A case study on a numerical code in C: Log-Sum-Exp

Paul Bonnot
joint work with Raphaël Rieu-Helft (TrustInSoft)
and Claude Marché (Inria-Saclay&LMF)

November 21st, 2022

Context

- Goal: formally verify two C functions defined on floating-point numbers : LSE and MLSE.
  - Absence of overflow
  - Bounds on the rounding errors w.r.t. to pure real computations
- Results obtained:
  - Formal proof of LSE in WhyML and in C.
  - Formal proof of MLSE in WhyML and (partly) in C.

LSE function

```
double lse(double a[], size_t size) {
  int i;
  double s = 0.0;
  for (i = 0; i < size; i++) {
    s += exp_approx(a[i]);
  }
  return log_approx(s);
}
```

Let a be a vector of size n. Then

$$LSE(a) = \log \sum_{i=0}^{n-1} \exp(a_i)$$

Objective: bound the error of FP computation compared to the real computation with errors A and B depending on input size.

$$\left|\widehat{\mathrm{LSE}}(a,n) - LSE(a,n)\right| \leq A|LSE(a,n)| + B$$

Let a be a vector of n numbers and x a number between 0 and 1

$$\mathrm{MLSE}(a,n,x) = \log(n) + \frac{x^2}{\log(4)} + \sum_{i=1}^{n} -\log\left(\sum_{j=1}^{n} \exp\left(\frac{-(a_i + x - a_j)^2}{2}\right)\right)$$

We look for values A' and B' such that:

$$\left|\widehat{\mathrm{MLSE}}(a,n,x) - MLSE(a,n,x)\right| \leq A'|MLSE(a,n,x)| + B'$$

ACSL: Formal specification of C code
Example [Boldo & Marché, 2011] :

```
/*@ requires |x| ≤ 2^{-5}; */
/*@ ensures |result - cos(x)| ≤ 2^{-23}; */
float my_cos(float x) {
  /*@ assert |1 - x^2/2 - cos(x)| ≤ 2^{-24}; */
  return 1 - x * x * 0.5;
}
```

- From C to WhyML : *TrustInSoft Analyzer* with the *J^3* plug-in
- From WhyML to SMT : *Why3*
- Solvers :
  - Generalistic SMT solvers: *Alt-Ergo*, *CVC4*, *CVC5*
  - Specialized solver: *Gappa*, for rounding errors, for absence of overflows
  - Specialized solver: *DReal*, for inequalities over the reals

The approach used

1. Analysis of the summation of vectors
    1.1 On paper
    1.2 Proof formalization in WhyML
2. LSE analysis
    2.1 On paper
    2.2 Proof formalization in WhyML
3. MLSE analysis
    3.1 On paper
    3.2 Proof formalization in WhyML
4. Formal proof on C code with ACSL annotations

Rounding Errors on Summation of vectors

Rounding Errors on Summation of vectors

Sum of doubles, on paper

For rounding mode *RNE* and format `double`, for all x :

$$|\,\mathrm{round}(x) - x| \leq |x|\varepsilon + \eta$$

with $\varepsilon = 2^{-53}$ and $\eta = 2^{-1075}$.

We define the sum of floating point numbers like this :

- $x \oplus y = \mathrm{round}(x + y)$
- $\bigoplus_{i=m}^{n} a_i = 0$ if $n \leq m$;
- $\bigoplus_{i=m}^{n} a_i = \left(\bigoplus_{i=m}^{n-1} a_i\right) \oplus a_n$ if $m < n$.

Better bound for addition:

$$|(x \oplus y) - (x + y)| \leq |x + y|\varepsilon$$

Sum of doubles

Result 1

Assuming $n \leq \frac{1}{2\varepsilon}$ and that for all i, $|a_i| \leq 2^{970}$:

$$\left| \bigoplus_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i \right| \leq 2\varepsilon n \sum_{i=1}^{n} |a_i|$$

Absolute error

$\sum_{i=1}^{n} |a_i|$ instead of $\left|\sum_{i}^{n} a_i\right|$

Sum of doubles

We first prove an intermediate bound by induction over n :

$$\left|\bigoplus_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i\right| \leq \sum_{i=1}^{n} |a_i|(\varepsilon n + \varepsilon^2 n^2)$$

We then use the fact that $n \leq \frac{1}{2\varepsilon}$ to prove the final bound

No overflow on inductive case : $\left(\bigoplus_{i=m}^{n-1} a_i\right) \oplus a_n \leq maxFloat$

Lemma

For all $c \geq 0$ and for all vector a such as $|a_i| \leq c$, we have:

$$nc(-1-2\varepsilon n) \leq \bigoplus_{i=1}^{n} a_i \leq nc(1+2\varepsilon n)$$

With $\varepsilon = 2^{-53}$, $n \leq 2^{51}$ and $c \leq 2^{970}$ we don't have an overflow.

Rounding Errors on Summation of vectors

WhyML formalization

WhyML formalization

```
let rec function sum_double (f:int -> double) (a b:int)
=
if (b <= a) then
  0
else
  (sum_double f a (b - 1)) ⊕ f (b - 1)
```

We use, from WhyML IEEE-float library:

- type `double`, which includes values for infinities and NaN
- $\oplus$ for addition, without check for overflow

WhyML formalization

```
constant max_val = 2^970
constant max_size = 2^51

let ghost sum_double_err (f:int -> double) (a b:int)
  variant { b - a }
  requires { 0 ≤ b - a ≤ max_size ∧
             ∀i. a ≤ i < b → |f i| ≤ max_val }
  ensures {
    |result - sum_real f a b| ≤
      (sum_real (fun i -> |f i|) a b) × (ε(b-a) + ε^2(b-a)^2)
  }
= ...
```

Formalizes the result on the bounds, with a pre-condition for absence of overflow

WhyML formalization: the proof

```
let ghost sum_double_err (f:int -> double) (a b:int)
  ...
= let s = sum_double f a (b - 1) in
  assert IH {
    |s - sum_real a (b - 1)| ≤
      (sum_real (fun i -> |f i|) a (b-1)) × ε(b-a-1)
        + ε^2(b-a-1)^2)
  };
  sum_real_bounds (-max_val) max_val f a (b-1);
  sum_real_bounds 0 max_val (fun i -> |f i|) a (b-1);
  assert s_bound { |s| ≤ 2^1023 };
  let s' = s ⊕ f (b-1) in ...
```

- Assertion IH: inductive hypothesis
- Calls to ghost function `sum_real_bounds`: put a constant bound on `sum_real`
- Assertion `s_bound` proves the absence of overflows

Error bounds on LSE

Error bounds on LSE

On paper

Contracts of floating-point versions of exp and log in WhyML

Notations: $E_{\log}$ (resp. $E_{\exp}$) relative error of $\widehat{\log}$ (resp. $\widehat{\exp}$)

```
constant exp_error:real
axiom exp_error_bounds = 2^-53 ≤exp_error≤ 2^-7 (* was 2^-40 *)
constant exp_max:real = 672   (* was 25 *)
val function exp_approx (x:double) : double
 requires { |x| ≤ exp_max }
 ensures { |result - exp(x)| ≤ exp_error × exp(x) }

constant log_error:real
constant max_size:real = 2^51   (* was 100 *)
axiom log_error_bounds = 2^-53 ≤log_error≤ 2^-2 (* was 2^-36 *)
val function log_approx (x:double) : double
 requires { 0<x ≤ 2×max_size×(1+exp_error)×exp(max_val) }
 ensures { |result - log(x)| ≤ log_error × |log(x)| }
```

Error bound on LSE

$\widehat{\mathrm{LSE}}(a,n)$: the floating-point implementation of LSE

Result 2

Assuming that $n \leq 2^{51}$ and for all i, $|a_i| \leq 672$:

$$\left|\widehat{\mathrm{LSE}}(a,n) - \mathrm{LSE}(a,n)\right|$$
$$\leq E_{\log}|\,\mathrm{LSE}(a,n)| - \log(1-(E_{\exp} + 2\varepsilon n(1+E_{\exp}))) \times (1+E_{\log})$$

Proof: combining rounding errors of float sum and these of $\widehat{\exp}$ and $\widehat{\log}$

The relative error is $E_{\log}$.

We can approximate the constant error to get

$2(E_{\exp} + 2\varepsilon n)) \times (1+E_{\exp})$

Error bounds on LSE

WhyML formalization

WhyML formalization of LSE

```
let lse (a:array double) : double
 requires { ∀i.0 ≤ i < a.length → |a[i]| ≤ 672 }
 requires { 0 < a.length ≤ 2^51 }
 ensures {
  let exact = log(sum_real (fun i -> exp(a[i])) 0 n) in
  let err = exp_error + 2ε a.length (1 + exp_error) in
  |result - exact| ≤
   log_error * |exact|+ log (1 - err) * (1 + log_error)
 }
=
let ref s = 0 in
for i = 0 to a.length - 1 do
 invariant {
   s = sum_double (fun i -> exp_approx (a[i])) 0 i }
 s <- s + exp_approx a[i]
done;
log_approx s
```

Error bound on MLSE

Error bound on MLSE

On paper

Error bound on MLSE

Reminder:

$$\mathrm{MLSE}(a,n,x) = \log(n) + \frac{x^2}{\log(4)} + \sum_{i=1}^{n} -\log\left(\sum_{j=1}^{n} \exp\left(\frac{-(a_i + x - a_j)^2}{2}\right)\right)$$

We define :

$$\mathrm{MLSE}_{||}(a,n,x) = \log(n) + \tfrac{x^2}{\log(4)} + \textstyle\sum_{i=1}^{n} \left|-\log\left(\sum_{j=1}^{n} \exp\left(\tfrac{-(a_i+x-a_j)^2}{2}\right)\right)\right|$$

Result 3

$$\begin{aligned}\left|\widehat{\mathrm{MLSE}}(a,n,x) - \mathrm{MLSE}(a,n,x)\right| \le \varepsilon\,|\mathrm{MLSE}(a,n,x)| \\ + MLSE_{||}(a,n,x) \times (2E_{\log} + 2^{-48}n) \\ -6n\log(1 - max(2^{-45}, E_{\exp}) \\ + 2E_{\log}|\log(n)| + 2^{-1072}\end{aligned}$$

Proving C code

Difficulties going from WhyML code to C code

J^3 is still a prototype under development:

- No ghost functions
  - Consequence: manual applications of lemmas in Why3 IDE
- Encoding layer between C and WhyML
  - Consequence: additionnal transformations need to be performed on Why3 IDE before applying lemmas

We lose in automation

Reusing Why3 theories

J^3 `attributes` to map an existing Why3 symbol to an ACSL symbol

```
/*@ axiomatic SumFloat __attribute__
                  ((j3_theory ("sum_double.Sum"))) {
  logic double sum_float(double f[], integer a,
    integer b) __attribute__((j3_symbol("Sum.sum")));
} /*
```

We can do part of the proof in Why3

Not working currently

Conclusions

- Verification of floating-point summation in WhyML with the bound:

$$\left|\bigoplus_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i\right| \leq 2\varepsilon n \sum_{i=1}^{n} |a_i|$$

- Verification of LSE function in WhyML and in C with the bound :

$$\begin{aligned}&\left|\widehat{\mathrm{LSE}}(a,n) - \mathrm{LSE}(a,n)\right| \\ &\leq E_{\log} |\,\mathrm{LSE}(a,n)| - \log(1 - (E_{\exp} + 2\varepsilon n(1 + E_{\exp}))) \times (1 + E_{\log})\end{aligned}$$

- Verification of MLSE function in WhyML with the bound :

$$\begin{aligned}\left|\widehat{\mathrm{MLSE}}(a,n,x) - \mathrm{MLSE}(a,n,x)\right| \leq \varepsilon\,&|\mathrm{MLSE}(a,n,x)| \\ &+ MLSE_{||}(a,n,x) \times (2E_{\log} + 2^{-48} n) \\ &-6n \log(1 - max(2^{-45}, E_{\exp}) \\ &+ 2E_{\log} |\log(n)| + 2^{-1072}\end{aligned}$$

Future work

- Technical details :
  - Proof of C versions with J^3 by supporting attributes
  - Support for anonymous functions in TIS-kernel and J^3
- Better automation for rounding errors combination ?
- Better handling of higher order in MLSE ?