Towards Automated Floating-point Tools for End Users

Zachary Tatlock
CoNGA 2019
Growing Community of Contributors

- David Thien
- Bill Zorn
- Nasrine Damouche
- Matthieu Martel
- Eva Darulova
- Heiko Becker
- Pavel Panchekha
- Alex Sanchez-Stern
- Chen Qiu
- Sorin Lerner
- Debasmita Lohar
- Dan Grossman
Floating Point’s Wild Success

Flexibility

vast range: $10^{-324}$ to $10^{308}$

Performance

cheap GFLOPS

Accuracy

ops w/ min error

$\lfloor x \times y \rfloor_F = \text{Round}(\lfloor x \times y \rfloor_R)$

Compute in $F$  Round to $F$  Compute in $R$
Floating Point’s Wild Success

Often floating point is close to real arithmetic

But not always!
Floating Point’s Wild Success

But not always!

Numerous articles retracted [Altman ’99, ’03]

Financial regulations [Euro ’98]

Market distortions [McCullough ’99, Quinn ’83]
While writing an MCMC sampler for a semi-parametric clustering model, I needed to calculate several likelihood ratios and posterior parameters. Each was relatively complicated, and using the naive formulas caused divide-by-zero errors, but I wasn't sure how to best rewrite the equations.
Error in Machine Learning

Harley
Montgomery
AI Researcher

Bigger, darker blocks better

\[
\frac{(\text{sig } s)^{c_p} (1 - \text{sig } s)^{c_n}}{(\text{sig } t)^{c_p} (1 - \text{sig } t)^{c_n}}, \text{ where } \text{sig } x = \frac{1}{1 + e^{-x}}
\]

Rounding error

\[
\exp \left( c_p \ln \frac{1 + e^{-t}}{1 + e^{-s}} + c_n \ln \frac{1 - \frac{1}{1 + e^{-s}}}{1 - \frac{1}{1 + e^{-t}}} \right)
\]
Rounding Error in CAD/CAM

Blake Courter
CAD Engineer

Rounding error

→
Rounding Error in CAD/CAM

The expression used for complex square root returns imprecise results for negative reals. To avoid this imprecision, the equation is rearranged not to add \( r \) to \( x \cdot r \) (which are of similar size and opposite sign).
Error in Complex Plotting

$f(z) = \frac{1}{\left(\sqrt{R(z)} - \sqrt{R(z)} + i \exp(-20z)\right)}$

Correct Output (smooth)

Floating Point (not smooth)
Existing options

- Unreliable
+ Fast Code

+ More Reliable
- Slow Code

+ Reliable
+ Fast Code
- Expert Task
Outline

Herbgrind: Finding error in large applications

Herbie: Automatically improving accuracy

FPBench: A standard format for composing tools

Titanic: A laboratory for exploring number systems
Outline

*Herbgrind: Finding error in large applications*

Herbie: Automatically improving accuracy

*FPBench: A standard format for composing tools*

Titanic: A laboratory for exploring number systems
Error in Complex Plotting

\[ f(z) = \frac{1}{\left( \sqrt{\Re(z)} - \sqrt{\Re(z)} + i \exp(-20z) \right)} \]

Correct Output (smooth)

Floating Point (not smooth)
Finding error in large programs

Multiple libraries, data types, complex memory layout

Erroneous output

Generate input values

Compute function

Interpolate and resample

Encode to colors

Generate bitmap
Finding error in large programs

Compute function

Source of error

\[
\frac{1}{\sqrt{R(z)} - \sqrt{R(z`)} + i \exp(-20z)}
\]

\[
u = e^x \sin(y) - \frac{x}{20}
\]

\[
v = \frac{1}{2} e^x \cos(y)
\]

solve \((t^4 - ut^2 - \frac{1}{4}v^2 = 0)\)

\[
(-b + \sqrt{b^2 - 4ac}) \quad b > 0
\]

Input ranges

Expression computed
Symptom of Error

cvttssd2si %xmm0, %eax
sall $16, %eax
movl %eax, %edx
movstd -48(%rbp), %xmm1
movsd .LC7(%rip), %xmm0
mulsd %xmm1, %xmm0
cvttssd2si %xmm0, %eax
sall $8, %eax
addl %eax, %edx
movstd -40(%rbp), %xmm1
movsd .LC7(%rip), %xmm0
mulsd %xmm1, %xmm0
Many operations cause rounding
Many operations cause rounding

Flow of values is complicated
Many operations cause rounding
Flow of values is complicated
Root cause is hidden across the program
Influenced by main.cpp:12 in sqrt(complex)

Compare at main.cpp:24 in run(int, int)
49% incorrect values (221878/477000)

\[ \sqrt{b^2 - 4 \cdot 1 \cdot c - b} \]

where

\[-2E-9 < b < 0.2 \quad -2E-9 < c < 0.2\]

Erroneous output

Root cause

[PLDI 18]
Herbgrind Example Output [PLDI 18]

1. **Compare** at main.cpp:24 in run(int, int)
   49% incorrect values (231878/477000)

2. **Influenced** by main.cpp:12 in sqrt(complex)

3. \[\sqrt{b^2 - 4 \cdot 1 \cdot c - b}\]

where

4. \(-2E-9 < b < 0.2\) \quad \(-2E-9 < c < 0.2\)
plotter.S

movapd  %xmm4, %xmm0
movsd   %xmm4, 16(%rsp)
subsd   .LC1(%rip), %xmm0
call    cexp
movsd   .LC2(%rip), %xmm5
pxor    %xmm2, %xmm2
movapd  %xmm5, %xmm3
movapd  %xmm5, (%rsp)
call    ___muldc3
movsd   16(%rsp), %xmm4
### Floating-point instructions

<table>
<thead>
<tr>
<th>Operation</th>
<th>Register 1</th>
<th>Register 2</th>
<th>Register 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5 ← mul $1</td>
<td>$3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$5 ← mul $5</td>
<td>$4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$4 ← mul $2</td>
<td>$2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$5 ← add $4</td>
<td>$5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$4 ← sqrt $5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$4 ← sub $4</td>
<td>$2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$5 ← add $1</td>
<td>$1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0 ← div $4</td>
<td>$5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Executing one instruction

Float 0.0 7.389

$4 \text{ sub } 7.389$
Executing one instruction

Shadow Memory

Float 0.0 7.389 7.389
Real -6.123234…e-17 7.389056… 7.389056…

Using MPFR
Herbgrind Example Output

1. **Compare** at main.cpp:24 in run(int, int)
   49% incorrect values (231878/477000)

2. **Influenced** by main.cpp:12 in sqrt(complex)

3. \[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]
   where

4. \(-2 \times 10^{-9} < b < 0.2 \quad -2 \times 10^{-9} < c < 0.2\)
Herbgrind Example Output

1. **Compare** at main.cpp:24 in run(int, int)
   49% incorrect values (231878/477000)

2. **Influenced** by main.cpp:12 in sqrt(complex)

3. \[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]

   where

4. \(-2 \times 10^{-9} < b < 0.2\) \(-2 \times 10^{-9} < c < 0.2\)
Executing one instruction

Shadow Memory

<table>
<thead>
<tr>
<th></th>
<th>Float</th>
<th>Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4$</td>
<td>0.0</td>
<td>-6.123234...e-17</td>
</tr>
</tbody>
</table>

$\text{sub}$

<table>
<thead>
<tr>
<th></th>
<th>$4$</th>
<th>$2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4$</td>
<td>7.389</td>
<td>7.389</td>
</tr>
<tr>
<td>$2$</td>
<td>7.389056...</td>
<td>7.389056...</td>
</tr>
</tbody>
</table>
Executing one instruction

$4$

$2$

7.389

7.389

rnd

$4$

$2$

7.389056...

7.389056...

sub

local error

0.0

-6.123234…e-17

Emphasize local error as key idea

Label fact that bottom left is value that is not computed by program in general
Executing one instruction

Dynamic taint analysis

<table>
<thead>
<tr>
<th>Floating Point</th>
<th>$4$</th>
<th>$4$</th>
<th>$2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td>0.0</td>
<td>7.389</td>
<td>7.389</td>
</tr>
<tr>
<td>Real</td>
<td>-6.123234…e-17</td>
<td>7.389056…</td>
<td>7.389056…</td>
</tr>
<tr>
<td>Sources</td>
<td>{ , }</td>
<td>{ }</td>
<td>{ }</td>
</tr>
</tbody>
</table>
Herbgrind Example Output

1. **Compare** at main.cpp:24 in run(int, int)
   49% incorrect values (231878/477000)

2. **Influenced** by main.cpp:12 in sqrt(complex)

3. \[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]
   where

4. \(-2 \times 10^{-9} < b < 0.2\) \(-2 \times 10^{-9} < c < 0.2\)
Influenced by main.cpp:12 in sqrt(complex)

\[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]

where

\(-2E-9 < b < 0.2 \quad -2E-9 < c < 0.2\)
Executing one instruction

\[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]

- \( b^2 \)
- \( 4 \cdot 1 \cdot c \)
- \( b \)

**Sources**

\{ , \}
\{
\}
\{ , \}

**Expr**

- \(-\)

**Shadow Memory**

- Float
  - 0
  - -6.23234…e-17

- Real
  - 7.389
  - 7.389056…

- Sources
  - \{ , \}
  - \{
  - \{ , \}

- sub

- $4$
- $2$
Herbgrind Example Output

1. **Compare** at `main.cpp:24` in `run(int, int)`
   49% incorrect values (231878/477000)

2. **Influenced** by `main.cpp:12` in `sqrt(complex)`

   \[ \sqrt{b^2 - 4 \cdot 1 \cdot c - b} \]

where

3. \[-2E-9 < b < 0.2 \]
4. \[-2E-9 < c < 0.2 \]
Influenced by main.cpp:12 in sqrt(complex)

Compare at main.cpp:24 in run(int, int)
49% incorrect values (231878/477000)

\[ \sqrt{b^2 - 4 \cdot 1 \cdot c - b} \]

where

\[-2E-9 < b < 0.2 \quad -2E-9 < c < 0.2\]
Executing one instruction

$$\sqrt{b^2 - 4 \cdot 1 \cdot c - b}$$

$$-2E-9 < b < 0.2$$  $$-2E-9 < c < 0.2$$

Sources: { }, {}
Herbgrind Example Output

1. **Compare** at main.cpp:24 in run(int, int)
   49% incorrect values (231878/477000)

2. **Influenced** by main.cpp:12 in sqrt(complex)

3. \[ \sqrt{b^2 - 4 \cdot 1 \cdot c - b} \]

where

4. \(-2E-9 < b < 0.2 \quad -2E-9 < c < 0.2\)
Herbgrind Example Output

**Compare** at main.cpp:24 in run(int, int)
49% incorrect values (231878/477000)

**Influenced** by main.cpp:12 in sqrt(complex)

\[ \sqrt{b^2 - 4 \cdot 1 \cdot c} - b \]

where

\(-2 \times 10^{-9} < b < 0.2\) \quad \(-2 \times 10^{-9} < c < 0.2\)
Welcome to the Emacs shell

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double solve_quadratic(double a, double b, double c) {
    return (-b + sqrt(b*b - 4*a*c)) / (2 * a);
}

int main(int argc, char** argv){
    double b = 1e-10;
    for (int i = 0; i < 20; i++) {
        b *= 10;
        printf("%e\n", solve_quadratic(2, b, 3));
    }
    return 0;
}
```
To be continued…

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double solve_quadratic(double a, double b, double c) {
  return (-b + sqrt(b*b - 4*a*c)) / (2 * a);
}

int main(int argc, char** argv){
  double b = 1e-10;
  for (int i = 0; i < 20; i++) {
    b *= 10;
    printf("%e\n", solve_quadratic(2, b, 3));
  }
  return 0;
}
```
Herbgrind Implementation

http://herbgrind.ucsd.edu

~ 11 KLOC in C
Herbgrind Case Studies

Many performance tricks:
- incrementalize
- bound expr size
- type analyses

1000x overhead in worst cases

Recover real abstractions:
- intercept library calls
- revert compiler bit tricks
- recognize compensation

Found confirmed bug in dihedral angle routine.

Few false positives; detected most tricks.
Outline

Herbgrind: Finding error in large applications

*Herbie: Automatically improving accuracy*

FPBench: A standard format for composing tools

Titanic: A laboratory for exploring number systems
Rounding Error in Quadratic

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

log(ULPs)
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\]
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\]
Rounding Error in Quadratic

\[-b + \sqrt{b^2 - 4ac} \over 2a\]

Overflow

If \( b \) is large, \( \lceil b^2 \rceil_F \) overflows and the whole expression returns \( \infty \).
Rounding Error in Quadratic

\[
\frac{-b + \sqrt{b^2 - 4ac}}{2a}
\]

\[\Rightarrow\]

\[
\begin{cases} 
\frac{c}{b} - \frac{b}{a} & \text{if } b \in \text{A} 
\end{cases}
\]

Pretty Accurate
Rounding Error in Quadratic

\[-b \pm \sqrt{b^2 - 4ac} \over 2a\]

\[
\begin{cases}
\frac{c}{b} - \frac{b}{a} & \text{if } b \in A \\
\frac{-b + \sqrt{b^2 - 4ac}}{2a} & \text{if } b \in B
\end{cases}
\]

**Catastrophic Cancellation**

If \( b \) is large, but \( a \) and \( c \) are small, \( b \approx \sqrt{b^2 - 4ac} \) and the difference is rounded off.
Rounding Error in Quadratic

\[-b + \frac{\sqrt{b^2 - 4ac}}{2a}\]

\[\begin{aligned}
&\frac{c}{b} - \frac{b}{a} \quad \text{if } b \in A \\
&\frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{if } b \in B \\
&\frac{-2c}{-b - \sqrt{b^2 - 4ac}} \quad \text{if } b \in C
\end{aligned}\]

Overflow again
Rounding Error in Quadratic

$$-b + \sqrt{b^2 - 4ac} \over 2a$$

$$\begin{cases} \frac{c}{b} - \frac{b}{a} & \text{if } b \in \mathbb{A} \\ \frac{-b + \sqrt{b^2 - 4ac}}{2a} & \text{if } b \in \mathbb{B} \\ \frac{2c}{-b - \sqrt{b^2 - 4ac}} & \text{if } b \in \mathbb{C} \\ -\frac{c}{b} & \text{if } b \in \mathbb{D} \end{cases}$$
Goal: Automatically improve floating point accuracy

[Distinguished Paper, PLDI 15]
Herbie Architecture

e \rightarrow \text{sample} \rightarrow \text{cands} \rightarrow \text{regimes} \rightarrow e'

\downarrow \text{focus} \downarrow \text{rewrite} \downarrow \text{simplify} \downarrow \text{series}
Herbie Architecture

e → sample → cands → regimes → e'

- focus
- rewrite
- simplify
- series
Determine ground truth

\[ X = \text{sample}(\text{domain}(e)) \]

e.g. \( X = \{1.2 \cdot 10^{-17}, -3.8 \cdot 10^{204}, 173.5, \ldots \} \)

Round(\([e]_\mathbb{R}(X)\))

\([e]_\mathbb{F}(X)\) error

e.g. \( \{13.2b, 51.7b, 1b, \ldots \} \)

Compute in \( \mathbb{F} \)

Get 64-bit prefix with MPFR.

\textbf{Subtle!} See paper.

64 random bits
Herbie Architecture

e \xrightarrow{\text{sample}} \text{cands} \xrightarrow{\text{regimes}} e'

\text{focus} \downarrow \text{rewrite} \downarrow \text{simplify} \downarrow \text{series}
Herbie Architecture

e \xrightarrow{\text{sample}} \text{cands} \xrightarrow{\text{regimes}} e'

focus \downarrow \downarrow \downarrow \downarrow \downarrow
 rewrite \downarrow \downarrow \downarrow \downarrow
 simplify \downarrow \downarrow \downarrow
 series

Note: The diagram shows the flow from e to e' through the Herbie Architecture, with the processes of sampling, regime selection, and transformation steps such as focus, rewrite, simplify, and series.
Focus: Localize error

Compute local errors
- start at leaves
- work bottom-up

Focus on biggest
- High local error = likely root cause
- Needs fixing; focus for rewrites
Herbie Architecture

e → sample → candies → regimes → e'

Focus → rewrite → simplify → series → focus
Herbie Architecture

e sample → candidates

regimes → e'

focus → rewrite → simplify → series → focus

Herbie Architecture
Rewrite: Generate Candidates

Apply rewrites to

\(-b\star\sqrt{b^2 - 4ac}\)
\[
\frac{2a}{2a}
\]

\(-x \sim 0 - x\)

\(x + y \sim \frac{x^2 - y^2}{x - y}\)

\((x - y) + z \sim x - (y - z)\)

... 180 more ...

Rule DB

difference of squares

associate

diff, algebra, etc.

 HERBIE

b

a
Rewrite: Generate Candidates

Apply rewrites to

\[-b \pm \sqrt{b^2 - 4ac} \]

Rule DB

... 180 more ...

\[-x \leadsto 0 - x\]

\[x + y \leadsto \frac{x^2}{x} - \frac{y^2}{y}\]

\[(x - y) + z \leadsto x - (y - z)\]
Herbie Architecture

e → sample → cans → regimes → e'

- Focus
- Rewrite
- Simplify
- Series
Simplify Expressions

\[
\left( \frac{(-b)^2 - (\sqrt{b^2 - 4ac})^2}{-b - \sqrt{b^2 - 4ac}} \right) / 2a
\]

\[= \left( \frac{b^2 - (\sqrt{b^2 - 4ac})^2}{-b - \sqrt{b^2 - 4ac}} \right) / 2a\]

\[= \left( \frac{b^2 - (b^2 - 4ac)}{-b - \sqrt{b^2 - 4ac}} \right) / 2a\]

\[= \left( \frac{4ac}{-b - \sqrt{b^2 - 4ac}} \right) / 2a\]

\[= \frac{2c}{-b - \sqrt{b^2 - 4ac}}\]

Tricky! [Caviness ’70]
- many possible rewrites
- “simpler” not always clear
- huge search space
- avoid undoing progress!

E-graphs [Nelson ’79]
- track equiv classes
- restrict rewrites
- select smallest AST
Herbie Architecture

\[ e \xrightarrow{\text{sample}} \text{cands} \xrightarrow{\text{regimes}} e' \]

- focus
- rewrite
- simplify
- series
Herbie Architecture

e → sample → cand → regimes → e'

- focus
- rewrite
- simplify
- series
Series Expansions

\[-b + \sqrt{b^2 - 4ac} \over 2a\]

\[= \quad -b + b\sqrt{1 - 4ac/b^2} \over 2a\]

\[\approx \quad -b + b(1 - 4ac/2b^2) \over 2a\]

\[= \quad -4ac/2b \over 2a\]

\[= \quad -\frac{c}{b}\]

\[b > 0 \rightarrow \infty\]

\[\sqrt{1 - x} \approx 1 - x/2\]

Custom series expander:
- auto expands diverse exprs
- determines # terms to take
- expand around arbitrary pt
Herbie Architecture

e \rightarrow \text{sample} \rightarrow \text{cands} \rightarrow \text{regimes} \rightarrow e' 

- focus
- rewrite
- simplify
- series
Only keep “good” candidates

Herbie Architecture

e → sample → cand → regimes → e'

focus → rewrite → simplify → series
Herbie Architecture

e → sample → cand → regimes → e'

focus
rewrite
simplify
series
Regime Inference

\[
\frac{c - b}{b} \quad \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \frac{2c}{-b - \sqrt{b^2 - 4ac}} \quad \frac{c}{b}
\]

Try each variable; keep best one
Regime Inference

\[
\begin{align*}
\frac{c - b}{b - a} & \quad \frac{-b + \sqrt{b^2 - 4ac}}{2a} & \quad \frac{2c}{-b - \sqrt{b^2 - 4ac}} & \quad \frac{c}{b}
\end{align*}
\]
Regime Inference

\[
\begin{align*}
\frac{c}{b} - \frac{b}{a} &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\
\frac{2c}{-b - \sqrt{b^2 - 4ac}} &= \frac{c}{b}
\end{align*}
\]

\[
\begin{align*}
b < -1.15E122 \quad &\quad b < 1.06E-304 \\
b < 4.62E63 \quad &\quad b > 4.62E63
\end{align*}
\]

Find branches via dynamic programming
Herbie Architecture

e → sample → cands

regimes

focus
rewrite
simplify
series

e'
Herbie Architecture

e \rightarrow \text{sample} \rightarrow \text{cands} \rightarrow \text{regimes} \rightarrow e'

\text{focus} \downarrow \text{rewrite} \downarrow \text{simplify} \downarrow \text{series}
Auto Address Classic Issues

Dramatic improvement

Accuracy of input

Accuracy of output

Average bits correct (longer is better)
Herbie Implementation

- herbie.uwplse.org
- 1000s of formulas
- Users at major labs
- Cited in papers, theses, etc.

“critical”
“fixed my case easily”
“key determinant factor”
Outline

Herbgrind: Finding error in large applications

Herbie: Automatically improving accuracy

*FPBench: A standard format for composing tools*

Titanic: A laboratory for exploring number systems
Yardsticks and Assembly Lines
Diverse Yardsticks Across Numerics

Accuracy

*absolute, relative, ulp, bound, average*

Performance

*space, runtime, analysis time*

Expressiveness (domain)

*HPC, embedded, comp geom, BLAS, libm*
Diverse Yardsticks Across Numerics

Accuracy

“It is impossible to escape the impression that people commonly use false standards of measurement -- that they seek power, success and wealth for themselves and admire them in others, and that they underestimate what is of true value in life.”

Sigmund Freud

_Civilization and Its Discontents_
Diverse Yardsticks Across Numerics

Accuracy

“It is impossible to escape the impression that people commonly use false standards of measurement -- that they seek power, success and wealth for themselves and admire them in others, and that they underestimate what is of true value in numerics.”

Sigmoid F-round

Simulation and Its Discontinuities
# Diverse Tools Across Numerics

<table>
<thead>
<tr>
<th>Test</th>
<th>Debug</th>
<th>Verify</th>
<th>Optimize</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPDebug [PLDI’11]</td>
<td>FPTuner [POPL’17]</td>
<td>STOKE [PLDI’14]</td>
<td></td>
</tr>
<tr>
<td>HOL Formal [NSV’16]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Accuracy / Performance / Domain
Diverse Tools Across Numerics

**Problem:**
- disjoint benchmarks / paper
- completely different reprs
- difficult to compare & combine

Accuracy / Performance / Domain
Measures & Fmts Across CS

Compilation (SPEC INT, EEMBC)

*compile time, run time, code size*

SAT/SMT (DIMACS, SMT-LIB)

*solver time, model size, theory support*

Synthesis (SyGuS)

*invariant synth, programming by example, etc.*

Enables independent progress on both sides of interface.
Goal: Std Numeric Tool Yardsticks + Interfaces

Formats

*core, imperative, precisions, std error defs*

Tools

*reference / baseline eval, infrastructure*

Benchmark Suites

*diverse domains, objectives, challenge catalog*

...
**Goal:** Std Numeric Tool Yardsticks + Interfaces

**Vision:**
- reproducible, fair comparisons
- lower barrier to entry for new research
- compose existing tools for new problems
- build community (regular competitions?)
Goal: Std Numeric Tool Yardsticks + Interfaces

Vision:
- reproducible, fair comparisons
- lower barrier to entry for new research
- **compose existing tools for new problems**
- build community (regular competitions?)
When we last saw our heroes…
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double solve_quadratic(double a, double b, double c) {
    return (-b + sqrt(b*b - 4*a*c)) / (2 * a);
}

int main(int argc, char** argv) {
    double b = 1e-10;
    for (int i = 0; i < 20; i++) {
        b *= 10;
        printf("%e\n", solve_quadratic(2, b, 3));
    }
    return 0;
}

Result @ demo.c:13 in main (addr 400516)
47.750810 bits average error
64.000000 bits max error
Aggregated over 20 instances
Influenced by erroneous expression:

    (FPCore (x)
     (/ (- (sqrt (- (* x x) (* (* 4.000000 2.000000) 3.000000) +000))) x) (+ 2.000000 2.000000)))
External Collab: Daisy + Herbie [FM 18]

MPI-SWS
https://github.com/malyzajko/daisy

✓ Faster Daisy

UW
https://herbie.uwplse.org/

✓ Validated Herbie

✓ Compare rewrite algorithms
Welcome to the Emacs shell

~/dagstuhl-talk $ |

(FPCore (x)
 :name "Example"
 :pre (<= 1e6 x 2e6)
 (- (/ 1 x) (/ 1 (+ x 1))))
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

http://fpbench.org
FPBench Formats: FPCore

(FPCore (u v T)
  :name "doppler1"
  :cite (darulova-kuncak-2014)
  :fpbench-domain science
  :precision binary64
  :pre (and (<= -100 u 100)
            (<= 20 v 20000)
            (<= -30 T 50))
  :rosa-ensuring 1e-12
  (let ([t1 (+ 331.4 (* 0.6 T))])
      (/ (* (- t1) v) (* (+ t1 u)
                 (+ t1 u)))))
FPBench Formats: FPCore

(FPCore (t0 w0 N)
  :name "Pendulum"
  :fpbench-domain science
  :pre (and (< -2 t0 2) (< -5 w0 5))
  :example ([N 1000])
  (let ([h 0.01]
          [L 2.0]
          [m 1.5]
          [g 9.80665]
         (while (< n N)
           ([t t0 (let ([k1w (* (/ (- g) L) (sin t))])
                        (+ t (* h k1w))])
            [w w0 (let ([k2w (* (/ (- g) L) (sin (+ t (* (/ h 2) w))))])
                          (+ w (* h k2w))])
           [n 0 (+ n 1)])
          t))))

loops

common C/Fortran ops
FPCore: Multi-precision, Multi-format

```
(FPCore (n)
  ...
  (let ((t2
    (let ((x (* i h)))
      (while (<= k 5)
        ((d0
          (! :precision binary32 2)
          (! :precision binary32 (* 2 d0)))
          (t0 x (+ t0 (/ (sin (* d0 x)) d0))))
          (k 1 (+ k 1)))
          t0))))
    (let ((s0 (sqrt (+ (* h h) (* (- t2 t1) (- t2 t1))))))
      (! :precision (float 15 113) (+ s1 s0))))
  ...
)```

**multiple precisions**

**custom formats**
FPBench Formats: FPCore

```fpcore
(let ((sr* 0.0785398163397) (sl* 0.0525398163397))
  (while (< t 1000)
    (delta_d 0.0
      (let ((delta_theta
          (let ((inv_l 0.1)
            (delta_dl (let ((c 12.34)) (* c sl)))
            (delta_dr (let ((c 12.34)) (* c sr))))
            (* (- delta_dr delta_dl) inv_l))))
      (+ x (* delta_d cosi))))
    (y 0.0
      (let ((sini
          (let ((arg
            (let ((delta_theta
                (let ((inv_l 0.1)
                  (delta_dl (let ((c 12.34)) (* c sl)))
                  (delta_dr (let ((c 12.34)) (* c sr))))
                  (* (- delta_dr delta_dl) inv_l))))
                (+ theta (* delta_theta 0.5))))
            sin))
          (+ y (* delta_d sini))))
      (theta -0.985
        (let ((delta_theta
          (let ((inv_l 0.1)
            (delta_dl (let ((c 12.34)) (* c sl)))
            (delta_dr (let ((c 12.34)) (* c sr))))
            (* (- delta_dr delta_dl) inv_l))))
          (+ theta (* delta_theta 0.5))))
        (t 0 (+ t 1))
        (j 0 (if (== j 50) 0 (+ j 1)))
        (tmp 0.0 (if (== j 50) sl tmp))
        (sl sl* (if (== j 50) sr sl))
        (sr sr* (if (== j 50) (let ((tmp sl)) tmp) sr)))))
```
FPBench Formats: FPCore

(FPCore
  (sr* sl*)
  :name "Odometry"
  :description "Compute the position of a robot from the speed of the wheels.\nInputs: Speed `sl`, `sr` of the left and right wheel, in rad/s."
  :cite (damouche-martel-chapoutot-fmics15)
  :fpbench-domain controls
  :type binary32
  :pre (and (< 0.05 sl (* 2 PI)) (< 0.05 sr (* 2 PI)))
  :example ((sr* 0.0785398163397) (sl* 0.0525398163397))
  (while (< t 100)
    (delta_d 0.0
      (let ((c 12.34)) (* c sl)))
    (delta_dr 0.0
      (let ((c 12.34)) (* c sr)))
    (delta_d (+ delta_dl delta_dr) 0.5))
    (delta_theta 0.0
      (let ((inv_l 0.1))
        (* (- delta_dr delta_dl) inv_l)))
    (theta (- theta delta_theta)))
  (t 0 (+ t 1))
  (j 0 (if (== j 50) 0 (+ j 1)))
  (tmp 0.0 (if (== j 50) sl tmp))
  (sl sl* (if (== j 50) sr sl))
  (sr sr* (if (== j 50) (let ((tmp sl)) tmp) sr)))
	(x 0.0)
	(let ((cosi (cos arg))
      (sini (sin arg))
      (y 0.0)
      (delta_d (+ delta_dl delta_dr) 0.5))
      (+ y (* delta_d sini))))

Things can get a little… verbose.
(FPImp (sr* sl*))
:cite (damouche-martel-chapoutot-fmics15)
:pre (and (< 0.05 sl (* 2 PI)) (< 0.05 sr (* 2 PI)))
:example ([sr* 0.0785398163397] [sl* 0.0525398163397])
(while (< t 1000)
 [= delta_dl (* c sl)] [= delta_dr (* c sr)]
 [= delta_d (* (+ delta_dl delta_dr) 0.5)]
 [= delta_theta (* (- delta_dr delta_dl) inv_l)]
 [= arg (+ theta (* delta_theta 0.5))] =
 [= cosi (+ (- 1 (* arg arg .5)) (* (* arg arg arg arg) .0416666666))] =
 [= x (+ x (* delta_d cosi))]
 [= sini (+ (- arg (* (* arg arg arg arg) 0.1666666666) (* (* arg arg arg arg arg) 0.0083333333)))]
 [= y (+ y (* delta_d sini))]
 [= theta (+ theta delta_theta)]
 [= t (+ t 1)]
(if
 [== j 50)   [= j 0]   [= tmp sl]
   [= sl sr] [= sr tmp]]
 [else
   [= j (+ j 1)])])
(output x y))
FPBench Tools

Filtering benchmarks

_source, features, precision_

Reference interpreters, error measures, stats

_support diff testing, ensure consistency_

Compiling FPCore to tool input formats

_currently: C, Wolfram, Z3, JavaScript, Scala, …_
Approx 100 benches from pubs

<table>
<thead>
<tr>
<th>Benchmark sources</th>
<th>Features used</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosa</td>
<td>Arithmetic</td>
<td>Textbooks</td>
</tr>
<tr>
<td></td>
<td>Temporaries</td>
<td>Mathematics</td>
</tr>
<tr>
<td>Herbie</td>
<td>Comparison</td>
<td>Controls</td>
</tr>
<tr>
<td>Salsa</td>
<td>Loops</td>
<td>Science</td>
</tr>
<tr>
<td>FPTaylor</td>
<td>Exponents</td>
<td>(unknown)</td>
</tr>
<tr>
<td></td>
<td>Trigonometry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Conditionals</td>
<td></td>
</tr>
</tbody>
</table>

From ~ 6 papers in FM, PLDI, POPL, FMICS, etc. 
*bound verifies, sound and heuristic improvers*

http://fpbench.org/benchmarks.html
Anecdotally…

Found some existing overlap

difficult to manually translate between fmts

Type system and reference impls identified bugs

typos easy, central suite improves confidence

Using as common IR between our own tools

in Herbgrind dev, easy interop with Herbie

Used in a couple courses at UW:

599 - Accurate Computing
548 - Computer Systems Architecture
FPBench
Common standards for the floating-point research community

FPBench makes it easier to compare and combine tools from the floating-point research community.

About
Benchmarks
Community
Download
Mailing list

Documentation
Tools
FPImp
Implementor Notes
Contributors

Standards
FPCore 1.1
Metadata 1.1
Measures 1.1
Older versions

Current Status
FPBench was introduced at NSV’16. Since then, the benchmark suite has grown to 111 benchmarks, several implementations have appeared, and the standards have been deepened and improved.
Average Error: \(38.2\)  
Time: \(23.7\) s  
Precision: \(50.3\)  
Internal Precision: \(64\)  

\[
\frac{1}{2} \cdot \sqrt{2 \cdot (\sqrt{re \cdot re + im \cdot im + re})} \\
\]

\[
\text{if } (im + re) \cdot 2 \leq -2.0781343418246825 \cdot 10^{+152} : \\
\frac{\sqrt{2 \cdot (im\cdot im)}}{\sqrt{(-re)-re}} \cdot \frac{1}{2} \\
\]

\[
\text{if } (im + re) \cdot 2 \leq 1.0269599942889963 \cdot 10^{-298} : \\
\frac{1}{2} \cdot \frac{\sqrt{2 \cdot |im|}}{\sqrt{re\cdot re + im \cdot im - re}} \\
\]

\[
\text{if } (im + re) \cdot 2 \leq 2.0255230297756466 \cdot 10^{+142} : \\
\frac{1}{2} \cdot \sqrt{(im + re) \cdot 2} \\
\]

\[
\text{if } (im + re) \cdot 2 \leq 1.8534788667081643 \cdot 10^{+278} : \\
\]
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

http://fpbench.org
Outline

Herbgrind: Finding error in large applications

Herbie: Automatically improving accuracy

FPBench: A standard format for composing tools

Titanic: A laboratory for exploring number systems
How to eval in arbitrary number systems?
How to eval in arbitrary number systems?

(FPCore (n)
  ...
  (let ((t2
    (let ((x (* i h)))
      (while (<= k 5)
        ((d0
           (! :precision binary32 2)
           (! :precision binary32 (* 2 d0)))
          (t0 x (+ t0 (/ (sin (* d0 x)) d0)))
          (k 1 (+ k 1)))
          t0)))
        (let ((s0 (sqrt (+ (* h h) (* (- t2 t1) (- t2 t1)))))
             (! :precision (float 15 113) (+ s1 s0)))
        ...
    (float 5 8), (posit 1 16), (fixed 2 8), ... ?
FPBench
Common standards for the floating-point research community

FPBench makes it easier to compare and combine tools from the floating-point research community.

About
- Benchmarks
- Community
- Download
- Mailing list

Documentation
- Tools
- FPImp
- Implementor Notes
- Contributors

Standards
- FPCore 1.1
- Metadata 1.1
- Measures 1.1
- Older versions

Current Status
FPBench was introduced at NSV’16. Since then, the benchmark suite has grown to 111 benchmarks, several implementations have appeared, and the standards have been deepened and improved.
Running Titanic in the 32-bit accuracy challenge

Compute the following expression, storing the result and any intermediates in only 32 bits:

\[ \left( \frac{27}{10} - e \right) \left( \frac{67}{16} \right) \left( \pi - (\sqrt{2} + \sqrt{3}) \right) \]

(FPCore ()
:name "Accuracy on a 32-bit budget"
:spec 302.882719655469549250146
(pow
  (/
    (- (/ 27 10) e)
    (- PI (+ (sqrt 2) (sqrt 3)))
    (/ 67 16)))

(link)
(FPCore ()
  :target 302.882719655469549250146
  :name "Accuracy on a 32-bit budget"
  :precision (float 8 24)
  (pow (/ (- (/ 27 10) E) (- PI (+ (sqrt 2) (sqrt 3)))) (/ 67 16)))

Titanic Evaluator

Evaluate FPCore with IEEE 754 floating-point

w:11  p:53  binary64 override

FPCore arguments:

Evaluate FPCore  permalink
Normal 32-bit "floats" get 4 digits correct

With a 5-bit exponent we get 6!
Can we do better with mixed precision?

<table>
<thead>
<tr>
<th>Format</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posits (es=1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7.05</td>
</tr>
<tr>
<td>binary32</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>4.37</td>
</tr>
<tr>
<td>IEEE (w=5)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6.24</td>
</tr>
<tr>
<td>IEEE mixed</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td><strong>7.40</strong></td>
</tr>
</tbody>
</table>

```scheme
(FPCore ()
  :name "Accuracy on a 32-bit budget"
  :spec 302.882719655469549250146
  (pow
    (/ ; A
      (+ ; B
        (/ 27 10)); C
        (- PI (
            (+ (sqrt 2) (sqrt 3)))); D
        (/ 67 16)))) ; E
)```
Can we do better with mixed precision?

(FPCore ()
:name "Accuracy on a 32-bit budget"
:spec 302.882719655469549250146
(! :precision (float 5 27) (pow
  (! :precision (float 8 24) (/
    (! :precision (float 3 30) (-
      (! :precision (float 3 29) (/ (! :precision binary32 27) (! :precision binary32 10)))
      (! :precision (float 2 30) E)))
    (! :precision (float 2 30) (-
      (! :precision (float 2 30) PI)
)

(link)
Arbitrary floating-point is “easy” to implement
- if you aren’t too concerned about speed…
- just do it literally: compute in Reals and Round!

\[
[x \ast y]_F = \text{Round}([x \ast y]_R)
\]

- FPCore lets us specify what ops / precisions we want
- many libs for real math (MPFR, Wolfram, CoRN)
- build a new number system = implement rounding!
Titanic brings together three key components:

1. Tools for parsing / interpreting FPCores
2. A universal number representation
   - Arbitrary precision values, also extensible
3. Bindings for math libraries

Implement rounding and you’re good to go!

- High-level code in Python
- From universal representation to universal representation
  - Don’t have to twiddle bits unless you want to

Bill added Posit Sinking Point over coffee yesterday :)
Outline

Herbgrind: Finding error in large applications

Herbie: Automatically improving accuracy

FPBench: A standard format for composing tools

Titanic: A laboratory for exploring number systems
New Challenge: Format Specialization

Can Herbie adapt code to new formats/precisions?
- Code originally written for float or double
- Needs to be ported to posits

Compare Herbie in float, double, posit mode
- Each mode produces best results for that precision
- Most expressions need different optimized expressions

Early Success:
- >50% of benchmarks have precision-specific optimizations
- >10b accuracy gain from precision-specific optimizations

See David to get the gritty details!
New Challenge: Format Specialization

Different versions for each precision

What Representation Optimizing for Gives the Best Result

Representations Graph:
To test Herbie’s adaptability across different number representation formats, we do the following test. First run Herbie in each representation to get what Herbie determines is the optimal program for each representation. Then convert each of those representations’ operations to each representation and check to see which has the lowest error. (e.g. convert the double and posit operations that herbie produces when optimizing for a double and posit programs into single operations and then compare each program to see which performs best.)
Each color represents the type of the operations we are running the program and each label on the x-axis represents the representation that when optimized for, produces the best result for that representation on a given test.
Note that this graph currently doesn’t take into account how much of a difference there is the accuracy, just which representation when optimized for, gives us the best result, so it can be a bit misleading as, when looking through the results individually, the cases where a different representation, when optimized for, is best, tends to have a difference of just a few bits, whereas when the representation we are running the test in is the best when optimized for, the difference is often significantly larger (not uncommon to see 20 or more bits).
Moving Forward

Support more types
  \textit{vectors, fixed point, double double}

Scriptable reproducibility
  \$\texttt{git clone} \&\& \texttt{make report}\$

Compose more cross-group tools
  \textit{Precimonius/Salsa, FPTaylor/Stoke, ???}

Establish challenge problems!
  \textit{serve as community touchstone}
Thank You!

Herbgrind: Finding error in large applications
http://herbgrind.ucsd.edu/

Herbie: Automatically improving accuracy
https://herbie.uwplse.org/

FPBench: A standard format for composing tools
http://fpbench.org/

Titanic: A laboratory for exploring number systems
http://titanic.uwplse.org/
Prove you are human:

0.1 + 0.2 = ?

0.3000000000000004

Welcome to the secret robot internet