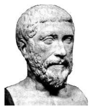
Formal Proofs
for
Computer Arithmetics

Laurent Théry
Marelle Project INRIA Sophia-Antipolis

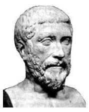
Computer arithmetics



Numbers are everywhere:

- Billing system
- Weather forecast
- Computer vision
- Cryptography
- . . .

Computer arithmetics



Different flavours:

- Integer Arithmetic: $a \in \mathbb{N}$
- Rational Arithmetic: $a \in \mathbb{Q}$
- Floating-Point Arithmetic: $a \in \mathbb{F}$
- Interval Arithmetic: $[a, b] \in \mathbb{F} \times \mathbb{F}$
- Exact Arithmetic: $a \in \mathbb{R}$
- . . .

Formal Proofs

Therac-25:

Arithmetic overflows could cause the software to bypass safety checks.

Patriot Missile:

The calculation of the time since boot was inacurate.

Pentium FDIV

Some division operations were wrong by a very small amount.

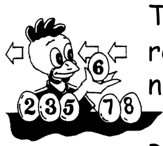
Ariane 5:

A conversion 64 bit floating-point number to 16 bit signed integer failed.

Outline

- A simple example that illustrates various aspects of formal verification.
- An overview of the formalisation of floating-point arithmetic.

An Example



Take a program that given a parameter n returns the list of the first n prime numbers

Prove it correct

Knuth: The Art of Computer Programming

Natural Numbers

```
Inductive ℕ:  Set :=
   O : ℕ
|  S : ℕ→ℕ.
```

```
O, S(O), S(S(O)), ...
```

$$\left.\begin{array}{l} P(\mathtt{O}) \\ \wedge \\ \forall n,\ P(n) \Rightarrow P(\mathtt{S}(n)) \end{array}\right. \Rightarrow \forall n,\ P(n)$$

Addition

```
Fixpoint + [a,b:ℕ] :  ℕ :=
match a with
      O => b
 | S a' => S(a' + b)
end.
```


Multiplication

```
Fixpoint * [a,b:ℕ] :  ℕ :=
match a with
      O => O
 | S a' => b + (a' * b)
end.
```


Divisibility

```
Definition a|b := ∃c,  b = c * a.
```


Primality

```
Definition prime(p) :=
     ∀n, n|p ⇒ n = 1 ∨ n = p
   ∧
     p ≠ 1.
```


Program Correctness

Verification Condition Generator

Program + Annotations ⇒ Conditions

Krakatoa: Java ⟶ Coq

Program Correctness

$$\{\ \textit{Pre-conditions}\ \}$$
$$\mathbf{P}$$
$$\{\ \textit{Post-conditions}\ \}$$

Example

```
{ odd(x) }
  x = x + 2;
{ odd(x) }
```

Generated condition:

$$\forall x,\ \mathtt{odd}(x) \Rightarrow \mathtt{odd}(x+2)$$

Program Correctness

Loop:

```
while (C) {
{ invariant I
  variant   V
}
     P

}
```


Program Correctness

Example:

```
while (0 < i--) {
  { invariant odd(x)
    variant  i
  }
    x = x + 2
}
```

Generated conditions:

$$\forall x.\ \mathtt{odd}(x) \Rightarrow \mathtt{odd}(x+2)$$
$$\forall i.\ 0 \leq i-1 < i$$

Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
```

$$\left\{\begin{array}{l} \forall k.\ 0 \leq k < n \Rightarrow \mathtt{prime}(res[k]) \\ \wedge \\ \forall k, j.\ 0 \leq k < j < n \Rightarrow res[k] < res[j] \\ \wedge \\ \forall k.\ \wedge \begin{array}{c} 0 \leq k \leq res[n-1] \\ \mathtt{prime}(k) \end{array} \Rightarrow \exists j.\ \wedge \begin{array}{c} 0 \leq j < n \\ res[j] = k \end{array} \end{array}\right\}$$

```
  return res;
}
```


Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
  int number = 2;
  boolean isPrime;
  for (int i = 0; i < n; i++) {
    while (true) {
      isPrime = true;
      for (int j = 2; j < number; j++) {
        if (number % j == 0) {
          isPrime = false;
          break;
        }
      }
      if (isPrime) break;
      number++;
    }
    res[i] = number;
    number++;
  }
  return res;
}
```

2	3	5	7	11

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11

$$\forall p,\ \mathtt{prime}(p) \Rightarrow 2 \leq p$$

$$\forall n,\ \exists p,\ n < p \wedge \mathtt{prime}(p)$$

$$\forall m\, n,\ m \mid n \wedge n \neq 0 \Rightarrow m < n$$

Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
  int number = 2;
  boolean isPrime;
  for (int i = 0; i < n; i++) {
    while (true) {
      isPrime = true;
      for (int j = 0; j < i; j++) {
        if (number % res[j] == 0) {
          isPrime = false;
          break;
        }
      }
      if (isPrime) break;
      number++;
    }
    res[i] = number;
    number++;
  }
  return res;
}
```

2	3	5	7	11

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11

$$\forall n,\ 2 \leq n \Rightarrow (\forall p, \mathtt{prime}(p) \wedge p < n \Rightarrow \neg(p \mid n)) \Rightarrow \mathtt{prime}(n)$$

Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
  res[0] = 2;
  int number = 3;
  boolean isPrime;
  for (int i = 1; i < n; i++) {
    while (true) {
      isPrime = true;
      for (int j = 1; j < i; j++) {
        if (number % res[j] == 0) {
          isPrime = false;
          break;
        }
      }
      if (isPrime) break;
      number += 2;
    }
    res[i] = number;
    number += 2;
  }
  return res;
}
```

2	3	5	7	11

3 5 7 ~~9~~ 11

prime(2)

$\forall p,\ \mathtt{prime}(p) \Rightarrow p = 2 \vee \mathtt{odd}(p)$

Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
  res[0] = 2;
  int number = 3, snum;
  boolean isPrime;
  for (int i = 1; i < n; i++) {
    while (true) {
      isPrime = true;
      snum = (int) Math.sqrt(number);
      for (int j = 1; j < i && res[j] <= snum; j++) {
        if (number % res[j] == 0) {
          isPrime = false;
          break;
        }
      }
      if (isPrime) break;
      number += 2;
    }
    res[i] = number;
    number += 2;
  }
  return res;
}
```

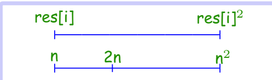
2	3	5	7	11

3 5 7 ~~9~~ 11

$\forall n\, p\, q,\ n = p * q \Rightarrow p \leq \sqrt{n} \vee q \leq \sqrt{n}$

Knuth Algorithm

```
int[] firstPrimes(int n) {
  int[] res = new int[n];
  res[0]=2;
  int number=3, snum;
  boolean isPrime;
  for (int i=1; i<n; i++) {
    while (true) {
      isPrime=true;
      snum = (int)Math.sqrt(number);
      for(int j=0; res[j]<= snum; j++) {
        if (number%res[j]==0) {
          isPrime=false;
          break;
        }
      }
      if (isPrime) break;
      number+=2;
    }
    res[i]=number;
    number+=2;
  }
  return res;
}
```



$\forall n,\ 2 \leq n \Rightarrow$
$\exists p,\ \mathtt{prime}(p) \wedge n < p < 2 * n$

Bertrand Postulate

For n greater than 2, there is always at least one prime number strictly between n and $2n$.

Proof by Contradiction (Erdös)

Upper Bound: $\binom{2n}{n} < (2n)^{\sqrt{2n}/2-1} 4^{2n/3}$

Lower Bound: $4^n \leq 2n\binom{2n}{n}$

Necessary Condition: $4^{n/3} < (2n)^{\sqrt{2n}/2}$

Example of properties

Upper Bound on the Product of Prime Numbers

$$\prod_{p \leq n} p < 4^n$$

By Strong Induction on n

$2 < 4^2$

If n is odd, $\prod_{p \leq n+1} p = \prod_{p \leq n} p < 4^n < 4^{n+1}$

If n is even, $\prod_{p \leq 2m+1} p = (\prod_{p \leq m+1} p) \quad (\prod_{m+1 < p \leq 2m+1} p)$

$\prod_{p \leq 2m+1} p < 4^{m+1} \quad (\prod_{m+1 < p \leq 2m+1} p)$

$\prod_{p \leq 2m+1} p < 4^{m+1} \binom{2m+1}{m+1}$

$\prod_{p \leq 2m+1} p < 4^{m+1} 4^m$

$\prod_{p \leq 2m+1} p < 4^{2m+1}$

Necessary Condition

$4^{n/3} < (2n)^{\sqrt{2n}/2}$

Logarithmic Scale

$\frac{n}{3}\, ln(4) < \frac{\sqrt{2n}}{2} ln(2n)$

Simplification:

$\sqrt{8n}\, ln(2) - 3\, ln(2n) < 0$

Function Analysis

Inequality: $\sqrt{8n}\,ln(2) - 3\,ln(2n) < 0$

Function: $f(x) = \sqrt{8x}\,ln(2) - 3ln(2x)$

Evaluation: $f(2^7) = 2^5 ln(2) - 3.2^3 ln(2) > 0$

Derivative: $f'(x) = \dfrac{\sqrt{2x}\,ln(2) - 3}{x}$

Conclusion: For $n \geq 2^7$, ok.

Remaining cases

The theorem is true for $n < 2^7$

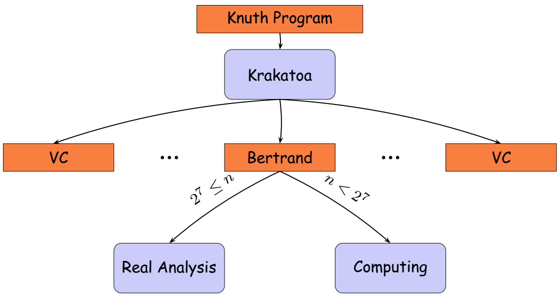
Computing inside Coq:

Write a program that checks the property

Prove it correct

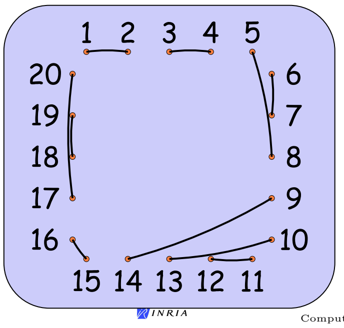
Run it

Proof Map



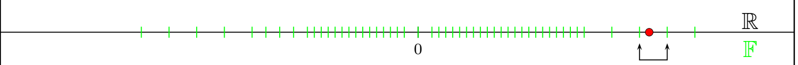
Little Problem

Sort the numbers from 1 to $2n$ in pairs (a_i, b_i) such that each $a_i + b_i$ is prime ?



Floating-Point Arithmetic

Floating-Point Numbers:



Correct Rounding (`IEEE 754`):

$$a \oplus b = \circ(a + b)$$

Floating-Point Numbers

Normal ($e > 0$): $(-1)^s \beta^{e-B}(1 + \sum_{i=1}^{p-1} f_i \beta^{-i})$

Subnormal ($e = 0$): $(-1)^s \beta^{1-B} \sum_{i=1}^{p-1} f_i \beta^{-i}$

$$x \ominus y = 0 \Longleftrightarrow x = y$$

Floating-Point Numbers

Number as pair:

Definition $\mathbb{F}$:= $\mathbb{Z} \times \mathbb{Z}$.

Projections:

Definition m(p) := let (x,_) = p in x.

Definition e(p) := let (_,y) = p in y.

Floating-Point Numbers

Value:

Definition v(p) := m(p) $* \beta^{\mathrm{e}(p)}$.

Equivalence:

Definition $p \simeq q$:= v(p) = v(q).

Bound

Bound as pair:

Definition $\mathbb{B}$:= $\mathbb{N} \times \mathbb{N}$.

Projections:

Definition M(b) := let (x,_) = b in x.

Definition E(b) := let (_,y) = b in y.

Bound

Bounded number:

Definition $\mathcal{B}_b(p)$:= $|\mathrm{m}(p)| \leq \mathrm{M}(b) \wedge -\mathrm{E}(b) \leq \mathrm{e}(p)$.

Rounding

Rounded Mode:
Toward +∞, Toward -∞, Toward 0, Closest.

Rounded as a Predicate:

$\mathcal{R}: \mathbb{R} \rightarrow \mathbb{F} \rightarrow$ Prop

Rounding: Min Max

Elements on the left and on the right:

Definition isMin(r,p) :=
$\mathcal{B}_b(p) \wedge p \leq r \wedge \forall q,\ \mathcal{B}_b(q) \wedge q \leq r \Rightarrow q \leq p$.

Definition isMax(r,p) :=
$\mathcal{B}_b(p) \wedge r \leq p \wedge \forall q,\ \mathcal{B}_b(q) \wedge r \leq q \Rightarrow p \leq q$.

Rounding chooses one of those:

Definition MinOrMax($\mathcal{R}$) :=
$\forall p\, r,\ \mathcal{R}(r,p) \Rightarrow$ isMin$(r,p) \vee$ isMax(r,p).

Rounding: Monotone

Rounding is non decreasing:

`Definition Monotone(`$\mathcal{R}$`) :=`
$\forall p_1\, p_2\, r_1\, r_2,\ \mathcal{R}(r_1, p_1) \wedge \mathcal{R}(r_2, p_2) \wedge r_1 < r_2 \Rightarrow p_1 \le p_2.$

Rounding is total:

`Definition Total(`$\mathcal{R}$`) :=` $\forall r,\ \exists p,\ \mathcal{R}(r, p).$

Rounding is compatible:

`Definition Compatible(`$\mathcal{R}$`) :=`
$\forall p_1\, p_2\, r,\ \mathcal{R}(r, p_1) \wedge \mathcal{B}_p(p_2) \wedge p_1 \simeq p_2 \Rightarrow \mathcal{R}(r, p_2).$

Example: Malcolm Test

```
float malcolmTest() {
  float x = 1;
  float y = 1;
  while (((x + 1) - x) == 1) {
    x = 2 * x;
  }
  while (((x + y) - x) != y) {
    y = y + 1;
  }
  return y;
}
```

x = (, 0)
x = (, 1)

$1 = (1, 0) \qquad \beta = (1, 1)$
$(m+1, e) \ominus (m, e) = (1, e)$
$(m, e) \oplus (1, e) \simeq (m+1, e)$
$(m_1, e) \ominus (m_2, e) = (m_1 - m_2, e)$

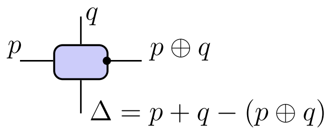
Example: Expansion

Ordered list of non-overlapping floats:

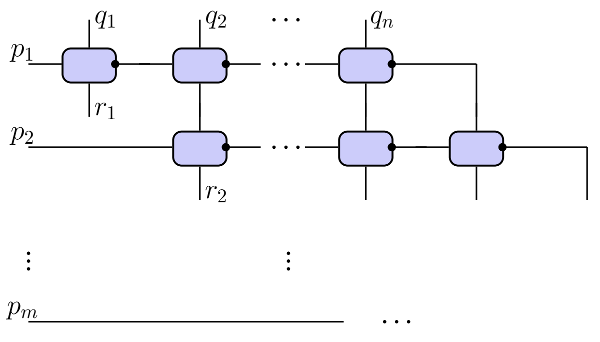
11011000111000000000001111111110000010

(11011,29); (11100,21); (11111,9); (11000,4); (10000,-3)

Addition:



Example: Expansion

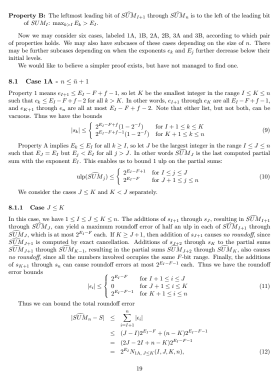


Pencil and Paper Proof

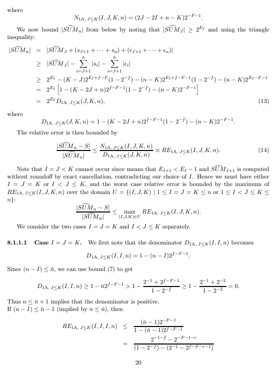
J. Demmel and Y. Hida,
Accurate floating point summation

19 page proof

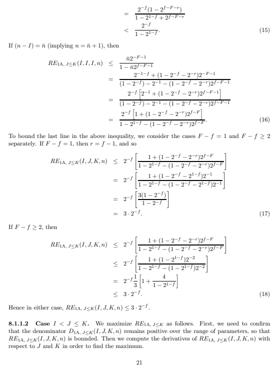
Pencil and Paper Proof



Pencil and Paper Proof



Pencil and Paper Proof



Benefits

Finding bugs in proof

typos

missing cases

missing side-conditions

large versus strict inequalities

...

$\longrightarrow$ Effective?

Benefits

Library of validated facts

statements as general as possible

explicit side-conditions

Example (Sterbenz)

if $\frac{1}{2}y \leq x \leq 2y$ then $x \ominus y = x - y$

Benefits

Improved Results

Example (Fast Two Sum)

$$b \ominus ((a \oplus b) \ominus a) = (a + b) - (a \oplus b)$$

if $|a| \leq |b|$

if $e_a \leq e_b$ where $a = \overline{m}_a 2^{e_a}$ and $b = \overline{m}_b 2^{e_b}$

if $e_a \leq e_b$ where $a = m_a 2^{e_a}$ and $b = m_b 2^{e_b}$

Benefits

Alternative Proofs

Different metric

Completly new proofs

Challenges

Pencil and Paper Proof + Formal Proof

Single Program ⇝ Library

Proof Checking ⇝ Computer Aided Proof

Pencil and Paper Proof

Proof + Annotations ⇒ Conditions

CYP tool: text ⟶ Coq

Colouring Proof

Theorem **Sterbenz**: Let p and q such that $bounded(p)$ and $bounded(q)$, if $q/2 \leq p \leq 2 * q$ then $bounded(p-q)$.

The proof proceeds like this. First of all, we restrict ourselves to the case $q \leq p \leq 2 * q$ because of the symmetry of the problem. For the exponent, by definition of the substraction, $e(p-q) = min(e(p), e(q))$, so $e(p-q) \geq -E(bound)$ since both p and q are bounded. For the mantissa, we do a case analysis on the value of $\min(e(p), e(q))$. If $min(e(p), e(q)) = e(q)$, the initial equation can be rewritten as $0 \leq p-q \leq q$ and since $e(p-q) = e(q)$, we obtain $0 \leq m(p-q) \leq m(q)$. As $bounded(q)$, we have $0 \leq m(p-q) < M(bound)$. Similarly if $min(e(p), e(q)) = e(p)$, we can rewrite the initial equation as $0 \leq p-q \leq q \leq p$ and since $e(p-q) = e(p)$ we have $0 \leq m(p-q) \leq m(p)$. In both cases we have $0 \leq m(p-q) < M(bound)$. The mantissa and the exponent are then bounded, so we have $bounded(p-q)$.

Related Works

Floating point arithmetic

Barrett (Z), Miner (Pvs), Russinoff (Acl2), Harrison (Hol), Boldo & al (Coq)

Interval arithmetic

Melquiond (Coq)

Multiprecision arithmetic

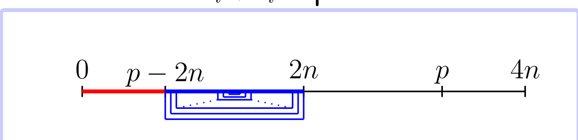
Bertot & al (Coq), Bondyfalat (Coq)

Exact arithmetic

Ciaffaglione & al (Coq), Lester & al (Pvs), Niqui (Coq)

Little problem

Sort the numbers from 1 to $2n$ in pairs (a_i, b_i) such that each $a_i + b_i$ is prime ?



p is prime

$p - 2n$ is odd

$p - 2n - 1$ is even