On Finding program input values maximizing the round-off error.

Mohammed Said Belaid$^1$, Claude Michel$^2$, Yahia Lebbah$^3$, Michel Rueher$^2$

$^1$Université des sciences et de la technologie d’Oran,

$^2$University of Nice--Sophia Antipolis, I3S/CNRS,

$^3$University of Oran, LITIO Lab.

5 September 2016
1 Introduction

2 Our approach

3 Motivating example

4 A greedy algorithm to maximize round off error

5 Optimizing operation round off error

6 Example

7 Conclusion
Introduction

Many programs over the floats are written with the semantic of reals in mind
  
  - e.g., timer: \( t = t + 0.1; \) each tenth of second
    - 0.1 cannot be exactly represented as a binary float
    - after a while, \( t \) absorbs 0.1

Arithmetic over \( \mathcal{F} \neq \text{arithmetic over } \mathcal{R} \)
  
  - for \( x \in \mathcal{F} \) and \( y \in \mathcal{F}, \) \( x + y \notin \mathcal{F} \) ⇔ requires rounding
  - rounding ⇔ loss of accuracy: \( o(x) \neq x \)
  - rounding occurs on each operation: \( o(o(x + y) + z) \)
  - rounding accumulation ⇔ might result in a huge difference between \( \mathcal{R} \) and \( \mathcal{F} \)

Tools to analyse computations over \( \mathcal{F} \) are required
  
  - CESTAC method: detect instability by randomly changing the rounding direction
  - FLUCTUAT: abstract interpreter with error domains
Our approach

Goal : finding input values that maximize the distance between $\mathcal{R}$ and $\mathcal{F}$

\[
\text{Obj : } \max | f_\mathcal{F}(x_1, \ldots, x_n) - f_\mathcal{R}(x_1, \ldots, x_n) | \quad \text{with } x_i \in \mathcal{F}
\]

- i.e., get input values that maximize the round off error
- which is a \textit{hard} global optimization problem

Our approach : a local search based on a greedy algorithm

- based on a maximization/minimization of basic operations, e.g. $\max | (x \oplus y) - (x + y) |$
- use heuristics to propagate optimization directions
- assume that a good approximation of the max round off error could be found knowing local basic operation optima
Motivating example

\[ z = x^2 \ominus y^2 \text{ with } x, y \in [0, 2^{24}] \]

(simple floats and rounding to the nearest even)
Motivating example

\[ z = x^2 \ominus y^2 \text{ with } x, y \in [0, 2^{24}] \]
Motivating example

\[ z = x^2 \ominus y^2 \text{ with } x, y \in [0, 2^{24}] \]

\[ x = 12582913, \ y = 14205109 \]

\[ err = 16777208 \]
A greedy algorithm to maximize Round-off error

- Try to compute a good round-off error from a local maximization/minimization of basic operations

Sketch of the algorithm

- Top/down propagation of maximization/minimization directions
- Optimization direction choice based on heuristics
- Computes instances of input variables optimizing round off error of basic operations
- Computes global round off error

Main issue : optimization of basic operations
Round-off error distribution

- Round off error distribution for the subtraction (positive minifloats, rounding to $-\infty$)

- maps Sterbenz property: if $y/2 \leq x \leq 2y$ then $x \ominus y = x - y$
Max round-off error of $c \otimes x$

**Proposition**

Let $x$ be a floating-point variable, and $c$ be a floating point constant. Assume that the values of $x$ and $c$ are positive normalized numbers. Let $p$ be the number of bits of the significand and assume that the rounding mode is set to $-\infty$. The maximal rounding error in this floating point product $c \otimes x$ can be obtained when:

$$x = M_x \cdot 2^{e_x - p + 1} \text{ with } 2^{p - 1} \leq M_x \leq 2^p - 1$$

$$M_x \equiv \text{err}' \cdot \frac{1}{M'_c} \left[2^{p - n_0 - 1}\right], \ e_x = e_x,$$

where $n_0$ is the number of least significand zeros $M_c$ in the mantissa of $c$. $M'_c = M_c \cdot 2^{-n_0}$. $\frac{1}{M'_c}$ is the modular inverse of $M'_c$.

- Other propositions for minimization, $\oplus, \ominus, \otimes, \oslash$, negative floats, other rounding modes.
An example

- Rump’s polynomial:
  
  \[ z = 333.75 \times y^6 + x^2 \times (11 \times x^2 \times y^2 - y^6 - 121 \times y^4 - 2) + 5.5 \times y^8 + \frac{x}{2 \times y} \]

- With 32 bits floats, \( x \) and \( y \in [32768, 131072] \) and rounding mode = \( -\infty \):
  
  - \( x = 98304.0078125 \), \( y = 131070.9453125 \)
  
  - which gives \( z = 4.3008808415907 \times 10^{41} \) and \( err = 3.45526791 \times 10^{35} \)

- Rump’s values:
  
  - \( x = 77617 \) and \( y = 33096 \)
  
  - which gives \( err = 4.86777830487641 \times 10^{32} \)

- However Rump’s relative error is \( \approx 6.084722881 \times 10^{33} \) while ours \( \approx 8.033 \times 10^{-7} \)
Conclusion

- A fast and effective greedy algorithm to maximize the round-off error
  - Preliminary experiments shows a fast and effective algorithm
  - Computed instances provide a good absolute error

- Further works
  - Cover more basic operations
  - Improve heuristics
  - Explore the capability to handle relative error
  - Use this local search in a complete approach
Open position

An open associate professor position is open in our team (application in march 2017).

Feel free to contact CP 2016 PC chair ... (a guy called Michel Rueher).