

Formal Proofs for Computer Arithmetics

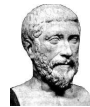
Laurent Théry
Marelle Project INRIA Sophia-Antipolis



Computer Arithmetics – p.1

Computer arithmetics

Numbers are everywhere:



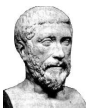
Billing system
Weather forecast
Computer vision
Cryptography
...



Computer Arithmetics – p.2

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: $a \in \mathbb{R}$
...



Computer Arithmetics – p.3

Formal Proofs

Therac-25:

Arithmetic overflows could cause the software to bypass safety checks.



Patriot Missile:

The calculation of the time since boot was inaccurate.

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.



Computer Arithmetics – p.4

Outline

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



Computer Arithmetics – p.5

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



Computer Arithmetics – p.6

Natural Numbers

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

0, S(0), S(S(0)), ...

$$P(0) \wedge \forall n, P(n) \Rightarrow P(S(n)) \Rightarrow \forall n, P(n)$$

Addition

```
Fixpoint + [a,b:N] : N :=  
  match a with  
  | 0 => b  
  | S a' => S(a' + b)  
end.
```

Multiplication

```
Fixpoint * [a,b:N] : N :=  
  match a with  
  | 0 => 0  
  | S a' => b + (a' * b)  
end.
```

Divisibility

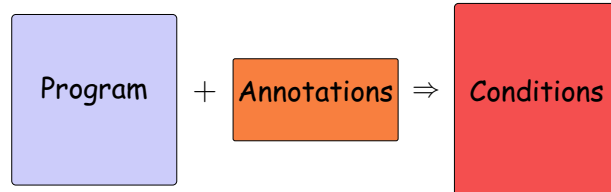
Definition $a|b := \exists c, b = c * a$.

Primality

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

Program Correctness

Verification Condition Generator



Krakatoa: Java → Coq

Program Correctness

Example

```

{ Pre-conditions }
  P
{ Post-conditions }

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

Generated condition:

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

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:

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

$$\forall i, 0 \leq i - 1 < i$$

Knuth Algorithm

```

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

```

{
   $\forall k, 0 \leq k < n \Rightarrow \text{prime}(\text{res}[k])$ 
   $\wedge$ 
   $\forall k, j, 0 \leq k < j < n \Rightarrow \text{res}[k] < \text{res}[j]$ 
   $\wedge$ 
   $\forall k, \wedge 0 \leq k \leq \text{res}[n-1] \Rightarrow \exists j, \wedge 0 \leq j < n$ 
   $\text{prime}(k) \Rightarrow \text{res}[j] = k$ 
}
    
```

```

  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

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

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

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

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

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

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	11	

prime(2)

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

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	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;
}
```

res[i]	res[i] ²
n	2n
n ²	

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

Bertrand Postulate

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

Proof by Contradiction (Erdős)

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, $\prod_{p \leq 2m+1} p = (\prod_{p \leq m+1} p) (\prod_{m+1 < p \leq 2m+1} p)$

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

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

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

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

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

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

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

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

Remaining cases

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

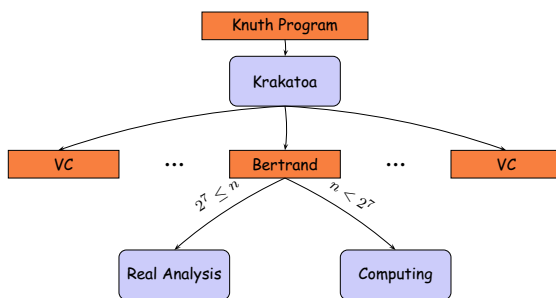
Computing inside Coq:

Write a program that checks the property

Prove it correct

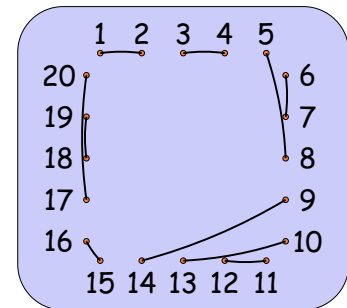
Run it

Proof Map



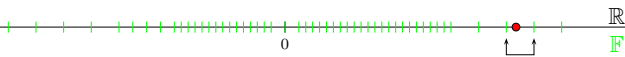
Little Problem

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):

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

Floating-Point Numbers



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

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

$$x \ominus y = 0 \iff x = y$$

Floating-Point Numbers

Number as pair:

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

Projections:

Definition $m(p) := \text{let } (x, _) = p \text{ in } x$.

Definition $e(p) := \text{let } (_, y) = p \text{ in } y$.

Floating-Point Numbers

Value:

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

Equivalence:

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

Bound

Bound as pair:

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

Projections:

Definition $M(b) := \text{let } (x, _) = b \text{ in } x$.

Definition $E(b) := \text{let } (_, y) = b \text{ in } y$.

Bound

Bounded number:

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



Rounding

Rounded Mode:

Toward $+\infty$, Toward $-\infty$, Toward 0, Closest.

Rounded as a Predicate:

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

Rounding: Min Max

Elements on the left and on the right:

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

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

Rounding chooses one of those:

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

Rounding: Monotone

Rounding is non decreasing:

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

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;
}
```

$$\begin{aligned}
 1 &= (1, 0) & \beta &= (1, 1) \\
 (m+1, e) \ominus (m, e) &= (1, e) \\
 (m, e) \oplus (1, e) &\simeq (m+1, e) \\
 (m_1, e) \ominus (m_2, e) &= (m_1 - m_2, e)
 \end{aligned}$$

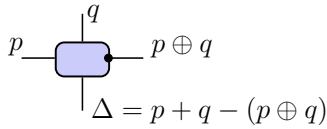
Example: Expansion

Ordered list of non-overlapping floats:

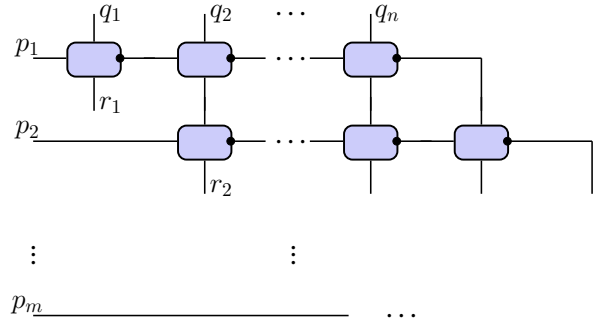
1101100011100000000001111110000010

(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

Property B: The leftmost leading bit of $SE_{M_{j+1}}$ through SE_{M_j} is to the left of the leading bit of SE_{M_j} minus $m_{j+1} E_j > 0$.

Now we may consider six cases, labeled 1A, 1B, 2A, 2B, 3A and 3B, according to which pair of properties holds. We may also have outcomes of these cases depending on the size of n . There may be further outcomes depending on when the exponent e_n and E_j differ between these initial levels.

We would like to believe a simpler proof exists, but have not managed to find one.

8.1 Case 1A - $n \leq n + 1$

Property 3 means $e_{j+1} \leq E_j - F - j - 1$, so let K be the smallest integer in the range $F \leq K \leq n$ such that $m_{j+1} \leq E_j - F - j - 2$ and $m_{j+1} > K$. In other words, m_{j+1} through m_n are all $E_j - F - j - 1$, and m_{j+1} through m_n are all at least $E_j - F - j - 2$. Note that other bits, but not both, can be nonzero. Thus we have the bounds:

$$\begin{aligned}
 |a_i| &\leq \begin{cases} 2^{m_{j+1}-i} (1 - 2^{-n}) & \text{for } 1 \leq i \leq K \\ 2^{m_{j+1}-i} (1 - 2^{-n}) & \text{for } K+1 \leq i \leq n \end{cases} \quad (9)
 \end{aligned}$$

Property A implies $E_k \leq E_j$ for all $k \geq j$, so let J be the largest integer in the range $F \leq J \leq n$ such that $E_j \geq E_k$ for all $j > J$. In other words SE_{M_j} is the last computed partial sum with the exponent E_j . This enables us to bound $|S_j|$ by the partial sum:

$$|S_j| SE_{M_j} \leq \begin{cases} 2^{m_{j+1}-i} & \text{for } F \leq i \leq J \\ 2^{m_{j+1}-i} & \text{for } J+1 \leq i \leq n \end{cases} \quad (10)$$

We consider the cases $J \leq K$ and $K < J$ separately.

8.1.1 Case $J \leq K$

In this case, we have $1 \leq J \leq K \leq n$. The addition of m_{j+1} through m_n resulting in $SE_{M_{j+1}}$ through SE_{M_n} can bring a maximum roundoff error of half an ulp in each of $SE_{M_{j+1}}$ through SE_{M_n} which is at most $2^{m_{j+1}-i}$ each. If $K \geq J+1$, then addition of m_{j+1} causes no roundoff since $SE_{M_{j+1}}$ is computed by exact cancellation. Addition of m_{j+2} through m_n to the partial sum $SE_{M_{j+1}}$ through $SE_{M_{j+1}}$, resulting in the partial sum $SE_{M_{j+2}}$ through $SE_{M_{j+2}}$, also causes no roundoff since all the numbers involved occupy the same F level. Finally, the addition of m_{j+1} through m_n can cause roundoff errors at most $2^{m_{j+1}-i}$ each. Thus we have the roundoff error bounds:

$$|S_j| \leq \begin{cases} 2^{m_{j+1}-i} & \text{for } F+1 \leq i \leq J \\ 2^{m_{j+1}-i} & \text{for } J+1 \leq i \leq K \\ 2^{m_{j+1}-i} & \text{for } K+1 \leq i \leq n \end{cases} \quad (11)$$

Thus we can bound the total roundoff error

$$\begin{aligned}
 |SE_{M_n} - S| &\leq \sum_{i=1}^n |a_i| \\
 &\leq (J-F)2^{m_{j+1}-i} + (n-K)2^{m_{j+1}-i} \\
 &= (J-F)2^{m_{j+1}-i} + (n-K)2^{m_{j+1}-i} \\
 &= 2^{m_{j+1}-i} N_{j+1}(J, K, n). \quad (12)
 \end{aligned}$$

19

Pencil and Paper Proof

where

$$|SE_{M,f}| = \sum_{k=1}^n (|s_{k1}| + |s_{k2}| + \dots + |s_{kn}|) = \sum_{k=1}^n |s_k|$$

We now bound $|SE_{M,f}|$ from below by noting that $|SE_{M,f}| \geq 2^n$ and using the triangle inequality:

$$\begin{aligned} |SE_{M,f}| &\geq \sum_{k=1}^n |s_k| \geq \sum_{k=1}^n (|s_{k1}| + |s_{k2}| + \dots + |s_{kn}|) \\ &\geq \sum_{k=1}^n (2^{n-1} - (K-2)2^{n-2}) = (n-1)2^{n-2} \end{aligned} \quad (13)$$

where $D_{K,2^{n-2}}(J,K,n) = 1 - (K-2)2^{n-2} = (n-1)2^{n-2}$.

The relative error is thus bounded by

$$\frac{|SE_{M,f}|}{D_{K,2^{n-2}}(J,K,n)} \leq \frac{1}{1 - (K-2)2^{n-2}} \quad (14)$$

Note that $f = J < K$ cannot occur since we assume that $E_{1,2,3} < E_2 < 1$ and $SE_{M,f}$ is computed where f is a sum of n multiplicands, contradicting our choice of J . Hence we may have either $f = J = K$ or $f < J < K$, and the worst case relative error is bounded by the maximum of $D_{K,2^{n-2}}(J,K,n)$ over the domain $\mathcal{F} = \{(J,K,n) \mid 1 \leq f = J = K \leq n$ or $1 \leq f < J < K \leq n\}$.

We consider the two cases $f = J = K$ and $f < J < K$ separately.

R.1.1.1 *Case $f = J = K$.* We first note that the determinant $D_{K,2^{n-2}}(J,K,n)$ becomes $D_{K,2^{n-2}}(J,K,n) = 1 - (K-2)2^{n-2}$.

Since $n - J \leq n$, we can use bound (7) to get

$$D_{K,2^{n-2}}(J,K,n) \geq 1 - (K-2)2^{n-2} \geq \frac{2^n + 2^{n-2}}{1 - 2^{n-2}}$$

Thus $n \leq n - 1$ implies that the denominator is positive.

If $n - J \leq n - 1$ (implied by $n \leq n$), then

$$|SE_{M,f}| \leq \frac{|SE_{M,f}|}{1 - (K-2)2^{n-2}}$$

20



Pencil and Paper Proof

$$\frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} \quad (15)$$

If $n - f = n$ (implying $n = n + 1$), then

$$|SE_{K,2^{n-2}}(J,K,n)| \leq \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

To bound the last line in the above inequality, we consider the cases $f = J = 1$ and $f = J \geq 2$ separately. If $f = J = 1$, then $n - f = n$, and we

$$|SE_{K,2^{n-2}}(J,K,n)| \leq \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

If $f = J \geq 2$, then

$$|SE_{K,2^{n-2}}(J,K,n)| \leq \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

$$= \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}} = \frac{2^{n-1} - 2^{n-2} - 1}{1 - 2^{n-2}}$$

Hence in either case, $|SE_{K,2^{n-2}}(J,K,n)| \leq 3 \cdot 2^{n-2}$.

R.1.1.2 *Case $f < J < K$.* We assume $R_{K,2^{n-2}}(J,K,n)$ follows. First, we need to confirm that the determinant $D_{K,2^{n-2}}(J,K,n)$ remains positive over the range of parameters, so that $|SE_{K,2^{n-2}}(J,K,n)|$ is bounded. Thus we compute the determinant of $D_{K,2^{n-2}}(J,K,n)$ with respect to J and K in order to find the maximum.

21



Benefits

Finding bugs in proof

- typos
- missing cases
- missing side-conditions
- large versus strict inequalities
- ...
- 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
- Completely new proofs



Challenges

Pencil and Paper Proof + Formal Proof

Single Program \rightsquigarrow Library

Proof Checking \rightsquigarrow Computer Aided Proof

Pencil and Paper Proof



CYP tool: text \longrightarrow Coq

Colouring Proof

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

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 subtraction, $e(p - q) = \min(e(p), e(q))$, so $e(p - q) \geq -E(\text{bound})$ 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 $\text{bounded}(q)$, we have $0 \leq m(p - q) < M(\text{bound})$. Similarly if $\min(e(p), e(q)) = e(p)$, we can rewrite the initial equation as $0 \leq p - q \leq p$ and since $e(p - q) = e(p)$ we have $0 \leq m(p - q) \leq m(p)$. In both cases we have $0 \leq m(p - q) < M(\text{bound})$. The mantissa and the exponent are then bounded, so we have $\text{bounded}(p - q)$.

Related Works

Floating point arithmetic

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

Interval arithmetic

Melquiond (Coq)

Multiprecision arithmetic

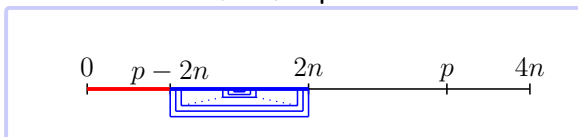
Bertot & al (Coq), Bondyfalat (Coq)

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