Representable correcting terms for possibly underflowing floating point operations

Sylvie Boldo & Marc Daumas
E-mail: Sylvie.Boldo@ENS-Lyon.Fr & Marc.Daumas@ENS-Lyon.Fr
Laboratoire de l'Informatique du Parallélisme
UMR 5668 CNRS – ENS de Lyon – INRIA
Lyon, France

Abstract

Studying floating point arithmetic, authors have shown that the implemented operations (addition, subtraction, multiplication, division and square root) can compute a result and an exact correcting term using the same format as the inputs. Following a path initiated in 1965, many authors supposed that neither underflow nor overflow occurred in the process. Overflow is not critical as this kind of exception creates persisting non numeric quantities. Underflow may be fatal to the process as it returns wrong numeric values with little warning. Our new conditions guarantee that the correcting term is exact when the result is a number. We have validated our proofs against Coq automatic proof checker. Our development has raised many questions, some of them were expected while other ones were surprising.

1 Introduction

It was recognized in 1991 [2] that the most widely used algebraic operations of floating point arithmetic can return an exact correcting term provided the same floating point format is used for the inputs, the result and the correcting term. First results on a similar subject were probably presented in 1965 [11, 16] as techniques to enhance precision for additions and accumulations.

As the IEEE 754 and 854 standards [21, 4] spread the use of correct rounding modes including rounding to the nearest floating point value, rationales were studied for the four mentioned operations leading to a generic theorem such as the one below.

Result 1 (Adapted from Bohlender *et al*, 1991) *Let $\mathbb{F}$ be the set of real numbers represented with the defined floating point arithmetic and let $\oplus$, $\ominus$, $\otimes$, $\oslash$, $\circ(\sqrt{\cdot})$ be the implemented addition, subtraction, multiplication, division and square root rounded to the nearest floating point value. Provided neither underflow nor overflow precludes a dependable behavior, an exact correcting term that is known to belong to $\mathbb{F}$ can be defined as follows from inputs x and y and the result,*

Result	*Correcting term*
$x \oplus y$	$x + y - x \oplus y$
$x \ominus y$	$x - y - x \ominus y$
$x \otimes y$	$x \times y - x \otimes y$
$x \oslash y$	$x - (x \oslash y) \times y$
$\circ(\sqrt{x})$	$x - (\circ(\sqrt{x}))^2$

The correcting term is still in $\mathbb{F}$ if we use a directed rounding mode for the multiplication $\otimes$ and the division $\oslash$.

The fact that many authors have overlooked consequences of an overflow or an underflow in their related work [6, 13, 2, 19], may have arisen from an inadequacy of the existing formalisms to build a tight condition for underflow to be harmless. We will see in this text that an error may occur even when neither inputs nor the result are subnormal numbers.

A sufficient condition to be able to define an exact correcting term is that the exact error lies above the gradual underflow threshold as we will see in the conclusion of this text. This is not a necessary condition. We will present examples and counter examples in the text to show that all our conditions are both sufficient and tight.

In the process of giving tight conditions for Result 1 to be correct even when underflow occurs, we have isolated a very surprising situation. In an intuitive deduction, the rounded result r is the most significant part of the exact result and the correcting term e is a remainder. It is easy to jump to a conclusion that

$|e|$ must be significantly smaller than $|r|$ in additions, subtractions and multiplications, and smaller than $|x|$ in divisions and square roots. We have exhibited cases were $|e|$ is close to $|x|$. Exploring the directed rounding modes, we have found cases where $|e|$ is significantly larger than $|x|$.

All theorems presented in this text have been developed with a strong focus on genericity. The radix, the number of significant digits of the mantissa, the underflow threshold and the rounding mode used are parameters possibly set in the hypotheses. Jumping to the conclusions of this text, we have no restriction on the radix provided it is an integer greater than or equal to 2 and no restriction on the number of digits of the mantissa provided once again it is greater than or equal to 2. The tie breaking mechanism when rounding to the nearest has no effect on our theorems and can be even, odd or use any combination. However, precise rounding to a nearest floating point number is necessary for additions and square roots.

Following Section 2, discussing our formalism and rounding, we present results with increasing difficulty on multiplications (Section 3), additions and subtractions (Section 4), divisions (Section 5), and square roots (Section 6). As our question is easily connected to the correct behavior of the remainder (FPREM) operation defined by the IEEE standard, we have built a machine-checked proof of this fact in the last section of this text, answering a question raised in 1995 [15] (Section 7). All the proofs can be downloaded through the Internet at the following address.

`http://www.ens-lyon.fr/~sboldo/coq/`

2 Floating point representation

Numbers are represented with pairs (n, e) that stand for $n \times \beta^e$ where β is the radix of the floating point system. We use both an integral signed mantissa n and an integral signed exponent e for sake of simplicity. As we use integral mantissa and exponent, components have to be slightly adapted to get the usual IEEE standard **exponent** and the usual IEEE standard **mantissa** stored in computers.

The underflow exponent, a constant, is the lowest exponent $-e_{min}$ available. We do not set an upper bound on the exponent as overflows are easily detected by other means. Mantissas are stored in signed-magnitude format using p digits. We define a **representable** pair (n, e) such that $|n| < \beta^p$ and $e \geq -e_{min}$.

The above definition is not sufficient to identify one unique pair (n, e) for a represented quantity. A representable pair is **normal** if $\beta \cdot |n| \geq \beta^p$ and it is **subnormal** if $\beta \cdot |n| < \beta^p$ and $e = -e_{min}$. Each represented number has one unique representation, the **machine** pair, that is either normal or subnormal.

In some sense, our internal representation is similar to the one proposed in the late 1950s [1]. However, our approach is very different as computers only manipulate unique machine representations. This formalism is used to model the numbers but not to compute on them. Additional representations are used to state conditions and prove theorems.

In all the following theorems, we use any representation. We do not have to use the machine representation. Our theorems are valid with subnormals but no example use them explicitly.

Our formalism was introduced in [5] for developments using Coq proof environment [9]. Other formalisms of floating point arithmetic are in use with PVS [15, 10], HOL [3, 8] or ACL2 [18]. Using Curry Howard isomorphism, Coq and HOL rely on a very small inference engine to check the correctness of proofs. Although Coq and HOL lack many automatic techniques implemented in PVS or ACL, they allow users to safely explore properties of a system.

Most available general purpose processors have long been compliant with the IEEE 754 standard on floating point arithmetic. It means that they implement precise rounding for the four arithmetic operations: addition, multiplication, division and square root. The result obtained for any of these computer operations is the one given by using a user-chosen rounding function on the result of the exact mathematical operation. The standard specifies four rounding functions: rounding to the nearest with the even tie breaking rule, rounding up, down or toward 0. The rounding functions and the arithmetic operators are defined and used in our formalism.

3 Multiplication

Concerning the question raised in this paper, the multiplication is most probably the simplest operation. This fact was recognized early and IBM S/370 has a special instruction to produce the rounded product and the exact correcting term [12].

An algorithm has long been developed and tested to compute these pairs with only IEEE standard operations [6, 13] although it relies on some developments on the addition presented Section 4. The task of producing the two quantities is much simpler with a computer that provides a full accuracy fused multiply and accumulate operator such as with Intel IA64 [14].

The following theorem is based on some early development of our project.

Theorem 2 (RepMult_gen in FroundMult) *Let $\otimes$ be the implemented multiplication rounded to a nearest floating point value or with a directed rounding mode. Given inputs x and y such that $x \otimes y$ is neither an infinity nor a NaN, the correcting term*

$$x \times y - x \otimes y$$

is representable if and only if there exist two representable pairs (n_x, e_x) and (n_y, e_y) representing x and y such that

$$e_x + e_y \geq -e_{min}.$$

Proof sketch adapted from [5]: The product of two p-digit mantissas produces a $2p$-digit mantissa that can be split into two p-digit floating point pairs whatever the rounding mode used.

The condition $e_x + e_y \geq -e_{min}$ is sufficient as it means that the extended mantissa is larger than the underflow threshold. It is also necessary. Suppose that there are no representable pairs for x and y such that $e_x + e_y \geq -e_{min}$, we conclude that $x \times y \bmod \beta^{-e_{min}}$ is different from zero. Such difference cannot be represented as it is below the underflow threshold.

□

Counter example against weaker conditions: We use the two following pairs representing the numbers $9 \times 2^{-\lfloor \frac{e_{min}}{2} \rfloor - 1}$ and $11 \times 2^{-\lceil \frac{e_{min}}{2} \rceil}$. Our notation uses $p = 4$ digits and an arbitrary value for e_{min}.

$$(1001_2, -\left\lfloor \frac{e_{min}}{2} \right\rfloor - 1)_2 \quad \text{and} \quad (1011_2, -\left\lceil \frac{e_{min}}{2} \right\rceil)_2$$

The exact product $99 \times 2^{-e_{min}-1}$ rounds to the nearest floating point pair $96 \times 2^{-e_{min}-1} = 12 \times 2^{-e_{min}+2}$ represented by the pair $(1100_2, -e_{min} + 2)_2$. The exact correcting term is $-3 \times 2^{-e_{min}-1}$ and cannot be represented. Still neither inputs nor the result is a subnormal pair.

4 Addition and subtraction

Authors have long exhibited two different situations in the production of an exact representable correcting term for additions and subtractions. If the exponents of the inputs are close enough, the exact result can be written with a $2p$-digit mantissa. In this case, the rounded value and the error can be stored with representable pairs whatever the rounding mode being either to a nearest or directed.

If the exponents of the inputs are too far away one from another, we have to make sure that the rounded result is the largest input in magnitude. This fact is obtained only when rounding to a nearest floating point value if the operations are precisely rounded. It was proved in some early part of our development [5] by adapting the proof of the correctness of the algorithm published in [19] to obtain the correcting term.

Theorem 3 (errorBoundedPlus in ClosestPlus) *Let $\oplus$ and $\ominus$ be the implemented addition and subtraction rounded to a nearest floating point value. Given inputs x and y, for each result $x \oplus y$ and $x \ominus y$ that is neither an infinity nor a NaN, the correcting terms*

$$x + y - x \oplus y \quad \text{and} \quad x - y - x \ominus y$$

are representable.

Counter example against weaker conditions: We present now an example where the double rounding of Texas Instruments' TMS 320 C3x or Intel's IA32 introduces an unrepresentable error. Let the radix be $\beta = 2$ and the precision $p > 4$ arbitrary large. We assume that the extended precision less than doubles the number of digits in the mantissa. We compute the sum of the two normalized numbers $(-(2^{p-1} + 1), p + 1)_2$ and $(2^p - 3, 0)_2$.

The first input has value $-2^{2p} - 2^{p+1}$ and the exact result is $-2^{2p} - 2^p - 3$. As the extended precision is limited, the last two bits of the result are lost and the first rounding returns $-2^{2p} - 2^p$. This result is next rounded to -2^{2p} and the error is $-2^p - 3$ that cannot be represented.

We obtain the same unrepresentable correcting term with the same inputs using the IEEE standard directed rounding mode to $+\infty$.

5 Division

Theorem 4 exhibits two conditions for the correcting term to be representable. The first condition (1) is expected. The second condition (2) is very new and deals with a situation that only occurs with some of the directed rounding modes.

Theorem 4 (errBoundedDiv in FroundDivSqrt.v) *Let $\oslash$ be the implemented division rounded to a nearest floating point value or with a directed rounding mode. Given inputs x and y, whenever $x \oslash y$ is neither an infinity nor a NaN, the correcting term*

$$x - (x \oslash y) \times y$$

is representable if and only if there exist two representable pairs (n_y, e_y) and (n_q, e_q) representing y and $x \oslash y$ such that

$$e_y + e_q \geq -e_{min}, \tag{1}$$

and

$$|x \oslash y| \neq \beta^{-e_{min}} \quad \textit{or} \quad \frac{\beta^{-e_{min}}}{2} \leq \left|\frac{x}{y}\right|. \tag{2}$$

Proof: We will prove that the exact remainder $r = x-(x\oslash y)\times y$ computed with appropriate care is representable so that the two conditions are sufficient. We first define $q = x \oslash y$ with any rounding mode. We assume that there exist some representations (n_q, e_q) of q and (n_y, e_y) of y that satisfy (1) and let (n_x, e_x) be a representation of x.

Let (n'_q, e'_q) be the machine representation of q, that means that

$$n_q \times \beta^{e_q} = n'_q \times \beta^{e'_q}$$

and we can define the **unit in the last place** function $\mathrm{ulp}(q) = \beta^{e'_q}$. We know from previous results that

$$e'_q \leq e_q \quad (\texttt{FcanonicLeastExp})$$

and

$$\left|\frac{x}{y} - q\right| < \mathrm{ulp}(q) \quad (\texttt{RoundedModeUlp}).$$

We will show that the floating point pair (n_r, e_r) with

$$\begin{array}{rcl} n_r & = & n_x \times \beta^{e_x - \min(e_x, e_q+e_y)} \\ & - & n_q \times n_y \times \beta^{e_q+e_y-\min(e_x,e_q+e_y)} \\ e_r & = & \min(e_x, e_q + e_y) \end{array}$$

is a representable pair for r. We easily check that (n_r, e_r) is a representation of $x - qy$. To prove that r is representable, we consider two cases.

First, if $e_q + e_y \leq e_x$, then $e_r = e_q + e_y$ and $e_r \geq -e_{min}$ from (1). We check that

$$\begin{array}{rcl} |n_r| & = & |r| \times \beta^{-e_r} \\ & = & |x - qy| \times \beta^{-e_q-e_y} \\ & \leq & |y| \times \left|\frac{x}{y} - q\right| \times \beta^{-e_q-e_y} \\ & < & |n_y| \times \mathrm{ulp}(q) \times \beta^{-e_q} \\ & < & |n_y| \times \beta^{e'_q-e_q} \\ & < & |n_y| \end{array}$$

and finally, $|n_r| < \beta^p$.

Second, if $e_x < e_q + e_y$ then $e_r = e_x$ and $e_r \geq -e_{min}$. So $|n_r| < \beta^p$ is the only question left to finish our proof. We examine three cases depending on $|n'_q|$. If $|n'_q| = 0$, then $q = 0$ and $r = x$. If $|n'_q| > 1$, we check that

$$\begin{array}{rcl} \left|\frac{x}{y} - q\right| & < & \mathrm{ulp}(q) \\ |n'_q| \times |x - qy| & < & |n'_q| \times \mathrm{ulp}(q) \times |y| \\ & < & |q| \times |y| \\ & \leq & |x| + |x - qy| \end{array}$$

and finally $\left(|n'_q| - 1\right) \times |x-qy| \leq |x|$. Since $|n'_q| \geq 2$, we deduce that $|x - qy| \leq |x|$ and then $|n_r| \leq |n_x| < \beta^p$.

The last case lies when $|n'_q| = 1$. Since (n'_q, e'_q) is a machine representation, it is subnormal and $q = \pm\beta^{-e_{min}}$. We deduce $\beta^{-e_{min}}/2 \leq |x/y|$ from hypothesis (2) and $|x/y| < 2 \times \beta^{-e_{min}}$ since we used a rounding mode, so that the successor of the rounded result bounds the real value. We conclude that

$$\frac{|q|}{2} \leq \left|\frac{x}{y}\right| < 2\,|q| \quad \text{and} \quad \left|\left|\frac{x}{y}\right| - |q|\right| \leq \left|\frac{x}{y}\right|.$$

As $\frac{x}{y}$ and q have the same sign (properties `RleRoundedR0` and `RleRoundedLessR0` of rounding modes independent of the operation),

$$\left|\frac{x}{y} - q\right| \leq \left|\frac{x}{y}\right| \quad \text{and} \quad |x - qy| \leq |x|.$$

That ends the proof.

□

Counter examples against weaker conditions: We will show that both hypotheses (1) & (2) are tight with an example when one of the hypotheses is not satisfied.

- The case where (1) is not satisfied follows an expected path. Let the radix be $\beta = 2$ and the precision $p = 4$ with

  $$\begin{array}{rcl} x & = & 1001_2 \times 2^{-e_{min}+3} \\ y & = & 1101_2 \times 2^{-\left\lfloor \frac{e_{min}}{2} \right\rfloor}. \end{array}$$

  The division is rounded to the nearest, that is towards $-\infty$ here. We get

  $$q = 1011_2 \times 2^{-1-\left\lceil \frac{e_{min}}{2} \right\rceil}.$$

  and $e_q + e_y = -e_{min} - 1$.

  The correcting term $x - qy$ is

  $$r = -1101011_2 \times 2^{-e_{min}-1},$$

  that cannot be represented.

- The case where (2) is not satisfied is more surprising with radix $\beta = 2$ and

$$\begin{aligned} x &= 2^{-e_{min}+p} - 2^{-e_{min}+1} \\ y &= 2^{p+1}. \end{aligned}$$

The exact division

$$x/y = \frac{2^{-e_{min}}}{2} - 2^{-e_{min}-p}$$

is rounded towards $+\infty$ and we get

$$q = 2^{-e_{min}}.$$

The correcting term $x - qy$ is

$$-(2^p + 1) \times 2^{-e_{min}}$$

that cannot be represented and that is surprisingly larger than x. A relatively larger correcting term arises rounding towards $+\infty$ with $x = 2^{-e_{min}+p}$ and $y = 2^{2p+1}$. Such a situation has never been presented in the past but it can be related to the switch from relative error to absolute error in the bounds of residuals presented in [7].

We have proved in theorems `errorBoundedDivClosest` and `errorBoundedDivToZero`, that hypothesis (2) is always true when rounding to a nearest floating point or towards zero.

We will discuss again in Section 7 on the choice of the method used to prove the theorems on the division. The proof published in [2] is based on the usual division algorithm and it uses properties that are not known *a priori* in a proof checking environment.

6 Square root

It is common knowledge that square root extractions are similar to division algorithms in many ways. As a consequence, we have written proofs of this section with the skeleton of the preceding ones. However the inequalities use distinct mathematical properties. Merging the two theorems into a single one is impossible unless we use a system able to prove that a large set of inequalities leads to our goal without human assistance. Such a system does not exist now.

As stated in the following theorem only one condition (3) has to be satisfied. On the other hand, the correcting term can be defined only when the operator precisely rounds the result to a nearest floating point number.

Theorem 5 (errBoundedSqrt in FroundDivSqrt.v) *Let $\circ(\sqrt{\cdot})$ be the implemented square root operation rounded to a nearest floating point value. Given the input x, whenever $\circ(\sqrt{x})$ is neither an infinity nor a NaN, the correcting term*

$$x - \circ(\sqrt{x}) \times \circ(\sqrt{x})$$

is representable if and only if there exist a representable pair (n_q, e_q) representing $\circ(\sqrt{x})$ such that

$$2\ e_q \geq -e_{min}. \tag{3}$$

Proof: Once again, we prove that the exact remainder $r = x - \circ(\sqrt{x}) \times \circ(\sqrt{x})$ computed with appropriate care is representable so that the condition is sufficient. We first define $q = \circ(\sqrt{x})$ rounded to a nearest floating point value. We assume that there exists some representation (n_q, e_q) of q that satisfies (3) and let (n_x, e_x) be a representation of x.

Let (n'_q, e'_q) be the machine representation of q.

$$\left|\sqrt{x} - q\right| \leq \frac{\mathrm{ulp}(q)}{2} \quad (\texttt{ClosestUlp}).$$

We will show that the floating point pair (n_r, e_r) with

$$\begin{aligned} n_r &= n_x \times \beta^{e_x - \min(e_x, 2\ e_q)} \\ &- n_q^2 \times \beta^{2\ e_q - \min(e_x, 2\ e_q)} \\ e_r &= \min(e_x, 2\ e_q) \end{aligned}$$

is a representable pair for r. We easily check that (n_r, e_r) is a representation of $x - q^2$. To prove that r is representable, we consider two cases.

First, if $2\ e_q \leq e_x$, then $e_r = 2\ e_q$ and $e_r \geq -e_{min}$ from (3). We check that

$$\begin{aligned} |n_r| &= \left|x - q^2\right| \times \beta^{-2\ e_q} \\ &= \left|\sqrt{x} - q\right| \times \left|\sqrt{x} + q\right| \times \beta^{-2\ e_q} \\ &\leq \frac{\mathrm{ulp}(q)}{2} \times \left(\left|\sqrt{x} - q\right| + 2\ |q|\right) \times \beta^{-2\ e_q} \\ &\leq \frac{\mathrm{ulp}(q)}{2} \times \left(\frac{\mathrm{ulp}(q)}{2} + 2\ |q|\right) \times \beta^{-2\ e_q} \\ &\leq \frac{1}{4} \times \beta^{2\ e'_q - 2\ e_q} + |n'_q| \times \beta^{2\ e'_q - 2\ e_q} \\ &\leq \frac{1}{4} + |n'_q| \\ &\leq \frac{1}{4} + (\beta^p - 1) \\ &\leq \beta^p - \frac{3}{4} \end{aligned}$$

and finally, $|n_r| < \beta^p$.

Second, if $e_x < 2\ e_q$ then $e_r = e_x$ and $e_r \geq -e_{min}$. So $|n_r| < \beta^p$ is only left to finish our proof. We examine two cases depending on (n'_q, e'_q) is normal or subnormal.

In the case where (n'_q, e'_q) is normal, we use the fact that $\mathrm{ulp}(q) \leq |q| \times \beta^{1-p}$ (`FulpLe2`) with $p \geq 2$, and that $q \geq 0$ (`RleRoundedR0`), so

$$\begin{aligned} q &\leq \sqrt{x} + \frac{\mathrm{ulp}(q)}{2} \\ &\leq \sqrt{x} + \frac{1}{2} \times \beta^{1-p} \times q \\ q &\leq \frac{\sqrt{x}}{1 - \frac{\beta^{1-p}}{2}} \\ q^2 &\leq x \times \left(\frac{1}{1 - \frac{\beta^{1-p}}{2}}\right)^2 \\ &\leq 2\ x \end{aligned}$$

As q^2 and x have the same sign and $q^2 \leq 2\ x$, we have $|x - q^2| \leq x$ and then again $|n_r| \leq |n_x| < \beta^p$.

The case where (n'_q, e'_q) is subnormal cannot happen with any of the *common* single and double precision floating-point formats and it was expectedly dismissed by previous authors. But if the radix is $\beta = 2$, the precision is $p = 5$ and the underflow exponent is $-e_{min} = -3$, the machine representation of the square root of 1, the number represented by $(1, 0)$ is subnormal as it is $(1000_2, -3)$.

Again, such a system would be extremely silly but neither automatic proof checkers nor higher order logic have *common sense*. We would need to define constraints such as the one presented in the IEEE 854 standard [4, § 3.2] to explain that 1 should not be a subnormal number. As a specification should be as concise as possible to remain trusted and since we are able to present a proof without additional definitions, we prefer to limit ourselves to the 17 definitions of our generic specification and no constraint at all but $\beta \geq 2$ and $p \geq 2$.

In this case, $e'_q = -e_{min}$ and

$$\begin{aligned} |n_r| &= \left|x - q^2\right| \times \beta^{-e_x} \\ &= \left|\sqrt{x} - q\right| \times \left|\sqrt{x} + q\right| \times \beta^{-e_x} \\ &\leq \frac{\beta^{-e_{min}}}{2} \times \left(\left|\sqrt{x} - q\right| + 2\ \sqrt{x}\right) \times \beta^{-e_x} \\ &\leq \frac{\beta^{-e_{min}}}{2} \times \left(\frac{\beta^{-e_{min}}}{2} + 2\ \sqrt{x}\right) \times \beta^{-e_x} \\ &\leq \frac{1}{4}\beta^{-2\ e_{min} - e_x} + \sqrt{n_x}\sqrt{\beta^{e_x}}\beta^{-e_{min} - e_x} \\ &\leq \frac{1}{4} + \sqrt{n_x} \times \sqrt{\beta^{-2e_{min} - e_x}} \\ &\leq \frac{1}{4} + n_x \leq \beta^p - \frac{3}{4} \end{aligned}$$

and finally, $|n_r| < \beta^p$, so that the property still holds in this case never studied before.

□

Counter examples against weaker conditions: We will show once again that both hypotheses (3) and the rounding mode being to the nearest are sufficient and tight with an example when one hypothesis is not satisfied.

- Let the radix be $\beta = 2$ and the precision be $p = 4$. The case where (1) is not satisfied follows an expected path. We distinguish whether e_{min} is even or not. In the first case (e_{min} is even), with

  $$\begin{aligned} x &= 1010_2 \times 2^{-e_{min}+1} \\ q &= 1001_2 \times 2^{-1-\frac{e_{min}}{2}}, \end{aligned}$$

  we check that $2\ e_q = -e_{min} - 2$ and the exact correcting term $x - q^2$ is

  $$-2^{-e_{min}-2},$$

  that cannot be represented in our floating-point format.

  In the second case (e_{min} is odd), we end up to the same conclusion with

  $$\begin{aligned} x &= 1010_2 \times 2^{-e_{min}+3} \\ q &= 1101_2 \times 2^{\frac{-1-e_{min}}{2}} \\ x - q^2 &= -1001_2 \times 2^{-e_{min}-1}. \end{aligned}$$

- To prove that precise rounding to a nearest is necessary for the square root operation, we focus on the number $x = 2^{2p+2} - 6 \times 2^{p+1}$ where the radix is $\beta = 2$ and the precision $p > 4$ is arbitrary large. A representation of x is $(2^p - 3, p + 2)_2$.

  The square root of x can be expressed in the form $\sqrt{x} = 2^{p+1} - 3 - \alpha \times 2^{-p}$ with α between $\frac{9}{16}$ and $\frac{9\sqrt{2}}{4}$ from Taylor-Lagrange inequality. Precise rounding would answer $2^{p+1} - 4$ represented by the pair $(2^p - 2, 2)_2$ but some hardware that discards $\alpha \times 2^{-p}$ may round the result to $2^{p+1} - 2$ and return the representable pair $(2^p - 1, 2)_2$.

  In the later case, the correcting term $x - q^2$ is $-2 \times 2^{p+1} - 4 = -(2^p + 1) \times 4$ and it cannot be represented.

  We have just presented the case of rounding to $+\infty$. A counter example rounding to $-\infty$ or toward 0, is obtained with $x = 2^{2p+2} + 6 \times 2^{p+2}$.

7 Remainder

IEEE 754 and 854 standards define the remainder operation that shall be implemented for a system to comply to the standards. Given two inputs x and y, we define n that is the rounding of x/y to the nearest integer value with the even tie breaking rule and the result is defined as

$$r = x - ny.$$

Both documents state that the remainder can always be represented using the same floating point format as the inputs. The best way to prove this assertion is to look at the common stepwise binary implementation of the Euclidean division. As the quotient is computed bit-wise, most significant bit first, the remainder always fit in the same format as the inputs. This invariant was also used in [2] to prove the theorem on divisions and the authors noticed that this invariant is not true for the stepwise square root extraction.

We would have to describe the Euclidean division with much details and properties to port this proof to an automatic proof checker. This is the reason why the question of the remainder being exact has never been answered with an automatic proof checker before [15]. We present in the following theorem a more elementary proof of this fact.

Theorem 6 (errBoundedRem in FroundDivSqrt.v) *Given inputs x and $y \neq 0$, and n the rounded value of x/y to a nearest integer the remainder*

$$x - n \times y$$

is representable if it is neither an infinity nor a NaN

Proof: We define a rounded value of real r to an integer as integer n such that

$$\forall n' \in \mathbb{Z}, \quad |n - r| \leq |n' - r|.$$

We will in fact solely use the property that if n is a rounded value of real r then $|n - r| \leq 1/2$.

We prove the theorem in the case where x and y are non-negative machine pairs. Other cases are handled similarly. Then $r =_{\mathbb{R}} x - n \times y$ is such that $e_r = \min(e_x, e_y)$.

First, if $e_y \leq e_x$ then $e_r = e_y \geq -e_{min}$ and

$$\begin{aligned} |n_r| &= |x - n \times y|\beta^{-e_y} \\ &= |y| \times \left|n - \tfrac{x}{y}\right| \beta^{-e_y} \\ &= |n_y| \times \left|n - \tfrac{x}{y}\right| \\ &\leq |n_y| \times 1/2 \\ &\leq |n_y| \\ &< \beta^p. \end{aligned}$$

Second, if $e_x < e_y$ then $e_r = e_x \geq -e_{min}$. We then have three subcases depending on the value of n.

If $n = 0$ then $r = x$ is representable. If $n = 1$, then $r =_{\mathbb{R}} x - y$ is representable because of Sterbenz's theorem [20]. Indeed as $n = 1$, we have $|1 - x/y| \leq 1/2$ and we deduce that $1/2 \leq x/y \leq 3/2$.

The other cases are impossible. As $x/y - n < 1$, we have $0 \leq x/y < n + 1$ so $n \geq 0$. And as x and y are machine pairs and $e_x < e_y$, we know that $x < y$ (`FcanonicPosFexpRlt`) so $x/y < 1$ and $n \leq 1$. So we have an integer n such that $n \geq 0$, $n \leq 1$ and n is neither 0 nor 1.

□

8 Conclusion

The urge for the development of automatic proof checking is evident from adventures of published proofs such as the ones described in [17]. We have proved and checked in this document old and new properties on floating point arithmetic. Some of the new properties are almost part of the common knowledge of the community of users of floating point arithmetic. However, validating them has made it possible to detect strange, very uncommon and counter-intuitive cases of failure. Should designers have decided to implement the functionality advocated in [2], such cases might have nurtured dormant bugs.

Although the properties shown in this document have been validated with an automatic proof checker, we do not regard them as unshakable truths. We have so far validated a large number of properties based on our specification and we have been able to connect many of these properties with results published in the literature. Working with an automatic proof checker, we like to compare it to a stubborn but helping colleague that will review our proof and will tirelessly ask for details.

As a conclusion, we regard these properties as highly trusted. They incurred a significant amount of testing not reported in the process of the proof. Most proofs have first been written and approved as a pen

and paper proofs before being checked with the computer. The most uncommon achievement of this work is probably the ability to extend our highly trusted properties to any radix, any precision and to digital signal processing circuits implementing a floating point arithmetic slightly different from the IEEE standard.

9 Acknowledgments

This work has benefited from the continuous help of Laurent Théry and Yves Bertot at the INRIA, in Sophia Antipolis. We have heavily used, modified and relied on the first specification of floating point arithmetic in Coq reported in [5] and partially funded by the AOC action of the INRIA.

The authors would also like to thank Peter Markstein for his help in promoting their work and his discussions on intermediate level properties of floating point arithmetic already in use in the development of mathematical libraries.

Finally, we thank the referees for they many comments and suggestions to enhance our work and its readability.

References

[1] R. L. Ashenhurst and N. Metropolis. Unnormalized floating point arithmetic. *Journal of the ACM*, 6(3):415–428, 1959.

[2] G. Bohlender, W. Walter, P. Kornerup, and D. W. Matula. Semantics for exact floating point operations. In P. Kornerup and D. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 22–26, Grenoble, France, 1991.

[3] V. A. Carreño. Interpretation of IEEE-854 floating-point standard and definition in the HOL system. Technical Report Technical Memorandum 110189, NASA Langley Research Center, 1995.

[4] W. J. Cody, R. Karpinski, et al. A proposed radix and word-length independent standard for floating point arithmetic. *IEEE Micro*, 4(4):86–100, 1984.

[5] M. Daumas, L. Rideau, and L. Théry. A generic library of floating-point numbers and its application to exact computing. In *14th International Conference on Theorem Proving in Higher Order Logics*, pages 169–184, Edinburgh, Scotland, 2001.

[6] T. J. Dekker. A floating point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[7] J. Demmel. Underflow and the reliability of numerical software. *SIAM Journal on Scientific and Statistical Computing*, 5(4):887–919, 1984.

[8] J. Harrison. A machine-checked theory of floating point arithmetic. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *12th International Conference on Theorem Proving in Higher Order Logics*, pages 113–130, Nice, France, 1999.

[9] G. Huet, G. Kahn, and C. Paulin-Mohring. The Coq proof assistant: a tutorial: version 7.2. Technical Report 256, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, France, 2002.

[10] C. Jacobi. Formal verification of a theory of IEEE rounding. In *14th International Conference on Theorem Proving in Higher Order Logics*, pages 239–254, Edinburgh, Scotland, 2001. supplemental proceedings.

[11] W. Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.

[12] A. H. Karp and P. Markstein. High precision division and square root. *ACM Transactions on Mathematical Software*, 23(4):561–589, 1997.

[13] S. Linnainmaa. Software for doubled precision floating point computations. *ACM Transactions on Mathematical Software*, 7(3):272–283, 1981.

[14] P. Markstein. *IA-64 and elementary functions: speed and precision*. Prentice Hall, 2000.

[15] P. S. Miner. Defining the IEEE-854 floating-point standard in PVS. Technical Report Technical Memorandum 110167, NASA Langley Research Center, 1995.

[16] O. Møller. Quasi double-precision in floating point addition. *BIT*, 5(1):37–50, 1965.

[17] J. Rushby and F. von Henke. Formal verification of algorithms for critical systems. In *Proceedings of the Conference on Software for Critical Systems*, pages 1–15, New Orleans, Louisiana, 1991.

[18] D. M. Russinoff. A mechanically checked proof of IEEE compliance of the floating point multiplication, division and square root algorithms of the AMD-K7 processor. *LMS Journal of Computation and Mathematics*, 1:148–200, 1998.

[19] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. In *Discrete and Computational Geometry*, volume 18, pages 305–363, 1997.

[20] P. H. Sterbenz. *Floating point computation*. Prentice Hall, 1974.

[21] D. Stevenson et al. An American national standard: IEEE standard for binary floating point arithmetic. *ACM SIGPLAN Notices*, 22(2):9–25, 1987.