Toward Multi-Precision, Multi-Format Numerics

David Thien  dthien@eng.ucsd.edu
Bill Zorn  billzorn@cs.uw.edu
Pavel Panchekha  pavpan@cs.utah.edu
Zachary Tatlock  ztatlock@cs.uw.edu
Computer Math is Hard

Numerous articles retracted [Altman 99, 03]

Financial regulations [Euro 98]

Market distortions [McCullough 99, Quinn 83]
Computer Math is Hard

Numerous articles retracted [Altman 99, 03]

Financial regulations [Euro 98]

Market distortions [McCullough 99, Quinn 83]

How bad is it? No one knows, but it’s not getting any better.

-- Bill Kahan (approx)
Demand for New Formats

- HPC bandwidth concerns
- ML with low precision
- Domain specific hardware
- Line between algorithm and implementation blurs
Accuracy on a 32-bit Budget

Proposed by John Gustafson

Maximize accuracy with a 32-bit representation

\[
\left( \frac{\frac{27}{10} - e}{\pi - \left( \sqrt{2} + \sqrt{3} \right)} \right)^{\frac{67}{16}}
\]
# Accuracy on a 32-bit Budget

<table>
<thead>
<tr>
<th>Precision</th>
<th>Accuracy (decimals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 754 binary32</td>
<td>4.37</td>
</tr>
</tbody>
</table>
Accuracy on a 32-bit Budget

Try it with posits

Floating-point has same density of numbers at every order of magnitude

Posts have more numbers around 1
## Accuracy on a 32-bit Budget

<table>
<thead>
<tr>
<th>Precision</th>
<th>Accuracy (decimals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 754 binary32</td>
<td>4.37</td>
</tr>
<tr>
<td>Posits</td>
<td>7.05</td>
</tr>
</tbody>
</table>
Accuracy on a 32-bit Budget

Able to get better precision with another format

What if we could switch format/precision mid-calculation

How to precisely specify computation
Introducing FPBench 1.2: Multi-Precision Multi-Format Computations
FPBench 1.2

FPCore: Input format (s-expressions)

Benchmark suite

Tools
- Exporter
- Transformer
FPCore 1.2 Syntax
\[ \sqrt{x + 1} - \sqrt{x} \]
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
FPCore 1.2

Previous syntax didn’t allow MPMF computations

Introduce rounding contexts
FPCore Rounding Contexts

\[(\text{sqrt} (\text{+ x 1}))\]
FPCore Rounding Contexts

\[
\text{(sqrt \hspace{1em} (+ x 1))}
\]

\[
(! \hspace{1em} :\text{precision binary64} \hspace{1em} (\text{sqrt} \hspace{1em} (! \hspace{1em} :\text{precision binary32} \hspace{1em} (+ x 1))))
\]
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
(FPCore (x)
  :name example
  :precision binary64
  (-
    (! :precision binary32 (sqrt (+ x 1)))
    (! :math-library gnu-libm-2.34 (sqrt x))))
Unified way to express MPMF computations

How can we actually run these
Titanic: An MPMF Laboratory
titanic.uwplse.org
Titanic: An MPMF Laboratory

Design and experiment with novel computer arithmetic formats

Python library and online tool

Online tool lets you experiment with FPCores
(FPCore
  (a b c)
  :name "NMSE p42, positive"
  :cite (hamming-1987 herbie-2015)
  :fpbench-domain textbook
  :pre
  (and (>=
    (* b b)
    (* 4 (* a c)))
    (!= a 0))
  (/ (+
    (- b)
    (sqrt (-
      (* b b)
      (* 4 (* a c)))))
    (* 2 a)))
(FPCore (a b c)
 :name "NMSE p42, positive"
 :cite (hamming-1987 herbie-2015)
 :fpbench-domain textbook
 :pre
 (and (>=
   (* b b)
   (* 4 (* a c)))
 (!= a 0))
 (/ (+
   (- b)
   (sqrt (-
     (* b b)
     (* 4 (* a c)))))
 (* 2 a)))
print(FPCore
(a b c)
:name
"NMSE p42, positive"
:cite
(hamming-1987 herbie-2015)
:fpbench-domain
textbook
:pre
(and (>=
    (* b b)
    (* 4 (* a c)))
    (= a 0))
(/ (+
   (- b)
   (sqrt (-
     (* b b)
     (* 4 (* a c))))))
(* 2 a)))
(FPCore
  (a b c)
  :name "NMSE p42, positive"
  :cite (hamming-1987 herbie-2015)
  :textbook
textbook
  :pre
  (and (>=
    (* b b)
    (* 4 (* a c)))
  (!= a 0))
(/
  (+
    (- b)
    (sqrt (-
      (* b b)
      (* 4 (* a c))))))
  (* 2 a)))
(FPCore
 (a b c)
 :name
  "NMSE p42, positive"
 :cite
 (hamming-1987 herbie-2015)
 :fpbench-domain
 textbook
 :pre
 (and (=
   (* b b)
   (* 4 (* a c)))
   (!= a 0))
 (/)
 (+
  (- b)
  (sqrt (-
    (* b b)
    (* 4 (* a c))))))
 (* 2 a)))
(FPCore
  (a b c)
  :name
  "NMSE p42, positive"
  :cite
  (hamming-1987 herbie-2015)
  :fpbench-domain
  textbook
  :pre
  (and (>=
    (b b)
    (* 4 (a c))
    (a 0))
  (/)
  (+
    (- b)
    (sqrt (-
      (b b)
      (* 4 (a c))))))
  (* 2 a))
Accuracy on a 32-bit Budget

Proposed by John Gustafson

Maximize accuracy with a 32-bit representation
Accuracy on a 32-bit Budget

(FPCore (x y)
  :name "Accuracy on a 32-bit budget"
  :pre (and (>= x 0) (>= y 0))
  (pow
    (/
      (- (/ 27 10) E)
      (- PI (+ (sqrt x) (sqrt y))))
    (/ 67 16)))
Accuracy on a 32-bit Budget

Titanic allows us to carry out MPMF computations

Also easy way to test new formats
Accuracy on a 32-bit Budget
Accuracy on a 32-bit Budget
## Accuracy on a 32-bit Budget

<table>
<thead>
<tr>
<th>Precision</th>
<th>Accuracy (decimals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 754 binary32</td>
<td>4.37</td>
</tr>
<tr>
<td>32-bit floating-point 5-bit exponent</td>
<td>6.24</td>
</tr>
<tr>
<td>Posits</td>
<td>7.05</td>
</tr>
</tbody>
</table>
Accuracy on a 32-bit Budget

Get increased accuracy with slightly different floating-point representation

What if we can mix precisions
Accuracy on a 32-bit Budget

(FPCore (x y)
  :name "Accuracy on a 32-bit budget"
  :pre (and (>= x 0) (>= y 0))
  (pow
    (/ (/
        (- (/ 27 10) E)
        (- PI (+ (sqrt x) (sqrt y))))
      (/ 67 16)))
Example: Accuracy on a 32-bit Budget

(FPCore (x y)
   :name "Accuracy on a 32-bit budget"
   :pre (and (>= x 0) (>= y 0))
   (! :precision A (pow
       ( ! :precision B (/
           (! :precision C (- (/ 27 10) E))
           (! :precision D (- PI (+ (sqrt x) (sqrt y)))))))))
   (! :precision E (/ 67 16))))
## Accuracy on a 32-bit Budget

<table>
<thead>
<tr>
<th>Representation</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Constants</th>
<th>Accuracy (decimals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 754</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>4.37</td>
</tr>
<tr>
<td>Uniform-IEEE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6.24</td>
</tr>
<tr>
<td>Uniform-posit</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7.05</td>
</tr>
<tr>
<td>Mixed-IEEE</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td><strong>7.40</strong></td>
</tr>
<tr>
<td>Mixed-posit</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7.38</td>
</tr>
</tbody>
</table>
MPMF Tools

MPMF can increase accuracy of computations

Tooling for MPMF presents its own challenges

Techniques for one precision might not work in another
Herbie 1.2: MPMF

herbie.uwplse.org
Herbie is a tool originally designed to find and fix floating-point computations

Extended to support posits

Easily extensible specification for new formats
Add 16-bit posits to Herbie

- No full libm
- Experimental number system
- Large precision accumulator (mixed-precisions?)
How does Herbie Work

Specify input computation

Establish ground truth

Rewrite with equivalence classes and taylor series

Partition into regimes
Herbie Interface

Interface requires specification of:

- Casts between format and bigfloat
- Casts between format and ordinals
- Special values (e.g. NaN)
- Operators

Optionally, users can also specify additional rewrite rules
Herbie Experiment

Wanted to test how well existing rewrite rules work for other formats

Compare Herbie’s output for several different precisions
<table>
<thead>
<tr>
<th>Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>F(x)</td>
</tr>
</tbody>
</table>

Herbie Format Adaptation
Herbie Format Adaptation

<table>
<thead>
<tr>
<th>Real</th>
<th>Impl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_A(x)$</td>
</tr>
<tr>
<td>$F(x)$</td>
<td>$f_B(x)$</td>
</tr>
<tr>
<td></td>
<td>$f_C(x)$</td>
</tr>
</tbody>
</table>

Abstract syntax to computer program
## Herbie Format Adaptation

<table>
<thead>
<tr>
<th>Real</th>
<th>Impl</th>
<th>Impl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_A(x)$</td>
<td>$f_A'(x)$</td>
</tr>
<tr>
<td>$F(x)$</td>
<td>$f_B(x)$</td>
<td>$f_B'(x)$</td>
</tr>
<tr>
<td></td>
<td>$f_C(x)$</td>
<td>$f_C'(x)$</td>
</tr>
</tbody>
</table>

Run herbie
Herbie Format Adaptation

<table>
<thead>
<tr>
<th>Real</th>
<th>Impl</th>
<th>Impl</th>
<th>Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_A(x)</td>
<td>f_A'(x)</td>
<td>F_A'(x)</td>
<td></td>
</tr>
<tr>
<td>F(x)</td>
<td>f_B(x)</td>
<td>f_B'(x)</td>
<td>F_B'(x)</td>
</tr>
<tr>
<td>f_C(x)</td>
<td>f_C'(x)</td>
<td>F_C'(x)</td>
<td></td>
</tr>
</tbody>
</table>

Convert new program to abstract program
### Herbie Format Adaptation

<table>
<thead>
<tr>
<th>Real</th>
<th>Impl</th>
<th>Impl</th>
<th>Real</th>
<th>Impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_A(x)</td>
<td>f_A'(x)</td>
<td>F_A'(x)</td>
<td>f_B'(x)</td>
<td>F_B'(x)</td>
</tr>
<tr>
<td>f_B(x)</td>
<td>f_B'(x)</td>
<td>F_B'(x)</td>
<td>f_B'(x)</td>
<td>F_B'(x)</td>
</tr>
<tr>
<td>f_C(x)</td>
<td>f_C'(x)</td>
<td>F_C'(x)</td>
<td>f_B'(x)</td>
<td>F_B'(x)</td>
</tr>
</tbody>
</table>

Convert real program to all precisions
Herbie Format Adaptation

The best program for each precision/format should be created when Herbie optimizes for that precision/format.

Herbie should take advantage of special features of a given precision/format.
What Now
Call to Action

Looking beyond MPMF

- Already started work on FPBench 1.3
- Figure out how to do tensors
Call to Action

Support MPMF in your tools

- A few tools that support the older FPBench 1.0 standard
- New standard is meant to provide all the expressibility you need
Call to Action

Join the FPBench community

- Send us your benchmarks
- Use the FPBench tools (and file issues/PRs if you see something to improve)
- Try out our FPCore format and play around with Titanic
Questions