A Formally-Proved Algorithm to Compute the Correct Average of Decimal Floating-Point Numbers

Sylvie Boldo, Florian Faissole, and **Vincent Tourneur**[1]

ARITH-25 - June 26th

[1]Thanks to the IEEE for the student travel award.

Introduction

- FP arithmetic: IEEE-754.
- IEEE-754 2008 revision adds radix-10 formats (`decimal32`, `decimal64`).
- Many algorithms designed for radix-2 FP numbers are not valid anymore.

Goal: Adapt an existing algorithm from radix-2 FP numbers literature to radix-10.

Average of two FP numbers

Compute the correct rounding of the average of two FP numbers:

$$\circ\left(\frac{a+b}{2}\right) \quad \text{with } \circ \text{ a rounding to nearest}$$

with as few tests as possible.

Outline

Outline

FP Average in Radix 2

- Studied by Sterbenz (1974):
  - $(a \oplus b) \oslash 2$: accurate, but may overflow when a and b share the same sign.
  - $(a \oslash 2) \oplus (b \oslash 2)$: accurate, except when underflow.
  - $a \oplus ((b \ominus a) \oslash 2)$: less accurate, but does not overflow. when a and b share the same sign
- A corresponding algorithm has been proved by Boldo to guarantee accuracy. This is a long program, since a full sign study is required to choose the correct formula.

FP Average in Radix 2

- Studied by Sterbenz (1974):
  - $(a \oplus b) \oslash 2$: accurate, but may overflow when a and b share the same sign.
  - $(a \oslash 2) \oplus (b \oslash 2)$: accurate, except when underflow.
  - $a \oplus ((b \ominus a) \oslash 2)$: less accurate, but does not overflow. when a and b share the same sign
- A corresponding algorithm has been proved by Boldo to guarantee accuracy. This is a long program, since a full sign study is required to choose the correct formula.

Correctly-Rounded Radix-2 Algorithm

A simpler algorithm that computes the **correctly-rounded** average is formally proved by Boldo (2015). Using radix-2 `binary64` FP numbers:

```
double average(double C, double x, double y) {
  if (C <= abs(x))
    return x/2+y/2;
  else
    return (x+y)/2;
}
```

C is a constant that can be chosen between 2^{-967} and 2^{970}.

Dividing FP numbers by 2

- In radix 2, dividing by 2 is exact (except when underflow).
- In radix 10, there are 2 different cases:
  - If the mantissa is even or small: the result is exact.
  - Otherwise, the mantissa is odd and the result is a **midpoint**.

Outline

Radix-10 FP Numbers Format

In this section, we use the following FP format:

- Radix: 10.
- Mantissa size: 4 digits.
- Unbounded exponent range.
- Rounding to nearest, tie-breaking to even.

Algorithms based on $(a+b)/2$

Algorithm: $(a \oplus b) \oslash 2$

Counter-example of correct rounding:

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001
b	1000
$a+b$	30010000001
$a \oplus b$	3001
$(a \oplus b)/2$	15005
$(a \oplus b) \oslash 2$	1500
$(a+b)/2$	150050000005
$\circ((a+b)/2)$	1501

$a/2$ is a midpoint, but b is positive, so the rounding should have been towards $+\infty$.

Algorithms based on $(a+b)/2$

Algorithm: $(a \oplus b) \oslash 2$

Counter-example of correct rounding:

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001
b	1000
$a+b$	3001 0000 001
$a \oplus b$	3001
$(a \oplus b)/2$	1500 5
$(a \oplus b) \oslash 2$	1500
$(a+b)/2$	1500 5000 0005
$\circ((a+b)/2)$	1501

$a/2$ is a midpoint, but b is positive, so the rounding should have been towards $+\infty$.

Algorithms based on $(a+b)/2$

Algorithm: $(a \oplus b) \oslash 2$

Counter-example of correct rounding:

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001
b	1000
$a+b$	3001 0000 001
$a \oplus b$	3001
$(a \oplus b)/2$	1500 5
$(a \oplus b) \oslash 2$	1500
$(a+b)/2$	1500 5000 0005
$\circ((a+b)/2)$	1501

$a/2$ is a midpoint, but b is positive, so the rounding should have been towards $+\infty$.

Algorithms based on $(a/2) + (b/2)$

Algorithm: $(a \oslash 2) \oplus (b \oslash 2)$

Counter-example of correct rounding: (same)

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001		
b			1000
$a/2$	1500	5	
$a \oslash 2$	1500		
$b/2$			5000
$(a \oslash 2) + (b \oslash 2)$	1500	0000	0005
$(a \oslash 2) \oplus (b \oslash 2)$	1500		
$(a + b)/2$	1500	5000	0005
$\circ((a + b)/2)$	1501		

Same issue, $a/2$ is a midpoint, and is rounded before taking into account the value of b.

Algorithms based on $(a/2) + (b/2)$

Algorithm: $(a \oslash 2) \oplus (b \oslash 2)$

Counter-example of correct rounding: (same)

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001			
b			1000	
$a/2$	1500	5		
$a \oslash 2$	1500			
$b/2$				5000
$(a \oslash 2) + (b \oslash 2)$	1500	00000005		
$(a \oslash 2) \oplus (b \oslash 2)$	1500			
$(a+b)/2$	1500	50000005		
$\circ((a+b)/2)$	1501			

Same issue, $a/2$ is a midpoint, and is rounded before taking into account the value of b.

Algorithms based on $(a/2) + (b/2)$

Algorithm: $(a \oslash 2) \oplus (b \oslash 2)$

Counter-example of correct rounding: (same)

$$a = 3001 \times 10^{10}, b = 1000 \times 10^{0}$$

a	3001
b	1000
$a/2$	15005
$a \oslash 2$	1500
$b/2$	5000
$(a \oslash 2) + (b \oslash 2)$	1500 0000 0005
$(a \oslash 2) \oplus (b \oslash 2)$	1500
$(a + b)/2$	1500 5000 0005
$\circ((a + b)/2)$	1501

Same issue, $a/2$ is a midpoint, and is rounded before taking into account the value of b.

Algorithms based on $(a/2) + (b/2)$, using FMA

Algorithm: $\circ(a \times 0.5 + (b \oslash 2))$
There is one rounding less thanks to the FMA operator.

Counter-example of correct rounding:

$$a = 2001 \times 10^{10}, b = 2001 \times 10^{8}$$

a	2001
b	2001
$b/2$	10005
$b \oslash 2$	1000
$a \times 0.5$	10005
$a \times 0.5 + (b \oslash 2)$	10105
$\circ(a \times 0.5 + (b \oslash 2))$	1010
$(a+b)/2$	1010505
$\circ((a+b)/2)$	1011

Algorithms based on $(a/2) + (b/2)$, using FMA

Algorithm: $\circ(a \times 0.5 + (b \oslash 2))$
There is one rounding less thanks to the FMA operator.

Counter-example of correct rounding:

$$a = 2001 \times 10^{10}, b = 2001 \times 10^{8}$$

a	2001
b	2001
$b/2$	10005
$b \oslash 2$	1000
$a \times 0.5$	10005
$a \times 0.5 + (b \oslash 2)$	10105
$\circ(a \times 0.5 + (b \oslash 2))$	1010
$(a+b)/2$	1010505
$\circ((a+b)/2)$	1011

Algorithms based on $(a/2) + (b/2)$, using FMA

Algorithm: $\circ(a \times 0.5 + (b \oslash 2))$
There is one rounding less thanks to the FMA operator.

Counter-example of correct rounding:

$$a = 2001 \times 10^{10}, b = 2001 \times 10^{8}$$

a	2001
b	2001
$b/2$	10005
$b \oslash 2$	1000
$a \times 0.5$	10005
$a \times 0.5 + (b \oslash 2)$	10105
$\circ(a \times 0.5 + (b \oslash 2))$	1010
$(a+b)/2$	1010505
$\circ((a+b)/2)$	1011

Outline

TwoSum

TwoSum (x, y) computes (with 6 flops) the sum of x and y, and the rounding error. It works in radix-10 and returns the rounding and the error of an FP addition (always representable exactly by an FP number).

$$(a, b) = \text{TwoSum}\,(x, y) \implies$$

$$x + y = a + b \quad \wedge \quad |b| \leq \frac{\text{ulp}\,(a)}{2}$$

Sketch of the Proof with Unbounded Exponent Range

```
1 Function Average10(x, y)
2     (a, b) = TwoSum (x, y)
3     if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
4         return ∘(b × 0.5 + (a ⊘ 2))
5     else
6         return ∘(a × 0.5 + b)
```

- The `if` checks whether $a/2$ is a FP number.
- If $a/2 \in \mathbb{F}$, we have $a \oslash 2 = a/2$. So the computations of line 4 are exact until the last rounding.
- In the other case, $a/2$ is a midpoint and we rely on the following lemma.
- In the other case, b is not divided by 2 contrary to intuition.

Sketch of the Proof with Unbounded Exponent Range

```
1 Function Average10(x, y)
2     (a, b) = TwoSum(x, y)
3     if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
4         return ∘(b × 0.5 + (a ⊘ 2))
5     else
6         return ∘(a × 0.5 + b)
```

- The `if` checks whether $a/2$ is a FP number.
- If $a/2 \in \mathbb{F}$, we have $a \oslash 2 = a/2$. So the computations of line 4 are exact until the last rounding.
- In the other case, $a/2$ is a midpoint and we rely on the following lemma.
- In the other case, b is not divided by 2 contrary to intuition.

Sketch of the Proof with Unbounded Exponent Range

```
1 Function Average10(x, y)
2     (a, b) = TwoSum (x, y)
3     if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
4         return ∘(b × 0.5 + (a ⊘ 2))
5     else
6         return ∘(a × 0.5 + b)
```

- The `if` checks whether $a/2$ is a FP number.
- If $a/2 \in \mathbb{F}$, we have $a \oslash 2 = a/2$. So the computations of line 4 are exact until the last rounding.
- In the other case, $a/2$ is a midpoint and we rely on the following lemma.
- In the other case, b is not divided by 2 contrary to intuition.

Sketch of the Proof with Unbounded Exponent Range

```
1 Function Average10(x, y)
2     (a, b) = TwoSum (x, y)
3     if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
4         return ∘(b × 0.5 + (a ⊘ 2))
5     else
6         return ∘(a × 0.5 + b)
```

- The if checks whether $a/2$ is a FP number.
- If $a/2 \in \mathbb{F}$, we have $a \oslash 2 = a/2$. So the computations of line 4 are exact until the last rounding.
- In the other case, $a/2$ is a midpoint and we rely on the following lemma.
- In the other case, b is not divided by 2 contrary to intuition.

Sketch of the Proof with Unbounded Exponent Range

```
1 Function Average10(x, y)
2     (a, b) = TwoSum (x, y)
3     if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
4         return ∘(b × 0.5 + (a ⊘ 2))
5     else
6         return ∘(a × 0.5 + b)
```

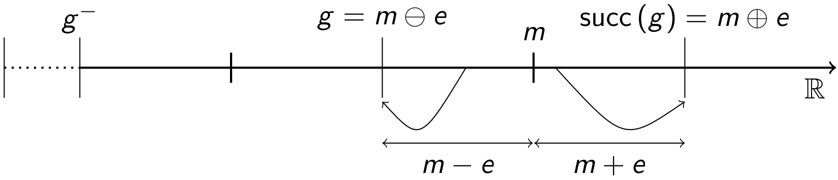
- The `if` checks whether $a/2$ is a FP number.
- If $a/2 \in \mathbb{F}$, we have $a \oslash 2 = a/2$. So the computations of line 4 are exact until the last rounding.
- In the other case, $a/2$ is a midpoint and we rely on the following lemma.
- In the other case, b is not divided by 2 contrary to intuition.

Technical Lemma

Lemma (Midpoint)

Let $m = g + \frac{\text{ulp}(g)}{2}$ *with* $g \in \mathbb{F}$, $m > 0$ *and* $0 < e \leq \frac{\text{ulp}(g)}{2}$.

- $m \ominus e = g$
- $m \oplus e = \text{succ}(g)$



Outline

Formal Proof Software

- The Coq proof assistant
- Floating-Point numbers library: `Flocq` (Boldo-Melquiond), which provides an FP numbers formalization and many results.
- There are several FP formats in Flocq, defined as subsets of reals numbers $\mathbb{R}$. All formats depend on a radix (β).
  - `FLX`: unbounded exponent range.
  - `FLT`: exponent has a minimal value (gradual underflow).

Format	Parameters	Constraints
FLX	β, p	$\|m\| < \beta^p$
FLT	β, p, e_{min}	$\|m\| < \beta^p,\ e \geq e_{min}$

A real number is a FP number if equal to $m \times \beta^e$

Definition of the Algorithm

We define our algorithm in Coq:

```
Definition average10 (x y : R) :=
  if (Req_bool (round (x/2 - round (x/2))) 0)
    then round (y/2 + round (x/2))
    else round (x/2 + y).
```

We assume that this function is called with the output of `TwoSum`.

Main Theorem

This is the main theorem, stating the correctness of the algorithm:

```
Theorem average10_correct :
  forall a b, format a → format b →
  Rabs b <= (ulp a) / 2 →
  average10 a b = round ((a + b) / 2).
```

- `format x` means that $x \in \mathbb{F}$. We define it depending on the chosen format.
- `round x` is $\circ(x)$. It also depends on the format, and is a rounding to nearest, with an arbitrary tie.
- `ulp x` is $\text{ulp}(x)$.

Main Theorem

This is the main theorem, stating the correctness of the algorithm:

```
Theorem average10_correct :
  forall a b, format a → format b →
  Rabs b <= (ulp a) / 2 →
  average10 a b = round ((a + b) / 2).
```

- `format x` means that $x \in \mathbb{F}$. We define it depending on the chosen format.
- `round x` is $\circ(x)$. It also depends on the format, and is a rounding to nearest, with an arbitrary tie.
- `ulp x` is $\mathrm{ulp}(x)$.

Proofs and Generalizations

```
Function Average10(x, y)
    (a, b) = TwoSum (x, y)
    if ∘(a × 0.5 − (a ⊘ 2)) = 0 then
        return ∘(b × 0.5 + (a ⊘ 2))
    else
        return ∘(a × 0.5 + b)
```

- We first prove it with an unbounded exponent range (FLX).
- We then prove that it holds with gradual underflow (FLT).
  - The test $\circ(a \times 0.5 - (a \oslash 2)) = 0$ is not equivalent to $a/2 \in \mathbb{F}$.
  - In the else case, one must compute $\circ(a \times 0.5 + b)$, instead of $\circ(a \times 0.5 + b \oslash 2)$ (both would work in FLX).
- We generalize it for any even radix.

Outline

Conclusion

- Summary:
  - The algorithm computes the correct rounding (to nearest) of the average of two FP numbers: $\circ((a+b)/2)$.
  - It holds with gradual underflow.
  - It holds with any tie-breaking rule.
  - It is formally-proved.
  - It has been generalized to any even radix.

- We have problems with spurious overflows (due to TwoSum).

- We showed that it may not be straightforward to adapt some existing algorithms from radix-2 literature to radix-10.

Conclusion

- Summary:
  - The algorithm computes the correct rounding (to nearest) of the average of two FP numbers: $\circ((a+b)/2)$.
  - It holds with gradual underflow.
  - It holds with any tie-breaking rule.
  - It is formally-proved.
  - It has been generalized to any even radix.
- We have problems with spurious overflows (due to TwoSum).
- We showed that it may not be straightforward to adapt some existing algorithms from radix-2 literature to radix-10.