_{\mathbb{F}}$ evaluates to a nonzero value. Therefore, $e > 0$. Hence, by Definition 5, $[e]_\uparrow \geq f_{\min}$. Then $x \preccurlyeq [e]_\uparrow$ holds.

In any case, $x \preccurlyeq [e]_\uparrow$ holds. By Definition 10, we conclude that $x \preccurlyeq [\![e]\!]_\uparrow$.

Next we prove (22). By Proposition 5, $e \leq [e]_\uparrow$. Hence, $x < [e]_\uparrow$. Since by hypothesis, $e \in E_{\mathbb{F}}$ is an expression that evaluates on $\overline{\mathbb{R}}$ to a nonzero value, we have three cases:

$[e]_\uparrow \neq 0$ **and** $x \neq 0$**:** In this case $x < [e]_\uparrow$ implies $x \prec [e]_\uparrow$.
$[e]_\uparrow = -0$**:** In this case, $-f_{\min} < e < 0$. Hence, $x < 0$. Therefore $x \prec [e]_\uparrow$ holds.
$x = 0$**:** In this case it must be the case that x is strictly smaller than e, since $e \in E_{\mathbb{F}}$ evaluates to a nonzero value. Therefore, $e > 0$. Hence, by Definition 5, $[e]_\uparrow \geq f_{\min}$. Then $x \prec [e]_\uparrow$ holds.

In any case, $x \prec [e]_\uparrow$ holds. By Definition 10, we conclude that $x \prec [\![e]\!]_\uparrow$. By Definition 4, we have the following cases on $[\![e]\!]_\uparrow$:

$[\![e]\!]_\uparrow = -f_{\max}$**:** In this case $\mathrm{pred}([\![e]\!]_\uparrow) = -\infty$. Since $x \prec [\![e]\!]_\uparrow$, this implies that $x = -\infty$. Then $x \preccurlyeq \mathrm{pred}([\![e]\!]_\uparrow)$, holds.
$f_{\min} < [\![e]\!]_\uparrow \leq f_{\max}$ **or** $-f_{\max} < [\![e]\!]_\uparrow \leq -f_{\min}$**:** In this case $\mathrm{pred}([\![e]\!]_\uparrow) = \max\{\, y \in \mathbb{F} \mid y < [\![e]\!]_\uparrow \,\}$. Since $x < [\![e]\!]_\uparrow$, $x \in \{\, y \in \mathbb{F} \mid y < [\![e]\!]_\uparrow \,\}$. Hence, $x \preccurlyeq \mathrm{pred}([\![e]\!]_\uparrow)$, holds.
$[\![e]\!]_\uparrow = +0$ **or** $[\![e]\!]_\uparrow = -0$**:** In this case $\mathrm{pred}([\![e]\!]_\uparrow) = -f_{\min}$. Since $x < [\![e]\!]_\uparrow$, this implies that $x \leq -f_{\min}$. Hence, $x \preccurlyeq \mathrm{pred}([\![e]\!]_\uparrow)$, holds.
$[\![e]\!]_\uparrow = f_{\min}$ **:** In this case $\mathrm{pred}([\![e]\!]_\uparrow) = +0$. Since $x < [\![e]\!]_\uparrow = f_{\min}$, $x \preccurlyeq +0$. Hence, $x \preccurlyeq \mathrm{pred}([\![e]\!]_\uparrow)$, holds.
$[\![e]\!]_\downarrow = +\infty$ **:** In this case $\mathrm{pred}([\![e]\!]_\uparrow) = f_{\max}$. Since $x < [\![e]\!]_\uparrow = \infty$, $x \preccurlyeq f_{\max}$. Hence, $x \preccurlyeq \mathrm{pred}([\![e]\!]_\uparrow)$, holds.

In order to prove (23) we first want to prove that $x \succcurlyeq [e]_\uparrow$. To this aim consider the following cases for e:

$e > f_{\max}$**:** In this case $[e]_\uparrow = +\infty$. On the hand, $x \geq e > f_{\max}$. Since $x \in \mathbb{F}$ implies that $x = +\infty$. Hence $x \succcurlyeq [e]_\uparrow$.

$e \leq -f_{\min}$ **or** $0 < e \leq f_{\max}$**:** In this case $[e]_\uparrow = \min\{\, z \in \mathbb{F} \mid z \geq e \,\}$. Since $x \geq e$, $x \in \{\, z \in \mathbb{F} \mid z \geq e \,\}$. Hence, $x \geq [e]_\uparrow$, holds and also $x \succcurlyeq [e]_\uparrow$.
$-f_{\min} < e < 0$**:** In this case $[e]_\uparrow = -0$. Since $x \geq e$ and $x \in \mathbb{F}$, $x \succcurlyeq -0$, holds.
$e = -\infty$**:** In this case $[e]_\uparrow = -\infty$ and $x \succcurlyeq -\infty$ holds.

Since by hypothesis $[e]_\uparrow = [\![e]\!]_\uparrow$, we can conclude that $x \succcurlyeq [\![e]\!]_\uparrow$ holds.

In order to prove (24) we first want to prove that $x \preccurlyeq [e]_\downarrow$. To this aim consider the following cases for e:

$e < -f_{\max}$**:** In this case $[e]_\downarrow = -\infty$. On the hand, $x \leq e < -f_{\max}$. Since $x \in \mathbb{F}$ implies that $x = -\infty$. Hence $x \preccurlyeq [e]_\downarrow$.
$e \geq f_{\min}$ **or** $-f_{\max} \leq e < 0$**:** In this case $[e]_\downarrow = \max\{\, z \in \mathbb{F} \mid z \leq e \,\}$. Since $x \leq e$, $x \in \{\, z \in \mathbb{F} \mid z \leq e \,\}$. Hence, $x \leq [e]_\downarrow$, holds and also $x \preccurlyeq [e]_\downarrow$.
$0 < e < f_{\min}$**:** In this case $[e]_\downarrow = +0$. Since $x \leq e$ and $x \in \mathbb{F}$, $x \preccurlyeq +0$, holds.
$e = +\infty$**:** In this case $[e]_\downarrow = \infty$ and $x \preccurlyeq +\infty$ holds.

Since by hypothesis $[e]_\downarrow = [\![e]\!]_\downarrow$, we can conclude that $x \preccurlyeq [\![e]\!]_\downarrow$ holds.

B.3 Proofs of Results in Section 4.3

Proof (of Theorem 1). Given the constraint $x = y \boxplus_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 1 sets $X' = [x'_l, x'_u] \cap X$; hence, we are sure $X' \subseteq X$. Moreover, by Proposition 1, for each $y \in Y$, $z \in Z$ and $r \in S$, we are sure that $y \boxplus_{r_l} z \leq y \boxplus_r z \leq y \boxplus_{r_u} z$; by monotonicity of $\boxplus$, we have $y_l \boxplus_{r_l} z_l \leq y \boxplus_{r_l} z \leq y \boxplus_r z \leq y \boxplus_{r_u} z \leq y_u \boxplus_{r_u} z_u$. Therefore, we can focus on finding a lower bound for $y_l \boxplus_{r_l} z_l$ and an upper bound for $y_u \boxplus_{r_u} z_u$.

Such bounds are given by the functions da_l and da_u of Figure (2). Almost all of the cases reported in the tables can be trivially derived from the definition of the addition operation in the IEEE 754 Standard [IEE08]; just two cases need further explanation. Concerning the entry of da_l in which $y_l = -\infty$ and $z_l = +\infty$, note that $z_l = +\infty$ implies $z_u = +\infty$. Then for any $y > y_l = -\infty$, $y \boxplus +\infty = +\infty$. On the other hand, by the IEEE 754 Standard [IEE08], $-\infty \boxplus +\infty$ is an invalid operation. For the simmetric case, i.e.. the entry of da_u in which $y_u = -\infty$ and $z_u = +\infty$, we can reason dually.

We are now left to prove that $\forall X'' \subset X : \exists r \in S, y \in Y, z \in Z : y \boxplus_r z \notin X''$. Let us focus on the lower bound x'_l, proving that there always exists a $r \in S$ such that $y_l \boxplus_r z_l = x'_l$. First, consider the cases in which $y_l \notin (\mathbb{R}_- \cup \mathbb{R}_+)$ or $z_l \notin (\mathbb{R}_- \cup \mathbb{R}_+)$. In these cases, a brute-force verification successfully proved that $\mathrm{da}_l(y_l, z_l, r_l)$ is equal to $y_l \boxplus_{r_l} z_l$. For the cases in which $y_l \in (\mathbb{R}_- \cup \mathbb{R}_+)$ and $z_l \in (\mathbb{R}_- \cup \mathbb{R}_+)$ we have $x'_l = y_l \boxplus_{r_l} z_l$, by definition of da_l of Figure (2). Remember that, by Proposition 1, there exists $r \in S$ such that $y_l \boxplus_{\mathrm{r_l}} z_l = y_l \boxplus_{\mathrm{r}} z_l$. Since $y_l \in Y$ and $z_l \in Z$, we can conclude that for any $X'' \subseteq X'$, $x'_l \notin X''$ implies $y_l \boxplus_{\mathrm{r}} z_l \notin X''$. An analogous reasoning allows us to conclude that there exists an $r \in S$ for which the following holds: for any $X'' \subseteq X'$, $x'_u \notin X''$ implies $y_u \boxplus_{\mathrm{r}} z_u \notin X''$.

Proof (of Theorem 2).

Given the constraint $x = y \boxplus_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 2 computes a new and refined domain Y' for variable y.

Firstly, observe that the newly computed interval $[y'_l, y'_u]$ is either intersected with the old domain Y, so that $Y' = [y'_l, y'_u] \cap Y$, or set to $Y' = \varnothing$. Hence, we are sure that $Y' \subseteq Y$ holds.

Proposition 1 and the monotonicity of $\boxplus$ allow us to find a lower bound for y by exploiting the constraint $y \boxplus_{\bar{r}_l} z_u = x_l$, and an upper bound for y by exploiting the constraint $y \boxplus_{\bar{r}_u} z_l = x_u$. We will now prove that the case analyses of functions ia_l, described in Figure 3, and ia_u, described in Figure 4, express such bounds correctly.

Concerning the operand combinations in which ia_l takes the value described by the case analysis a_4, remember that, by the IEEE 754 Standard [IEE08], whenever the sum of two operands with opposite sign is zero, the result of that sum is $+0$ in all rounding-direction attributes except roundTowardNegative: in that case the result is -0. Then, since $z_u \boxplus_{\downarrow} (-z_u) = -0$, when $\bar{r}_l = \downarrow$, y_l can safely be set to $\mathrm{succ}(-z_u)$.

As for the case in which ia_l takes one of the values determined by a_5, the IEEE 754 Standard [IEE08] asserts that $+0 \boxplus_{\downarrow} +0 = +0$, while $-0 \boxplus_{\downarrow} +0 = -0$: the correct lower bound for y is $y'_l = +0$, in this case. As we already pointed out, for any other rounding-direction attribute $+0 \boxplus -0 = +0$ holds, which allows us to include -0 in the new domain.

Concerning cases of ia_l that give the result described by the case analysis a_6, we clearly must have $y = +\infty$ if $\bar{r}_l = \downarrow$; if $\bar{r}_l = \uparrow$, it should be $y + z_u > f_{\max}$ and thus $y > f_{\max} - z_u$ and, by (22) of Proposition 4, $y \succcurlyeq \mathrm{succ}(f_{\max} \boxminus_{\downarrow} z_u)$. If $\bar{r}_l = \mathrm{n}$, there are two cases:

$z_u < \nabla_2^{\mathrm{n}+}(f_{\max})/2$. In this case, y must be greater than $f_{\max}$, since $f_{\max} + z_u < f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2$ implies that $f_{\max} \boxplus_{\mathrm{n}} z_u = f_{\max} < +\infty$. Note that in this case $\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u \geq f_{\min}$, hence $f_{\max} \boxplus_{\uparrow} \left(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u\right) = +\infty$.

$z_u \geq \nabla_2^{\mathrm{n}+}(f_{\max})/2$. Since $\mathrm{odd}(f_{\max})$, for $x_l = +\infty$ we need y to be greater than or equal to $f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u$. Note that $y \geq f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u$ together with

$$\left[f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u\right]_{\uparrow} = f_{\max} \boxplus_{\uparrow} \left(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u\right) \tag{36}$$

allows us to apply (23) of Proposition 4 in order to conclude that $y \succcurlyeq f_{\max} \boxplus_{\uparrow} \left(\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u\right)$.
Equality (36) holds because either the application of '$\boxminus_{\uparrow}$' is exact or the application of '$\boxplus_{\uparrow}$' is exact. In fact, since $z_u = m \cdot 2^e \geq \nabla_2^{\mathrm{n}+}(f_{\max})/2 = 2^{e_{\max}-p}$, for some $1 \leq m < 2$, there are two cases: either $e = e_{\max}$ or $e_{\max} - p \leq e < e_{\max}$. Suppose first that $e = e_{\max}$, hence

$$\begin{aligned} \nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u &= 2^{e_{\max}-p} - m \cdot 2^{e_{\max}} \\ &= -2^{e_{\max}}(m - 2^{-p}), \end{aligned}$$

and thus

$$\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u = \begin{cases} -2^{e_{\max}}(m - 2^{1-p}), & \text{if } m > 1; \\ -2^{e_{\max}-1}(2 - 2^{1-p}), & \text{if } m = 1. \end{cases}$$

Since if $e = e_{\max}$ the application of '$\boxminus_{\uparrow}$' is not exact, we prove that the application of '$\boxplus_{\uparrow}$' is exact. Hence, if $m > 1$, we prove that

$$\begin{aligned} f_{\max} + (\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u) &= 2^{e_{\max}}(2 - 2^{1-p}) - 2^{e_{\max}}(m - 2^{1-p}) \\ &= 2^{e_{\max}}(2 - 2^{1-p} - m + 2^{1-p}) \\ &= 2^{e_{\max}}(2 - m) \\ &= 2^{e_{\max}-k}\big(2^k(2 - m)\big) \end{aligned}$$

where $k \stackrel{\text{def}}{=} -\lfloor \log_2(2 - m) \rfloor$. It is worth noting that $2^k(2 - m)$ can be represented by a normalized mantissa; moreover, since $1 \leq k \leq p - 1$, $e_{\min} \leq e_{\max} - k \leq e_{\max}$, hence, $f_{\max} + (\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u) \in \mathbb{F}$. If, instead, $m = 1$,

$$\begin{aligned} f_{\max} + (\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u) &= 2^{e_{\max}}(2 - 2^{1-p}) - 2^{e_{\max}-1}(2 - 2^{1-p}) \\ &= (2^{e_{\max}} - 2^{e_{\max}-1})(2 - 2^{1-p}) \\ &= 2^{e_{\max}-1}(2 - 2^{1-p}) \end{aligned}$$

and, also in this case, $f_{\max} + (\nabla_2^{\mathrm{n}+}(f_{\max})/2 \boxminus_{\uparrow} z_u) \in \mathbb{F}$.

Suppose now that $e_{\max} - p \leq e < e_{\max}$ and let $h \stackrel{\text{def}}{=} e - e_{\max} + p$ so that $0 \leq h \leq p - 1$. In this case we show that the application of '$\boxminus_{\uparrow}$' is exact. Indeed, we have

$$\begin{aligned} \nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u &= 2^{e_{\max}-p} - m \cdot 2^e \\ &= -2^{e_{\max}-p}(m \cdot 2^h - 1) \\ &= -2^{e_{\max}-p+h}(m - 2^{-h}). \end{aligned}$$

If $e = e_{\max} - p$ and $m = 1$, then $h = 0$, $m - 2^{-h} = 0$ and thus $\nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u = 0$. Otherwise, let $k \stackrel{\text{def}}{=} -\lfloor \log_2(m - 2^{-h}) \rfloor$. We have

$$\nabla_2^{\mathrm{n}+}(f_{\max})/2 - z_u = -2^{e_{\max}-p+h-k}\big(2^k(m - 2^{-h})\big),$$

which is an element of $\mathbb{F}$.

Dual arguments w.r.t. the ones used to justify cases of ia_l that give the result described by a_4, a_6 and a_5 can be used to justify the cases of ia_u described by the case analyses a_9, a_{10} and a_7.

We now tackle the cases of ia_l that give the result described by the case analysis a_3 and the cases of ia_u that give the result described by the case analysis

a_8. Exploiting $x \preccurlyeq y \boxplus z$ and $x \succcurlyeq y \boxplus z$, by Proposition 3, we have

$$y + z \begin{cases} \geq x, & \text{if } \bar{r}_l = \downarrow; \\ > x + \nabla^{\uparrow}(x) = \mathrm{pred}(x), & \text{if } \bar{r}_l = \uparrow; \\ \geq x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{37}$$

$$y + z \begin{cases} < x + \nabla^{\downarrow}(x) = \mathrm{succ}(x), & \text{if } \bar{r}_u = \downarrow; \\ \leq x, & \text{if } \bar{r}_u = \uparrow; \\ \leq x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{38}$$

The same case analysis gives us

$$y \begin{cases} \geq x - z, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x) - z, & \text{if } \bar{r}_l = \uparrow \\ \geq x + \nabla_2^{\mathrm{n}-}(x)/2 - z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > x + \nabla_2^{\mathrm{n}-}(x)/2 - z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x); \end{cases} \tag{39}$$

$$y \begin{cases} < \mathrm{succ}(x) - z, & \text{if } \bar{r}_u = \downarrow; \\ \leq x - z, & \text{if } \bar{r}_u = \uparrow; \\ \leq x + \nabla_2^{\mathrm{n}+}(x)/2 - z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < x + \nabla_2^{\mathrm{n}+}(x)/2 - z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{40}$$

We can now exploit the fact that $x \in [x_l, x_u]$ and $z \in [z_l, z_u]$ with $x_l, x_u, z_l, z_u \in \mathbb{F}$ to obtain, using Proposition 2 and the monotonicity of 'pred' and 'succ':

$$y \begin{cases} \geq x_l - z_u, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x_l) - z_u, & \text{if } \bar{r}_l = \uparrow; \\ \geq x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 - z_u, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x_l); \\ > x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 - z_u, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x_l). \end{cases} \tag{41}$$

$$y \begin{cases} < \mathrm{succ}(x_u) - z_l, & \text{if } \bar{r}_u = \downarrow; \\ \leq x_u - z_l, & \text{if } \bar{r}_u = \uparrow; \\ \leq x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x_u); \\ < x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x_u). \end{cases} \tag{42}$$

We can now exploit Proposition 4 and obtain

$$y'_l \stackrel{\mathrm{def}}{=} \begin{cases} -0, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l = z_u; \\ x_l \boxminus_{\uparrow} z_u, & \text{if } \bar{r}_l = \downarrow \text{ and } x_l \neq z_u; \\ \mathrm{succ}\big(\mathrm{pred}(x_l) \boxminus_{\downarrow} z_u\big), & \text{if } \bar{r}_l = \uparrow; \end{cases} \tag{43}$$

and

$$y'_u \stackrel{\text{def}}{=} \begin{cases} \operatorname{pred}\big(\operatorname{succ}(x_u) \boxminus_\uparrow z_l\big), & \text{if } \bar{r}_u = \downarrow; \\ +0, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u = z_l; \\ x_u \boxminus_\downarrow z_l, & \text{if } \bar{r}_u = \uparrow \text{ and } x_u \neq z_l. \end{cases} \tag{44}$$

In fact, if $x_l = z_u$, then, according to IEEE 754 [IEE08, Section 6.3], for each non-NaN, nonzero and finite $w \in \mathbb{F}$, -0 is the least value for y that satisfies $w = y \boxplus_\downarrow w$. If $x_l \neq z_u$, then case (23) of Proposition 4 applies and we have $y \succcurlyeq x_l \boxminus_\uparrow z_u$. Suppose now that $\operatorname{pred}(x_l) = z_u$, then $\operatorname{pred}(x_l) \boxminus_\downarrow z_u \equiv 0$ and $\operatorname{succ}\big(\operatorname{pred}(x_l) \boxminus_\downarrow z_u\big) = f_{\min}$, coherently with the fact that, for each non-NaN, nonzero and finite $w \in \mathbb{F}$, $f_{\min}$ is the least value for y that satisfies $w = y \boxplus_\uparrow \operatorname{pred}(w)$. If $\operatorname{pred}(x_l) \neq z_u$, then case (20) of Proposition 4 applies and we have $y \succcurlyeq \operatorname{succ}\big(\operatorname{pred}(x_l) \boxminus_\downarrow z_u\big)$. A symmetric argument justifies (44).

For the remaining cases, we first show that when $\nabla_2^{\mathrm{n}+}(x) = f_{\min}$,

$$\big[x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l\big]_\downarrow = [x_u - z_l]_\downarrow. \tag{45}$$

The previous equality has the following main consequences: we can perform the computation in $\mathbb{F}$, that is, we do not need to compute $\nabla_2^{\mathrm{n}+}(x)/2$ and, since $[x_u - z_l]_\downarrow = [\![x_u - z_l]\!]_\downarrow$, we can apply (24) of Proposition 4, obtaining a tight bound for y'_u.

Let us then prove (45). Suppose that $\nabla_2^{\mathrm{n}+}(x_u) = f_{\min}$. First assume $x_u \neq z_l$. There are two cases:

$[x_u - z_l]_\downarrow = x_u - z_l$: then we have $y \leq [x_u - z_l]_\downarrow = x_u - z_l$ since the addition of $\nabla_2^{\mathrm{n}+}(x_u)/2 = f_{\min}/2$ is insufficient to reach $\operatorname{succ}(x_u - z_l)$, whose distance from $x_u - z_l$ is at least $f_{\min}$.

$[x_u - z_l]_\downarrow < x_u - z_l < [x_u - z_l]_\uparrow$: since by Definition 2 every finite floating-point number is an integral multiple of $f_{\min}$, so are $x_u - z_l$ and $[x_u - z_l]_\uparrow$. Therefore, again, $y \leq [x_u - z_l]_\downarrow$, since the addition of $\nabla_2^{\mathrm{n}+}(x_u)/2 = f_{\min}/2$ $x_u - z_l$ is insufficient to reach $[x_u - z_l]_\uparrow$, whose distance from $x_u - z_l$ is at least $f_{\min}$.

In the case where $x_u = z_l$ we have $\big[x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l\big]_\downarrow = [0 + f_{\min}/2]_\downarrow = +0$. Hence(45) holds. As we have already pointed out, this allows us to apply (24) of Proposition 4 to the case $\nabla_2^{\mathrm{n}+}(x_u) = f_{\min}$, obtaining the bound $y \preccurlyeq [x_u - z_l]_\downarrow$.

Similar arguments can be applied to $\nabla_2^{\mathrm{n}-}(x_l)$ whenever $\nabla_2^{\mathrm{n}-}(x_l) = -f_{\min}$ to prove that $\big[x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 - z_u\big]_\uparrow = [x_l - z_u]_\uparrow$. Then, by (23) of Proposition 4, we obtain $y \succcurlyeq [x_l - z_u]_\uparrow$.

When the terms $\nabla_2^{\mathrm{n}-}(x_l)$ and $\nabla_2^{\mathrm{n}+}(x_u)$ are non negligible, we need to approximate the values of the expressions $e_l \stackrel{\text{def}}{=} x_l + \nabla_2^{\mathrm{n}-}(x_l)/2 - z_u$ and $e_u \stackrel{\text{def}}{=} x_u + \nabla_2^{\mathrm{n}+}(x_u)/2 - z_l$. Hence, we have the cases $[\![e_l]\!]_\uparrow = [e_l]_\uparrow$ and $[\![e_u]\!]_\downarrow = [e_u]_\downarrow$ as well as $[\![e_l]\!]_\uparrow > [e_l]_\uparrow$ and $[\![e_u]\!]_\downarrow < [e_u]_\downarrow$. Thus, when $[\![e_u]\!]_\downarrow < [e_u]_\downarrow$ by (42) and (21) of Proposition 4 we obtain $y \preccurlyeq [\![e_u]\!]_\uparrow$, while, when $[\![e_l]\!]_\downarrow > [e_l]_\downarrow$ by (42) and (19) of Proposition 4 we obtain $y \succcurlyeq [\![e_l]\!]_\downarrow$. Finally, when $\operatorname{odd}(x_u)$, by (42) and (22) of Proposition 4, we obtain $y \preccurlyeq \operatorname{pred}\big([\![e_u]\!]_\uparrow\big)$. Dually, when $\operatorname{odd}(x_l)$

by (41) and (20) of Proposition 4, we obtain $y \succcurlyeq \mathrm{succ}\big([\![e_l]\!]_\downarrow\big)$. Thus, for the case where $\bar{r}_l = \mathrm{n}$ we have

$$y'_l \stackrel{\text{def}}{=} \begin{cases} -0, & \text{if } \nabla_2^{\mathrm{n}-}(x_l) = f_{\min} \text{ and } x_l = z_u; \\ x_l \boxminus_\uparrow z_u, & \text{if } \nabla_2^{\mathrm{n}-}(x_l) = f_{\min} \text{ and } x_l \neq z_u; \\ [\![e_l]\!]_\uparrow, & \text{if even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq f_{\min} \text{ and } [\![e_l]\!]_\uparrow = [e_l]_\uparrow; \\ [\![e_l]\!]_\downarrow, & \text{if even}(x_l),\ \nabla_2^{\mathrm{n}-}(x_l) \neq f_{\min} \text{ and } [\![e_l]\!]_\uparrow > [e_l]_\uparrow; \\ \mathrm{succ}\big([\![e_l]\!]_\downarrow\big), & \text{otherwise.} \end{cases} \tag{46}$$

whereas, for the case where $\bar{r}_u = \mathrm{n}$, we have

$$y'_u \stackrel{\text{def}}{=} \begin{cases} +0, & \text{if } \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u = z_l; \\ x_u \boxminus_\downarrow z_l, & \text{if } \nabla_2^{\mathrm{n}+}(x_u) = f_{\min} \text{ and } x_u \neq z_l; \\ [\![e_u]\!]_\downarrow, & \text{if even}(x_u),\ \nabla_2^{\mathrm{n}+}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\downarrow = [e_u]_\downarrow; \\ [\![e_u]\!]_\uparrow, & \text{if even}(x_u),\ \nabla_2^{\mathrm{n}-}(x_u) \neq f_{\min} \text{ and } [\![e_u]\!]_\uparrow < [e_u]_\uparrow; \\ \mathrm{pred}\big([\![e_u]\!]_\uparrow\big), & \text{otherwise.} \end{cases} \tag{47}$$

Proof (of Theorem 9). Given the constraint $x = y \boxdot_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, then $X' = [x'_l, x'_u] \cap X$. Hence, we are sure that $X' \subseteq X$.

It should be immediate to verify that function σ of Figure 7, related to the case $\mathrm{sgn}(y_l) = \mathrm{sgn}(y_u)$, chooses the appropriate interval extrema y_L, y_U, z_L, z_U, necessary for computing bounds for x. Indeed, note that such choice is completely driven by the sign of the resulting product. Analogously, the correct interval extrema y_L, y_U, z_L, z_U related to the case $\mathrm{sgn}(z_l) = \mathrm{sgn}(z_u)$ can be determined by applying function σ of Figure 7, but swapping the role of y and z. Hence, if the sign of y or of z is constant (see the second part of Algorithm 9) function σ of Figure 7 finds the appropriate extrema for y and z to compute the bound for x.

Concerning the cases $\mathrm{sgn}(y_l) = \mathrm{sgn}(z_l) = -1$ and $\mathrm{sgn}(y_u) = \mathrm{sgn}(z_u) = 1$ (first part of Algorithm 9), note that we have only two possibilities for the interval extrema y_L and z_L, that are y_l and z_u or y_u and z_l. Since the product of y_L and z_L will have a negative sign in both cases, the right extrema for determining the lower bound x'_l have to be chosen by selecting the smallest product of y_L and z_L. Analogously, for y_U and z_U there are two possibilities: y_l and z_l or y_u and z_u. Since the product of y_U and z_U will have a positive sign in both cases, the appropriate extrema for determining the upper bound x'_u have to be chosen as the biggest product of y_U and z_U.

Remember that by Proposition 1, following the same reasoning as in the previous proofs, it suffices to find a lower bound for $y_L \boxdot_{r_l} z_L$ and an upper bound for $y_U \boxdot_{r_u} z_U$.

We now comment on some critical case analyses of function dm_l of Figure 17. Consider, for example, when $y_L = \pm\infty$ and $z_L = \pm 0$. In particular, we analyze the case in which $y_L = -\infty$ and $z_L = \pm 0$. Note that $y_L = -\infty$ implies $y_l = -\infty$. Assume, first, that $z_L = +0$. Recall that by the IEEE 754 Standard [IEE08] $\pm\infty \boxdot \pm 0$ is an invalid operation. However, since $y_l = -\infty$, we have two cases:

$y_u \geq -f_{\max}$: note that, in this case, $-f_{\max} \boxdot +0 = -0$;

$y_u = -\infty$: in this case, z_L must correspond to z_u (see the last three cases of function σ). Since $-\infty \boxdot z$ for $z < 0$ results in $+\infty$, we can conclude that -0 is a correct lower bound for x.

A similar reasoning applies for the cases $y_L = +\infty$, $z_L = \pm 0$. Dually, the only critical entries of function dm_u of Figure 17 are those in which $y_U = \pm\infty$ and $z_U = \pm 0$. In these cases we can reason in a similar way, too.

We are left to prove that $\forall X'' \subset X : \exists r \in S, y \in Y, z \in Z \,.\, y \boxdot_r z \notin X''$. Let us focus on the lower bound x'_l, proving that there exist values $r \in S, y \in Y, z \in Z$ such that $y \boxdot_r z = x'_l$. Consider the particular values of y_L, z_L and r_l that correspond to the value of x'_l chosen by Algorithm 9, that is y_L, z_L and r_l are such that $\mathrm{dm}_l(y_L, z_L, r_l) = x'_l$. By Algorithm 9, such values of y_L and z_L must exist. First, consider the cases in which $y_L \notin (\mathbb{R}_- \cup \mathbb{R}_+)$ or $z_L \notin (\mathbb{R}_- \cup \mathbb{R}_+)$. In these cases, a brute-force verification was successfully conducted to verify that $y \boxdot_{r_l} z = x'_l$. For the cases in which $y_L \in (\mathbb{R}_- \cup \mathbb{R}_+)$ and $z_L \in (\mathbb{R}_- \cup \mathbb{R}_+)$ we have, by definition of dm_l of Figure (17), that $x'_l = y_L \boxdot_{r_l} z_L$. Remember that, by Proposition 1, there exist $r' \in S$ such that $y_L \boxdot_{\mathrm{r_l}} z_L = y_L \boxdot_{\mathrm{r'}} z_L$. Since $y_L \in Y$ and $z_L \in Z$, we can conclude that $x'_l \notin X''$ implies that $y'_L \boxdot_{\mathrm{r'}} z_L \notin X''$. An analogous reasoning allows us to conclude that $\exists r \in S$ for which the following holds: $x'_u \notin X''$ implies $y_U \boxdot_{\mathrm{r}} z_U \notin X''$.

Proof (of Theorem 10). Given the constraint $x = y \boxdot_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 10 computes Y', a new and refined interval for variable y.

First, note that either $Y' := (Y \cap [y_l^-, y_u^-]) \uplus (Y \cap [y_l^+, y_u^+])$ or $Y' = \varnothing$, hence, in both cases, we are sure that $Y' \subseteq Y$ holds.

By Proposition 1, we can focus on finding a lower bound for $y \in Y$ by exploiting the constraint $y \boxdot_{\bar{r}_l} z = x$ and an upper bound for $y \in Y$ by exploiting the constraint $y \boxdot_{\bar{r}_u} z = x$.

Now, in order to compute correct bounds for y, we first need to split the interval of z into the sign-homogeneous intervals Z_- and Z_+, because it is crucial to be sure of the sign of z. As a consequence, for $W = Y_-$ (and, analogously, for $W = Y_+$) function τ of Figure 5 picks the appropriate interval extrema of X and W to be used to compute the new lower and upper bounds for y. It is easy to verify that the values of x_L and w_L (resp., x_U and w_U) computed using function τ of Figure 5 are the boundaries of X and W upon which y touches its minimum (resp., maximum). Functions im_l of Figure 18 and im_u of Figure 19 are then employed to find the new bounds for y. The so obtained intervals for y are then joined using convex union between intervals, in order to obtain the refining interval for y.

Observe that functions im_l of Fig 18 and im_u of Fig 19 are dual to each other: every row/column of one table can be found in the other table reversed and changed of sign. This is due to the fact that, for each $r \in R$ and each

$D \subseteq \mathbb{F} \times \mathbb{F}$, we have

$$\begin{aligned}
&\min\{\, y \in \mathbb{F} \mid (x,z) \in D, x = y \boxdot_r z \,\} \\
=& -\max\{\, y \in \mathbb{F} \mid (x,z) \in D, -x = y \boxdot_r z \,\} \\
=& -\max\{\, y \in \mathbb{F} \mid (x,z) \in D, x = y \boxdot_r -z \,\}.
\end{aligned}$$

Concerning the case analysis of im_l marked as a_4 of Fig 18, we must consider the following cases:

$\bar{r}_l = \uparrow$: we clearly must have $y = +\infty$ in this case;
$\bar{r}_l = \downarrow$: inequality $y \cdot w_L < -f_{\max}$ must hold and thus, since w_L is negative, $y > -f_{\max}/w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(-f_{\max} \boxslash_\downarrow w_L)$.
$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\max})$, for $x_L = -\infty$ we need y to be greater or equal than $\big(-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L$. If $[\![\big(-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]\!]_\uparrow = [-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]_\uparrow$, by (23) of Proposition 4, we can conclude $y \succcurlyeq [\![\big(-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]\!]_\uparrow$. On the other hand, if $[\![\big(-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]\!]_\uparrow \neq [-f_{\max}+\nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]_\uparrow$, then we can only apply (19) of Proposition 4, obtaining $y \succcurlyeq [\![\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)/w_L]\!]_\downarrow$.

Regarding the case analysis of im_l marked a a_5 of Fig 18, we have the following cases:

$\bar{r}_l = \downarrow$: in this case, we must have $y = -0$;
$\bar{r}_l = \uparrow$: inequality $y \cdot w_L > -f_{\min}$ must hold and thus, since w_L is positive, $y > -f_{\min}/w_L$ and, by (22) of Proposition 4, $y \succcurlyeq \mathrm{succ}(-f_{\min} \boxslash_\downarrow w_L)$.
$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\min})$, for $x_L = -0$ we need y to be greater or equal than $-f_{\min}/(2 \cdot w_L)$. Since in this case $[\![-f_{\min}/(2 \cdot w_L)]\!]_\uparrow = [-f_{\min}/(2 \cdot w_L)]_\uparrow = (-f_{\min}) \boxslash_\uparrow (2 \cdot w_L)$, by (23) of Proposition 4, we can conclude $y \succcurlyeq -f_{\min} \boxslash_\uparrow (2 \cdot w_L)$.

As for the case analysis of im_l marked as a_6 of Figure 18, the following cases must be studied:

$\bar{r}_l = \uparrow$: we must have $y = +0$ in this case;
$\bar{r}_l = \downarrow$: it should be $y \cdot w_L < f_{\min}$ and thus, since w_L is negative, $y > f_{\min}/w_L$ and, by (22) of Proposition 4, $y \succcurlyeq \mathrm{succ}(-f_{\min} \boxslash_\downarrow w_L)$.
$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\min})$, for $x_L = -0$ we need y to be greater than or equal to $\big(f_{\min}/(2 \cdot w_L)\big)$. Since in this case $[\![f_{\min}/(2 \cdot w_L)]\!]_\uparrow = [f_{\min}/(2 \cdot w_L)]_\uparrow = f_{\min} \boxslash_\uparrow (2 \cdot w_L)$, by (23) of Proposition 4 we can conclude $y \succcurlyeq f_{\min} \boxslash_\uparrow (2 \cdot w_L)$.

Finally, for the case analysis of im_l marked as a_7 of Fig 18, the following cases must be considered:

$\bar{r}_l = \downarrow$: in this case we must have $y = +\infty$;
$\bar{r}_l = \uparrow$: it should be $y \cdot w_L > -f_{\max}$ and thus, since w_L is positive, $y > f_{\max}/w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(f_{\max} \boxslash_\downarrow w_L)$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\max})$, for $x_L = +\infty$ we need y to be greater than or equal to $\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L$. If $[\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]\!]_\uparrow = [f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]_\uparrow$, by (23) of Proposition 4, we can conclude $y \succcurlyeq [\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]\!]_\uparrow$. On the other hand, if $[\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]\!]_\uparrow \neq [f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]_\uparrow$, then we can only apply (19) of Proposition 4, obtaining $y \succcurlyeq [\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big)/w_L]\!]_\downarrow$.

Similar arguments can be used to prove the case analyses of im_u of Figure 19 marked as a_9, a_{10}, a_{11} and a_{12}.

We now analyze the case analyses of im_l of Fig 18 marked as a_3^- and a_3^+ and the ones of im_u of Fig 19 marked as a_8^- and a_8^+, for which we can assume $x_L, w_L \in \mathbb{F} \cap \mathbb{R}$ and $x_U, w_U \in \mathbb{F} \cap \mathbb{R}$, and $\mathrm{sgn}(w_l) = \mathrm{sgn}(w_u)$. Exploiting $x \preccurlyeq y \boxdot z$ and $x \succcurlyeq y \boxdot z$, by Proposition 3 we have

$$y \cdot z \begin{cases} \geq x, & \text{if } \bar{r}_l = \downarrow; \\ > x + \nabla^{\uparrow}(x) = \mathrm{pred}(x), & \text{if } \bar{r}_l = \uparrow; \\ \geq x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{48}$$

$$y \cdot z \begin{cases} < x + \nabla^{\downarrow}(x) = \mathrm{succ}(x), & \text{if } \bar{r}_u = \downarrow; \\ \leq x, & \text{if } \bar{r}_u = \uparrow; \\ \leq x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{49}$$

Since the case $z = 0$ is handled separately by im_l of Figure 18 and by im_u of Figure 19, we can assume $z \neq 0$. Thanks to the splitting of Z into a positive and a negative part, the sign of z is determined. In the following, we will prove the case analyses marked as a_3^+ and a_8^+. Hence, assuming $z > 0$, the previous case analysis gives us

$$y \begin{cases} \geq x/z, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x)/z, & \text{if } \bar{r}_l = \uparrow \\ \geq \big(x + \nabla_2^{\mathrm{n}-}(x)/2\big)/z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > \big(x + \nabla_2^{\mathrm{n}-}(x)/2\big)/z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x); \end{cases} \tag{50}$$

$$y \begin{cases} < \mathrm{succ}(x)/z, & \text{if } \bar{r}_u = \downarrow; \\ \leq x/z, & \text{if } \bar{r}_u = \uparrow; \\ \leq \big(x + \nabla_2^{\mathrm{n}+}(x)/2\big)/z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < \big(x + \nabla_2^{\mathrm{n}+}(x)/2\big)/z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{51}$$

Note that the numerator and the denominator of the previous fractions are independent. Therefore, we can find the minimum of the fractions by minimizing the numerator and maximizing the denominator. Since we are analyzing the case in which $W = Z_+$, let (x_L, w_L, x_U, w_U) as the result of function τ of Figure 5.

Hence, by Proposition 2 and the monotonicity of 'pred' and 'succ we obtain

$$y \begin{cases} \geq x_L/w_L, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x_L)/w_L, & \text{if } \bar{r}_l = \uparrow \\ \geq \big(x_L + \nabla_2^{\mathrm{n}-}(x_L)/2\big)/w_L, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > \big(x_L + \nabla_2^{\mathrm{n}-}(x_L)/2\big)/w_L, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x); \end{cases} \tag{52}$$

$$y \begin{cases} < \mathrm{succ}(x_U)/w_U, & \text{if } \bar{r}_u = \downarrow; \\ \leq x_U/w_U, & \text{if } \bar{r}_u = \uparrow; \\ \leq \big(x_U + \nabla_2^{\mathrm{n}+}(x_U)/2\big)/w_U, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < \big(x_U + \nabla_2^{\mathrm{n}+}(x_U)/2\big)/w_U, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{53}$$

We can now exploit Proposition 4 and obtain:

$$y_l' \stackrel{\mathrm{def}}{=} \begin{cases} x_L \boxslash_\uparrow w_L, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}\big(\mathrm{pred}(x_L) \boxslash_\downarrow w_L\big), & \text{if } \bar{r}_l = \uparrow; \end{cases} \tag{54}$$

$$y_u' \stackrel{\mathrm{def}}{=} \begin{cases} \mathrm{pred}\big(\mathrm{succ}(x_U) \boxslash_\uparrow w_U\big), & \text{if } \bar{r}_u = \downarrow; \\ x_U \boxslash_\downarrow w_U, & \text{if } \bar{r}_u = \uparrow. \end{cases} \tag{55}$$

Indeed, if $x_L \neq 0$, then Proposition 4 applies and we have $y \succcurlyeq x_L \boxslash_\uparrow w_L$. On the other hand, if $x_L = 0$, since by hypothesis $z > 0$ implies $w_L > 0$, according to IEEE 754 [IEE08, Section 6.3], we have $(x_L \boxslash_\uparrow w_L) = \mathrm{sgn}(x_L) \cdot 0$ and, indeed, for each non-NaN, nonzero and finite $w \in \mathbb{F} \cap [+0, +\infty]$, $\mathrm{sgn}(x_L) \cdot 0$ is the least value for y that satisfies $\mathrm{sgn}(x_L) \cdot 0 = y \boxdot_\downarrow w$.

Analogously, if $x_L \neq f_{\min}$, then Proposition 4 applies and we have $\mathrm{succ}\big(\mathrm{pred}(x_L) \boxslash_\downarrow w_L\big)$. On the other hand, if $x_L = f_{\min}$, $\mathrm{succ}\big(\mathrm{pred}(x_L) \boxslash_\downarrow w_L\big) = f_{\min}$, which is consistent with the fact that, for each non-NaN, nonzero and finite $w \in \mathbb{F} \cap [+0, +\infty]$, $f_{\min}$ is the lowest value of y that satisfies $f_{\min} = y \boxdot_\uparrow w$.

A symmetric argument justifies (55).

As before, we will consider both the cases $[\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow$ and $[\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow$ as well as $[\![e_l^+]\!]_\uparrow > [e_l^+]_\uparrow$ and $[\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow$. Thus, when $[\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow$ by (53) and (21) of Proposition 4 we obtain $y \preccurlyeq [\![e_u^+]\!]_\uparrow$. Instead, when $[\![e_l^+]\!]_\downarrow > [e_l^+]_\downarrow$, by (53) and (19) of Proposition 4 we obtain $y \succcurlyeq [\![e_l^+]\!]_\downarrow$. In conclusion, for the case in which $\bar{r}_l = \mathrm{n}$, since $e_u \neq 0$ and $e_l \neq 0$, by Proposition 4, we have

$$y_l' \stackrel{\mathrm{def}}{=} \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if } \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if } \mathrm{even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow \neq [e_l^+]_\uparrow; \\ \mathrm{succ}\big([\![e_l^+]\!]_\downarrow\big), & \text{otherwise;} \end{cases} \tag{56}$$

whereas, for the case in which $\bar{r}_u = \mathrm{n}$,

$$y_u' \stackrel{\mathrm{def}}{=} \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if } \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if } \mathrm{even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow \neq [e_u^+]_\downarrow; \\ \mathrm{pred}\big([\![e_u^+]\!]_\uparrow\big), & \text{otherwise.} \end{cases} \tag{57}$$

An analogous reasoning with $z < 0$ allows us to obtain the case analyses marked as a_3^- and a_8^-.

Proof (of Theorem 3). Given the constraint $x = y \boxslash_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 3 computes a new refining interval X' for variable x. Note that $X' = [x'_l, x'_u] \cap X$, which assures us that $X' \subseteq X$.

As for the proof of Theorem 10, it is easy to verify that y_L and w_L (resp., y_U and w_U) computed using function τ of Figure 5, are the boundaries of Y and W upon which x touches its minimum (resp., maximum). Moreover, remember that by Proposition 1, following the same reasoning of the proofs of the previous theorems, we can focus on finding a lower bound for $y_L \boxslash_{r_l} w_L$ and an upper bound for $y_U \boxslash_{r_u} w_U$.

We will now comment only on the most critical entries of function dd_l of Figure 6: let us briefly discuss the cases in which $y_L = -\infty$ and $w_L = \pm\infty$.

$w_L = -\infty$. In this case, by function τ of Figure 5 (see the first three cases), we have $y_L = y_u = -\infty$, while either $w_L = w_l$ or $w_L = w_u$. Since by the IEEE 754 Standard [IEE08] dividing $\pm\infty$ by $\pm\infty$ is an invalid operation, we are left to consider the case $w_L = w_l$. In this case, recall that by the IEEE 754 Standard [IEE08], dividing $-\infty$ by a finite negative number yields $+\infty$. Hence, we can conclude $x_l = +\infty$.

$w_L = +\infty$. By function τ of Figure 5 (see the fourth and last case), we have $y_L = y_l = -\infty$, while $w_L = w_l = +\infty$. Hence, $x_l = -0$, since dividing a negative finite number by $+\infty$ gives -0.

A similar reasoning applies for the cases $y_L = +\infty$, $w_L = \pm\infty$. Dually, the only critical entries of function dd_u of Figure 6 are those in which $y_U = \pm\infty$ and $w_U = \pm\infty$ and can be handled analogously.

We are left to prove that $\forall X'' \subset X, \exists r \in S, y \in Y, z \in Z : y \boxslash_r z \notin X''$. Let us focus on the lower bound x_l^+ proving that, if $[x_l^+, x_l^+] \neq \varnothing$, then there exist $r \in S, y \in Y, z \in Z$ such that $y \boxslash_r z = x_l^+$. Consider the particular values y_L, $z_l = w_L$ and r_l that correspond to x_l^+ in Algorithm 3, i.e. y_L and w_L and r_l are such that $\mathrm{dd}_l(y_L, w_L, r_l) = x_l^+$. By Algorithm 3, such y_L and w_L must exist. First consider the cases in which $y_L \notin (\mathbb{R}_- \cup \mathbb{R}_+)$ or $w_L \notin (\mathbb{R}_- \cup \mathbb{R}_+)$. A brute-force verification was successfully conducted, in this cases, to prove that $y_L \boxslash_{r_l} w_L = x_l^+$. For the cases in which $y_L \in (\mathbb{R}_- \cup \mathbb{R}_+)$ and $w_L \in (\mathbb{R}_- \cup \mathbb{R}_+)$ we have, by definition of dd_l of Figure 6, that $x_l^+ = y_L \boxslash_{r_l} w_L$. Remember that, by Proposition 1, there exists $r \in S$ such that $y_L \boxslash_{\mathrm{r_l}} w_L = y_L \boxslash_{\mathrm{r}} w_L$. Since $y_L \in Y$ and $w_L \in Z$, we can conclude that $x_l^+ \notin X''$ implies that $y_L \boxslash_{\mathrm{r}} w_L \notin X''$, for any $X'' \subseteq X'$. An analogous reasoning applies to x_l^-, to x_u^+ and x_u^-. This allows us to prove the optimality claim.

Proof (of Theorem 4). Given the constraint $x = y \boxslash_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 4 computes a new, refining interval Y' for variable y. It returns either $Y' := (Y \cap [y_l^-, y_u^-]) \uplus (Y \cap [y_l^+, y_u^+])$ or $Y' = \varnothing$: hence, in both cases, we are sure that $Y' \subseteq Y$.

By Proposition 1, we can focus on finding a lower bound for $y \in Y$ by exploiting the constraint $y \boxslash_{\bar{r}_l} z = x$ and an upper bound for y by exploiting the constraint $y \boxslash_{\bar{r}_u} z = x$.

In order to compute correct bounds for y, Algorithm 5 first splits the interval of z into the sign-homogeneous intervals Z_- and Z_+, since knowing the sign of z is crucial to determine correct bounds for y. Hence, for $W = Z_-$ (and, analogously, for $W = Z_+$), it calls function σ of Figure 7 to determine the appropriate extrema of intervals X and W to be used to compute the new lower and upper bounds for y. As we did in the proof of Theorem 9, it is easy to verify that x_L and w_L (resp., x_U and w_U), computed using function σ of Figure 7, are the boundaries of X and W upon which y touches its minimum (resp., maximum). Functions id_l^f of Figure 8 and id_u^f of Figure 9 are then used to find the new bounds for y. The so obtained intervals for y will be eventually joined using convex union to obtain the refining interval for y.

We will now prove the non-trivial parts of the definitions of functions id_l^f and id_u^f. Concerning the case analysis of id_l^f (Fig 8) marked as a_4, the result changes depending on the selected rounding mode:

$\bar{r}_l = \uparrow$: we clearly must have $y = +\infty$, according to the IEEE 754 Standard [IEE08];

$\bar{r}_l = \downarrow$: it must be $y/w_L < -f_{\max}$ and thus, since w_L is negative, $y > -f_{\max} \cdot w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(-f_{\max} \boxdot_{\downarrow} w_L)$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\max})$, for $w_L = -\infty$ we need y to be greater than or equal to $\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L$. If $[\![\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]\!]_\uparrow = [\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]_\uparrow$, by (23) of Proposition 4, we can conclude $y \succcurlyeq [\![\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]\!]_\uparrow$. On the other hand, if $[\![\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]\!]_\uparrow \neq [\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]_\uparrow$, then we can only apply (19) of Proposition 4, obtaining $y \succcurlyeq [\![\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big) \cdot w_L]\!]_\downarrow$.

The case analysis of id_l^f (Fig 8) marked as a_5 can be explained as follows:

$\bar{r}_l = \downarrow$: we must have $y = +\infty$, according to the IEEE 754 Standard [IEE08];

$\bar{r}_l = \uparrow$: inequality $y/w_L > f_{\max}$ must hold and thus, since w_L is positive, $y > f_{\max} \cdot w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(f_{\max} \boxdot_{\downarrow} w_L)$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\max})$, for $x_L = +\infty$ we need y to be greater than or equal to $\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L$. If $[\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]\!]_\uparrow = [\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]_\uparrow$, by (23) of Proposition 4, we can conclude $y \succcurlyeq [\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]\!]_\uparrow$. On the other hand, if $[\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]\!]_\uparrow \neq [\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]_\uparrow$ then, we can only apply (19) of Proposition 4, obtaining $y \succcurlyeq [\![\big(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2\big) \cdot w_L]\!]_\downarrow$.

The explanation for the case analysis of id_l^f (Fig 8) marked as a_6 is the following:

$\bar{r}_l = \uparrow$: the lowest value of y that yields $x_L = +0$ with $w_L \in \mathbb{R}_-$ is clearly $y = -0$;

$\bar{r}_l = \downarrow$: inequality $y/w_L < f_{\min}$ should hold and thus, since w_L is negative, $y > f_{\min} \cdot w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(f_{\min} \boxdot_{\downarrow} w_L)$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\min})$, for $x_L = +0$ we need y to be greater than or equal to $(f_{\min} \cdot w_L)/2$. Since in this case $[\![(f_{\min} \cdot w_L)/2]\!]_{\uparrow} = [(f_{\min} \cdot w_L)/2]_{\uparrow} = (f_{\min} \boxdot_{\uparrow} w_L)/2$, by (23) of Proposition 4, we can conclude $y \succcurlyeq (f_{\min} \boxdot_{\uparrow} w_L)/2$.

Concerning the case analysis of id_l^f (Fig 8) marked as a_7, we must distinguish between the following cases:

$\bar{r}_l = \downarrow$: considering $x_L = -0$ and $w_L \in \mathbb{R}_+$, we clearly must have $y = -0$;

$\bar{r}_l = \uparrow$: it should be $y/w_L > -f_{\min}$ and thus, since w_L is positive, $y > -f_{\min} \cdot w_L$ and, by (20) of Proposition 4, $y \succcurlyeq \mathrm{succ}(-f_{\min} \boxdot_{\downarrow} w_L)$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\min})$, for $x_L = -0$ we need y be to greater than or equal to $\big(-f_{\min} \cdot w_L\big)/2$. Since in this case $[\![\big(-f_{\min} \cdot w_L\big)/2]\!]_{\uparrow} = [\big(-f_{\min} \cdot w_L\big)/2]_{\uparrow} = (-f_{\min} \boxdot_{\uparrow} w_L)/2$, by (23) of Proposition 4, we can conclude $y \succcurlyeq (-f_{\min} \boxdot_{\uparrow} w_L)/2$.

Similar arguments can be used to prove the case analyses of id_u^f of Fig 9 marked as a_9, a_{10}, a_{11} and a_{12}.

We will now analyze the case analyses of id_l^f of Fig 8 marked as a_3^- and a_3^+, and the ones of id_u^f of Fig 9 marked as a_8^- and a_8^+. We can assume, of course, $X = [x_l, x_u]$, $Y = [y_l, y_u]$ and $Z = [w_l, w_u]$, where $x_l, x_u, w_l, w_u \in \mathbb{F} \cap \mathbb{R}$, $x_l \leq x_u$, $w_l \leq w_u$ and $\mathrm{sgn}(w_l) = \mathrm{sgn}(w_u)$. Exploiting $x \preccurlyeq y \boxslash z$ and $x \succcurlyeq y \boxslash z$, by Proposition 3, we have

$$y/z \begin{cases} \geq x, & \text{if } \bar{r}_l = \downarrow; \\ > x + \nabla^{\uparrow}(x) = \mathrm{pred}(x), & \text{if } \bar{r}_l = \uparrow; \\ \geq x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > x + \nabla_2^{\mathrm{n}-}(x)/2, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{58}$$

$$y/z \begin{cases} < x + \nabla^{\downarrow}(x) = \mathrm{succ}(x), & \text{if } \bar{r}_u = \downarrow; \\ \leq x, & \text{if } \bar{r}_u = \uparrow; \\ \leq x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < x + \nabla_2^{\mathrm{n}+}(x)/2, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{59}$$

Since the case $z = 0$ is handled separately by id_l^f of Fig 8 and by id_u^f of Fig 9, we can assume $z \neq 0$. Thanks to the split of Z into a positive and a negative part, the sign of z is determinate. In the following, we will prove the case analyses marked as a_3^+ and a_8^+, hence assuming $z > 0$. From the previous case analysis

we can derive

$$y \begin{cases} \geq x \cdot z, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x) \cdot z, & \text{if } \bar{r}_l = \uparrow \\ \geq \big(x + \nabla_2^{\mathrm{n}-}(x)/2\big) \cdot z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > \big(x + \nabla_2^{\mathrm{n}-}(x)/2\big) \cdot z, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x); \end{cases} \tag{60}$$

$$y \begin{cases} < \mathrm{succ}(x) \cdot z, & \text{if } \bar{r}_u = \downarrow; \\ \leq x \cdot z, & \text{if } \bar{r}_u = \uparrow; \\ \leq \big(x + \nabla_2^{\mathrm{n}+}(x)/2\big) \cdot z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < \big(x + \nabla_2^{\mathrm{n}+}(x)/2\big) \cdot z, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{61}$$

Note that the members of the product are independent. Therefore, we can find the minimum of the product by minimizing each member of the product. Since we are analyzing the case in which $W = Z_+$, let (x_L, x_U, w_L, w_U) as defined in function σ of Figure 7, replacing the role of y with z and the role of z with x. Hence, by Proposition 2 and the monotonicity of 'pred' and 'succ' we obtain

$$y \begin{cases} \geq x_L \cdot w_L, & \text{if } \bar{r}_l = \downarrow; \\ > \mathrm{pred}(x_L) \cdot w_L, & \text{if } \bar{r}_l = \uparrow \\ \geq \big(x_L + \nabla_2^{\mathrm{n}-}(x_L)/2\big) \cdot w_L, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{even}(x); \\ > \big(x_L + \nabla_2^{\mathrm{n}-}(x_L)/2\big) \cdot w_L, & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \mathrm{odd}(x); \end{cases} \tag{62}$$

$$y \begin{cases} < \mathrm{succ}(x_U) \cdot w_U, & \text{if } \bar{r}_u = \downarrow; \\ \leq x_U \cdot w_U, & \text{if } \bar{r}_u = \uparrow; \\ \leq \big(x_U + \nabla_2^{\mathrm{n}+}(x_U)/2\big) \cdot w_U, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{even}(x); \\ < \big(x_U + \nabla_2^{\mathrm{n}+}(x_U)/2\big) \cdot w_U, & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \mathrm{odd}(x). \end{cases} \tag{63}$$

We can now exploit Proposition 4 and obtain:

$$y_l' \stackrel{\mathrm{def}}{=} \begin{cases} x_L \boxdot_\uparrow w_L, & \text{if } \bar{r}_l = \downarrow; \\ \mathrm{succ}\big(\mathrm{pred}(x_L) \boxdot_\downarrow w_L\big), & \text{if } \bar{r}_l = \uparrow; \end{cases} \tag{64}$$

$$y_u' \stackrel{\mathrm{def}}{=} \begin{cases} \mathrm{pred}\big(\mathrm{succ}(x_U) \boxdot_\uparrow w_U\big), & \text{if } \bar{r}_u = \downarrow; \\ x_U \boxdot_\downarrow w_U, & \text{if } \bar{r}_u = \uparrow. \end{cases} \tag{65}$$

Indeed, if $\bar{r}_l = \uparrow$ and $x_L \neq 0$, then part (23) of Proposition 4 applies and we have $y \succcurlyeq x_L \boxdot_\uparrow w_L$. On the other hand, if $x_L = 0$, since by hypothesis $z > 0$ implies $w_L > 0$, according to IEEE 754 [IEE08, Section 6.3], we have $x_L \boxdot_\uparrow w_L = \mathrm{sgn}(x_L) \cdot 0$ and, indeed, for each non-NaN, nonzero and finite $w \in \mathbb{F} \cap [+0, +\infty]$, $\mathrm{sgn}(x_L) \cdot 0$ is the least value for y that satisfies $\mathrm{sgn}(x_L) \cdot 0 = y \boxslash_\downarrow w$.

Analogously, if $\bar{r}_l = \uparrow$ and $x_L \neq f_{\min}$, then Proposition 4 applies and we have $\mathrm{succ}\big(\mathrm{pred}(x_L) \boxdot_\downarrow w_L\big)$. On the other hand, if $x_L = f_{\min}$, in this case, $\mathrm{succ}\big(\mathrm{pred}(x_L) \boxdot_\downarrow w_L\big) = f_{\min}$ which is consistent with the fact that, for each non-NaN, nonzero and finite $w \in \mathbb{F} \cap [+0, +\infty]$, $f_{\min}$ is the lowest value for y that satisfies $f_{\min} = y \boxslash_\uparrow w$.

A symmetric argument justifies (65).

As before, we need to approximate the values of the expressions $e_l^+ = \big(x_L + \nabla_2^{\mathrm{n}-}(x_L)/2\big)\cdot w_L$ and $e_u^+ = \big(x_U + \nabla_2^{\mathrm{n}+}(x_U)/2\big)\cdot w_U$. We leave this as an implementation choice, thus taking into account the case $[\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow$ and $[\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow$ as well as $[\![e_l^+]\!]_\uparrow > [e_l^+]_\uparrow$ and $[\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow$. Therefore, when $[\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow$ by (63) and (21) of Proposition 4 we obtain $y \preccurlyeq [\![e_u^+]\!]_\uparrow$, while, when $[\![e_l^+]\!]_\downarrow > [e_l^+]_\downarrow$ by (63) and (19) of Proposition 4 we obtain $y \succcurlyeq [\![e_l^+]\!]_\downarrow$.

Thus, for the case in which $\bar{r}_l = \mathrm{n}$, since $e_u^+ \neq 0$ and $e_l^+ \neq 0$, by Proposition 4, we have

$$y_l' \stackrel{\text{def}}{=} \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if even}(x_L) \text{ and } [\![e_l^+]\!]_\uparrow \neq [e_l^+]_\uparrow; \\ \mathrm{succ}\big([\![e_l^+]\!]_\downarrow\big), & \text{otherwise;} \end{cases} \tag{66}$$

whereas, for the case in which $\bar{r}_u = \mathrm{n}$,

$$y_u' \stackrel{\text{def}}{=} \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if even}(x_U) \text{ and } [\![e_u^+]\!]_\downarrow \neq [e_u^+]_\downarrow; \\ \mathrm{pred}\big([\![e_u^+]\!]_\uparrow\big), & \text{otherwise.} \end{cases} \tag{67}$$

An analogous reasoning, but with $z < 0$, allows us to obtain the case analyses marked as a_3^- and a_8^-.

Proof (of Theorem 5). Given the constraint $x = y \boxslash_S z$ with $x \in X = [x_l, x_u]$, $y \in Y = [y_l, y_u]$ and $z \in Z = [z_l, z_u]$, Algorithm 5 finds a new, refined interval Z' for variable z.

Since it assigns either $Z' := (Z \cap [z_l^-, z_u^-]) \uplus (Z \cap [z_l^+, z_u^+])$ or $Z' = \varnothing$, in both cases we are sure that $Z' \subseteq Z$. By Proposition 1, as in the previous proofs, we can focus on finding a lower bound for $z \in Z$ by exploiting the constraint $y \boxslash_{\bar{r}_l} z = x$ and an upper bound for z by exploiting the constraint $y \boxslash_{\bar{r}_u} z = x$.

We first need to split interval X into the sign-homogeneous intervals X_- and X_+, because knowing the sign of x is crucial for determining correct bounds for z. Hence, for $V = X_-$ (and, analogously, for $V = X_+$) function τ of Figure 5 determines the appropriate interval extrema of Y and V to be used to compute the new lower and upper bounds for z. As in the previous proofs (see, for example, proof of Theorem 10), it is easy to verify that y_L and v_L (resp., y_U and v_U) computed using function τ of Figure 5 are the boundaries of Y and V upon which z touches its minimum (resp., maximum). Functions id_l^s of Figure 10 and id_u^s of Figure 11 are then used to find the new bounds for z. The so obtained intervals for z will be then joined with convex union in order to obtain the refining interval for z.

We will prove the most important parts of the definitions of id_l^s (Fig. 10) and id_u^s (Fig. 11) only, starting with the case analysis marked as a_4. Depending on the rounding mode in effect, the following arguments are given:

$\bar{r}_l = \downarrow$: in this case, the only possible way to obtain -0 as the result of the division is having $z = +\infty$ (with $y \in \mathbb{R}_-$);

$\bar{r}_l = \uparrow$: it should be $y_L/z > -f_{\min}$ and thus, since y_L and x_L are negative, we can conclude that z is positive. Thus, $y_L > -f_{\min} \cdot z$ implies $y_L/-f_{\min} < z$, and by (20) of Proposition 4, $z \succcurlyeq \mathrm{succ}(z_L \boxslash_{\downarrow} -f_{\min})$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(-f_{\min})$, for $v_L = -0$ we need $y_L/z \geq (-f_{\min} + \nabla_2^{\mathrm{n}+}(-f_{\min})/2) = (-f_{\min} + f_{\min}/2) = -f_{\min}/2$. As before, since y_L and v_L are negative, we can conclude that z is positive: hence $y_L \geq (-f_{\min}/2) \cdot z$. Therefore, $z \geq y_L/(-f_{\min}/2) = z \geq (y_L/-f_{\min}) \cdot 2$. Since in this case $[\![(y_L/-f_{\min}) \cdot 2]\!]_{\uparrow} = [(y_L/-f_{\min}) \cdot 2]_{\uparrow} = (y_L \boxslash_{\uparrow} -f_{\min}) \cdot 2$, by (23) of Proposition 4, we can conclude $y \succcurlyeq (y_L \boxslash_{\uparrow} -f_{\min}) \cdot 2$.

As for the case analysis of id_l^s (Fig. 10) marked as a_5, we must distinguish between the following cases:

$\bar{r}_l = \uparrow$: we must have $z = +\infty$ in order to obtain $x = +0$;

$\bar{r}_l = \downarrow$: inequality $y_L/z < f_{\min}$ must hold and thus, since positive y_L and v_L imply a positive z, $z > y_L/f_{\min}$ and, by (20) of Proposition 4, $z \succcurlyeq \mathrm{succ}(y_L \boxslash_{\downarrow} f_{\min})$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\min})$, for $v_L = +0$ we need $y_L/z \leq f_{\min}/2$. As z is positive in this case, $(y_L/f_{\min}) \cdot 2 \leq z$. Since $[\![(y_L/f_{\min}) \cdot 2]\!]_{\uparrow} = [(y_L/f_{\min}) \cdot 2]_{\uparrow} = (y_L \boxslash_{\uparrow} f_{\min}) \cdot 2$, by (23) of Proposition 4, we can conclude $y \succcurlyeq (y_L \boxslash_{\uparrow} f_{\min}) \cdot 2$.

Concerning the case analysis of id_l^s (Fig 10) marked as a_6, we must distinguish between the following cases:

$\bar{r}_l = \downarrow$: the lowest value of z that gives $x = +\infty$ with $y \in \mathbb{R}_-$ is $z = -0$;

$\bar{r}_l = \uparrow$: inequality $y_L/z > f_{\max}$ must hold; since y_L is negative and v_L is positive, z must be negative, and therefore $y_L < f_{\max} \cdot z$. Hence, $y_L/f_{\max} < z$. By (20) of Proposition 4, we obtain $z \succcurlyeq \mathrm{succ}(y_L \boxslash_{\downarrow} f_{\max})$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(f_{\max})$, for $v_L = +\infty$ we need $y_L/z \geq (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)$. As before, since w_L is negative and v_L is positive, we can conclude that z is negative, and, therefore, $y_L \leq (f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2) \cdot z$ holds. As a consequence, $y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2) \leq z$. If $[\![y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]\!]_{\uparrow} = [y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]_{\uparrow}$, by (23) of Proposition 4, we can conclude $z \succcurlyeq [\![y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]\!]_{\uparrow}$. On the other hand, if $[\![y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]\!]_{\uparrow} \neq [y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]_{\uparrow}$ then, we can only apply (19) of Proposition 4, obtaining $z \succcurlyeq [\![y_L/(f_{\max} + \nabla_2^{\mathrm{n}+}(f_{\max})/2)]\!]_{\downarrow}$.

Regarding the case analysis of id_l^s (Fig 10) marked as a_7, we have the following cases:

$\bar{r}_l = \uparrow$: the lowest value of z that yields $x = -\infty$ with $y \in \mathbb{R}_+$ is $z = -0$;

$\bar{r}_l = \downarrow$: inequality $y_L/z < -f_{\max}$ must hold and thus, since a positive y_L and a negative v_L imply that the sign of z is negative, $y_L > -f_{\max} \cdot z$. Hence, $y_L/-f_{\max} < z$. By (20) of Proposition 4, $z \succcurlyeq \mathrm{succ}(y_L \boxslash_{\downarrow} -f_{\max})$.

$\bar{r}_l = \mathrm{n}$: since $\mathrm{odd}(-f_{\max})$, for $v_L = -\infty$ we need $y_L/z \leq -f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2$. Since z in this case is negative, we obtain the inequality $z \geq y_L/(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2)$. If $[\![y_L/\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)]\!]_{\uparrow} = [y_L/\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)]_{\uparrow}$, by (23) of Proposition 4, we can conclude $y \succcurlyeq [\![y_L/\big(-f_{\max} + \nabla_2^{\mathrm{n}-}(-f_{\max})/2\big)]\!]_{\uparrow}$.

On the other hand, if $[\![y_L/(-f_{\max} + \nabla_2^{n-}(-f_{\max})/2)]\!]_\uparrow \neq [y_L/(-f_{\max} + \nabla_2^{n-}(-f_{\max})/2)]_\uparrow$, then we can only apply (19) of Proposition 4, obtaining $y \succcurlyeq [\![y_L/(-f_{\max} + \nabla_2^{n-}(-f_{\max})/2)]\!]_\downarrow$.

Similar arguments can be used to prove the case analyses of function id_u^s of Fig. 11 marked as a_9, a_{10}, a_{11} and a_{12}.

We will now analyze the case analyses of id_l^s of Fig. 10 marked as a_3^- and a_3^+, and the ones of id_u^s of Fig. 9 marked as a_8^- and a_8^+. In this proof, we can assume $y_L, v_L \in \mathbb{R}_- \cup \mathbb{R}_+$, $y_U, v_U \in \mathbb{R}_- \cup \mathbb{R}_+$ and $\mathrm{sgn}(v_L) = \mathrm{sgn}(v_U)$. First, note that the argument that leads to (60) and (61) starting from $x \preccurlyeq y \boxtimes z$ and $x \succcurlyeq y \boxtimes z$ is in common with the proof of Theorem 4.

Provided that interval X is split into intervals X_+ and X_-, it is worth discussing the reasons why it is not necessary to partition also Y directly in Algorithm 5. Assume $Y = [-a, b]$ with $a, b > 0$ and consider the partition of Y into two sign-homogeneus intervals $Y \cap [-\infty, -0]$ and $Y \cap [+0, +\infty]$, as usual. Note that the values $-0 \in Y \cap [-\infty, -0] = [-a, -0]$ and the values $+0 \in Y \cap [+0, +\infty] = [+0, b]$ can never be the boundaries of Y upon which z touches its minimum (resp., maximum). This is because y will be the numerator of fractions (see expressions (68) and (69)). Moreover, by the definition of functions id_l^s of Fig 10 and id_u^s of Fig 11, it easy to verify that the partition of Y would not prevent the interval computed for y from being equal to the empty set. That is, if $\mathrm{id}_l^s(y_L, v_L, \bar{r}_l) = \text{unsat.}$ or $\mathrm{id}_u^s(y_U, v_U, \bar{r}_u) = \text{unsat.}$, then partitioning also Y into sign-homogeneus intervals and then applying the procedure of Algorithm 5 to the two distinct intervals results again into an empty refining interval for z.

Hence, to improve efficiency, Algorithm 5 does not split interval Y into sign-homogeneous intervals. However, in this proof it is necessary to partition Y into intervals Y_- and Y_+ in order to determine the correct formulas for lower and upper bounds for z. In the following, for the sake of simplicity, we will analyze the special case X_+ and $Y = Y_+$, so that Y does not need to be split because it is already a sign-homogeneous interval. The remaining cases in which Y is sign-homogeneous as well as those in which it is not can be derived analogously. To sum up, in this case we assume $x \geq 0$ and $y \geq 0$, and therefore $z > 0$.

Now, we need to prove the cases marked as a_3^+ and a_8^+. The case analysis of (58) and (59) yields (60) and (61). Remember that the case $x = \pm 0$ is handled separately by functions id_l^s of Fig. 10 and id_u^s of Fig. 11, hence assuming $x > 0$,

we obtain

$$z \begin{cases} \leq y/x, & \text{if } \bar{r}_u = \downarrow; \\ < y/\operatorname{pred}(x), & \text{if } \bar{r}_u = \uparrow \text{ and } x \neq f_{\min}; \\ \leq f_{\max}, & \text{if } \bar{r}_u = \uparrow \text{ and } x = f_{\min}; \\ \leq y/\big(x + \nabla_2^{\mathrm{n}-}(x)/2\big), & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \operatorname{even}(x); \\ < y/\big(x + \nabla_2^{\mathrm{n}-}(x)/2\big), & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \operatorname{odd}(x); \end{cases} \tag{68}$$

$$z \begin{cases} > y/\operatorname{succ}(x), & \text{if } \bar{r}_l = \downarrow \text{ and } x \neq -f_{\min}; \\ \geq -f_{\max}, & \text{if } \bar{r}_l = \downarrow \text{ and } x = -f_{\min}; \\ \geq y/x, & \text{if } \bar{r}_l = \uparrow; \\ \geq y/\big(x + \nabla_2^{\mathrm{n}+}(x)/2\big), & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \operatorname{even}(x); \\ > y/\big(x + \nabla_2^{\mathrm{n}+}(x)/2\big), & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \operatorname{odd}(x). \end{cases} \tag{69}$$

Since the members of the divisions are independent, we can find the minimum of said divisions by minimizing each one of their members. Let (y_L, y_U, v_L, v_U) be as returned by function τ of Figure 5. By Proposition 2 and the monotonicity of 'pred' and 'succ' we obtain

$$z \begin{cases} \leq y_U/v_U, & \text{if } \bar{r}_u = \downarrow; \\ < y_U/\operatorname{pred}(v_U), & \text{if } \bar{r}_u = \uparrow \text{ and } v_U \neq f_{\min}; \\ \leq f_{\max}, & \text{if } \bar{r}_u = \uparrow \text{ and } v_U = f_{\min}; \\ \leq y_U/\big(v_U + \nabla_2^{\mathrm{n}-}(v_U)/2\big), & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \operatorname{even}(v_U); \\ < y_U/\big(v_U + \nabla_2^{\mathrm{n}-}(v_U)/2\big), & \text{if } \bar{r}_u = \mathrm{n} \text{ and } \operatorname{odd}(v_U); \end{cases} \tag{70}$$

$$z \begin{cases} > y_L/\operatorname{succ}(v_L), & \text{if } \bar{r}_l = \downarrow \text{ and } v_L \neq -f_{\min}; \\ \geq -f_{\max}, & \text{if } \bar{r}_l = \downarrow \text{ and } v_L = -f_{\min}; \\ \geq y_L/v_L, & \text{if } \bar{r}_l = \uparrow; \\ \geq y_L/\big(v_L + \nabla_2^{\mathrm{n}+}(v_L)/2\big), & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \operatorname{even}(v_L); \\ > y_L/\big(v_L + \nabla_2^{\mathrm{n}+}(v_L)/2\big), & \text{if } \bar{r}_l = \mathrm{n} \text{ and } \operatorname{odd}(v_L). \end{cases} \tag{71}$$

We can now exploit Proposition 4 and obtain:

$$z'_l \stackrel{\text{def}}{=} \begin{cases} y_L \boxslash_{\uparrow} v_L, & \text{if } \bar{r}_l = \uparrow; \\ \operatorname{succ}\big(y_L \boxslash_{\downarrow} \operatorname{succ}(v_L)\big), & \text{if } \bar{r}_l = \downarrow \text{ and } v_L \neq -f_{\min}; \end{cases} \tag{72}$$

$$z'_u \stackrel{\text{def}}{=} \begin{cases} \operatorname{pred}\big(y_U \boxslash_{\uparrow} \operatorname{pred}(v_U)\big), & \text{if } \bar{r}_u = \uparrow \text{ and } v_U \neq f_{\min}; \\ y_U \boxslash_{\downarrow} v_U, & \text{if } \bar{r}_u = \downarrow. \end{cases} \tag{73}$$

Since $y_L \neq 0$, then $y_L/\operatorname{succ}(v_L) \neq 0$. Hence, Proposition 4 applies and we have $z \succcurlyeq y_L \boxslash_{\uparrow} v_L$ if $\bar{r}_l = \uparrow$ and $z \succcurlyeq \operatorname{succ}\big(y_L/\operatorname{succ}(v_L)\big)$ if $\bar{r}_l = \downarrow$ and $v_L \neq -f_{\min}$. Analogously, since $y_U \neq 0$, then $y_U/\operatorname{pred}(v_L) \neq 0$. Hence, by Proposition 4 we obtain (73).

Note that, since division by zero is not defined on real numbers, we had to separately address the case $\bar{r}_u = \uparrow$ and $x = f_{\min}$ in (68), and the case $\bar{r}_l = \downarrow$ and

$x = -f_{\min}$ in (69). Division by zero is, however, defined on IEEE 754 floating-point numbers. Indeed, if we evaluate the second case of (72) with $v_L = -f_{\min}$, we obtain $\mathrm{succ}(y_L \boxslash_\downarrow \mathrm{succ}(-f_{\min})) = -f_{\max}$, which happens to be the correct value for z'_l, provided $y_L > 0$. The same happens for (73). Therefore, there is no need for a separate treatment when variable x takes the values $\pm f_{\min}$.

As before, we need to approximate the values of the expressions $e_u^+ \stackrel{\text{def}}{=} y_U/\big(v_U + \nabla_2^{\mathrm{n}-}(v_U)/2\big)$ and $e_l^+ \stackrel{\text{def}}{=} y_L/\big(v_L + \nabla_2^{\mathrm{n}+}(v_L)/2\big)$. Thus, when $[\![e_u^+]\!]_\downarrow < [e_u^+]_\downarrow$ by (63) and (21) of Proposition 4 we obtain $y \preccurlyeq [\![e_u^+]\!]_\uparrow$, while, when $[\![e_l^+]\!]_\downarrow > [e_l^+]_\downarrow$ by (63) and (19) of Proposition 4 we obtain $y \succcurlyeq [\![e_l^+]\!]_\downarrow$. Thus, for the case where $\bar{r}_l = \mathrm{n}$, since $e_u^+ \neq 0$ and $e_l^+ \neq 0$, by Proposition 4, we have

$$y'_l \stackrel{\text{def}}{=} \begin{cases} [\![e_l^+]\!]_\uparrow, & \text{if even}(v_L) \text{ and } [\![e_l^+]\!]_\uparrow = [e_l^+]_\uparrow; \\ [\![e_l^+]\!]_\downarrow, & \text{if even}(v_L) \text{ and } [\![e_l^+]\!]_\uparrow \neq [e_l^+]_\uparrow; \\ \mathrm{succ}\big([\![e_l^+]\!]_\downarrow\big), & \text{otherwise;} \end{cases} \tag{74}$$

whereas, for the case in which $\bar{r}_u = \mathrm{n}$,

$$y'_u \stackrel{\text{def}}{=} \begin{cases} [\![e_u^+]\!]_\downarrow, & \text{if even}(v_U) \text{ and } [\![e_u^+]\!]_\downarrow = [e_u^+]_\downarrow; \\ [\![e_u^+]\!]_\uparrow, & \text{if even}(v_U) \text{ and } [\![e_u^+]\!]_\downarrow \neq [e_u^+]_\downarrow; \\ \mathrm{pred}\big([\![e_u^+]\!]_\uparrow\big), & \text{otherwise.} \end{cases} \tag{75}$$

An analogous reasoning allows us to prove the case analyses marked as a_3^- and a_8^-.