

Towards Numerical Assistants

*Trust, Measurement, Community, and Generality
for the Numerical Workbench*



Pavel Panchekha
`pavpan@cs.utah.edu`
University of Utah



Zachary Tatlock
`ztatlock@cs.washington.edu`
University of Washington



Writing Numerical Code

Diabolical errors

- Silent, non-compositional, non-local



Subtle tradeoffs

- Performance, accuracy, determinism



Evolving landscape

- bfloat, posits, Flexpoint, AdaptivFloat, ...



Writing Numerical Code

amazon

Books numerical analysis

Hello, Sign in Account & Lists Returns & Orders Try Prime

Select your address

Best Sellers Customer Service Today's Deals New Releases Find a Gift Low prices on s

Books Advanced Search New Releases Best Sellers & More Children's Books Textbooks Textbook Rentals Best Books of the Month

1-16 of over 10,000 results for "numerical analysis"

Amazon Prime ✓prime

Eligible for Free Shipping Free Shipping by Amazon All customers get FREE Shipping on orders over \$25 shipped by Amazon

Department < Any Department Books Mathematical Analysis Mathematics Applied Mathematics Calculus Computer Science Computers & Technology Engineering & Transportation Science & Math

Avg. Customer Review 5 & Up 4 & Up 3 & Up 2 & Up 1 & Up

Book Series Dover Books on Mathematics Essentials Pragmatic Programmers International Series in Pure and Applied Mathematics

Deals Today's Deals

New Releases Last 30 days Last 90 days Coming Soon

Author

O'REILLY Explore Data Science Shop O'Reilly Media >

Data Science from Scratch: First Principles with Python Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python Data Science for Business: What You Need to Know about Data to Be a Data-informed Manager

Numerical Analysis by Richard L. Burden, J. Douglas Faires, et al. | Jan 1, 2015

Numerical Analysis. (3rd Edition) by Timothy Sauer | Nov 9, 2017

1-16 of over 10,000 results for "numerical analysis"

non-local

determinism

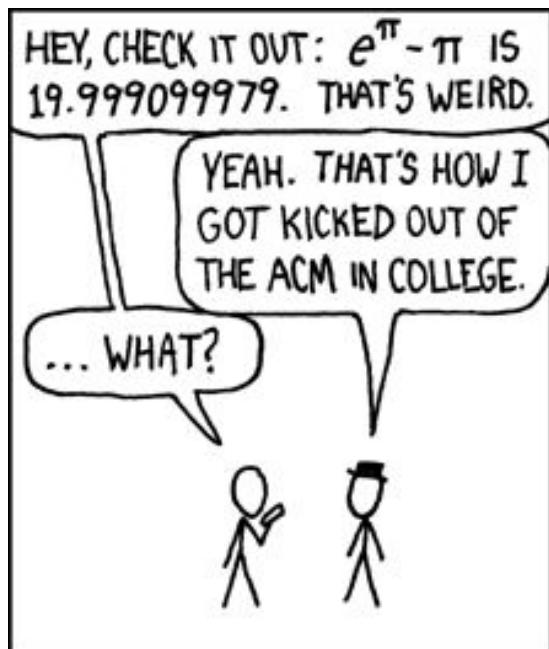
AdaptivFloat, ...

Ideal Numerical Coding

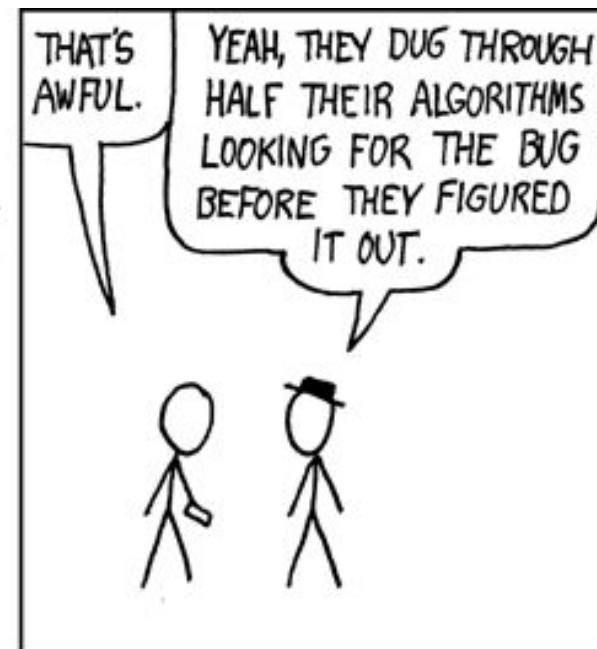
1. Look up the formula you need from Wikipedia.
2. Type formula into your program.
3. Get accurate results.

Ideal Numerical Coding

1. Look up the formula you need from Wikipedia.
2. Type formula into your program.
3. Get accurate results.

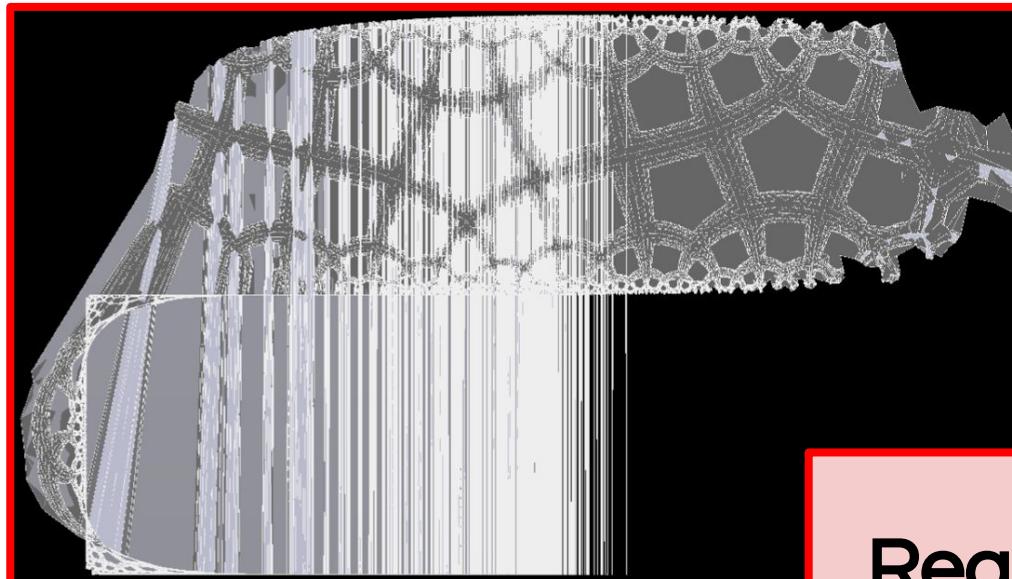


DURING A COMPETITION, I TOLD THE PROGRAMMERS ON OUR TEAM THAT $e^\pi - \pi$ WAS A STANDARD TEST OF FLOATING-POINT HANDLERS -- IT WOULD COME OUT TO 20 UNLESS THEY HAD ROUNDING ERRORS.



Ideal Numerical Coding

1. Look up the formula you need from Wikipedia.
2. Type formula into your program.
3. Get accurate results.

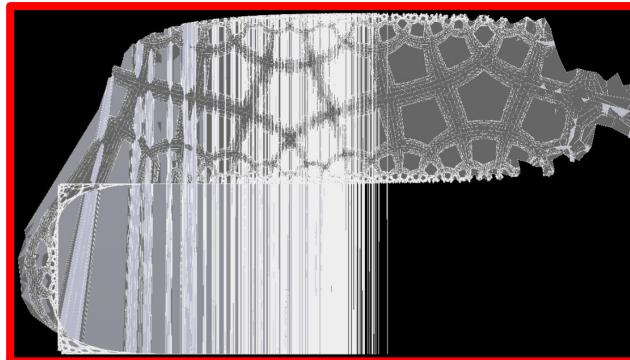


Reality.

Numerical Code in Practice



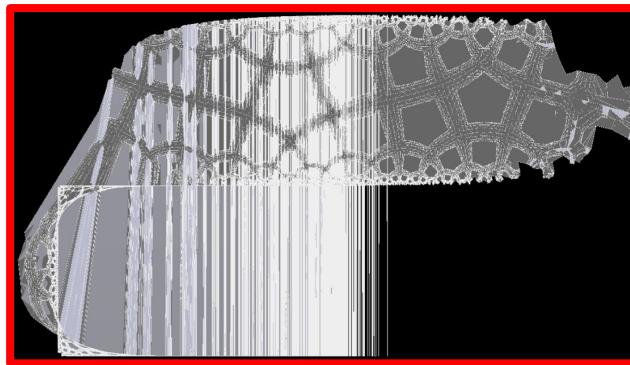
Blake Counter
CAD Researcher



Numerical Code in Practice



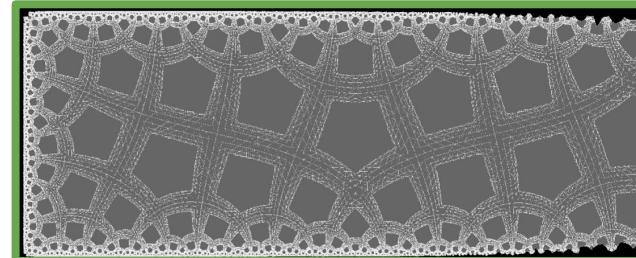
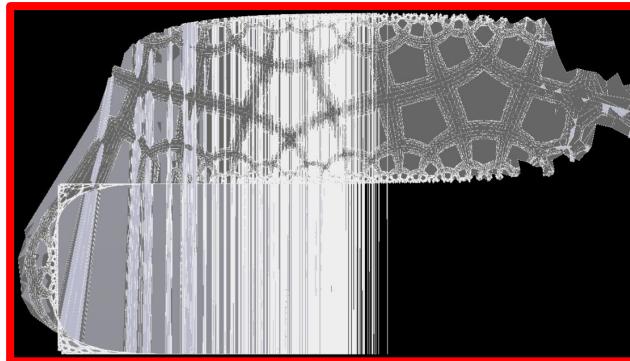
Blake Counter
CAD Researcher



Vision: Numerical Assistants



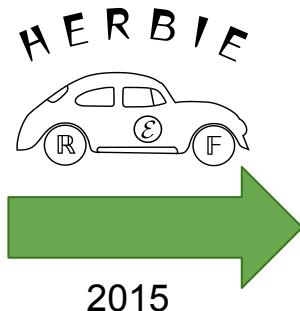
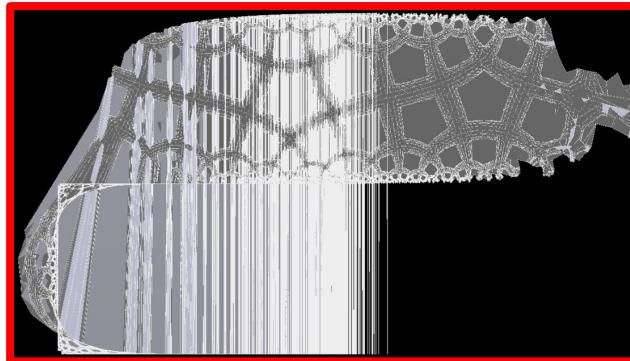
Blake Counter
CAD Researcher



Vision: Numerical Assistants



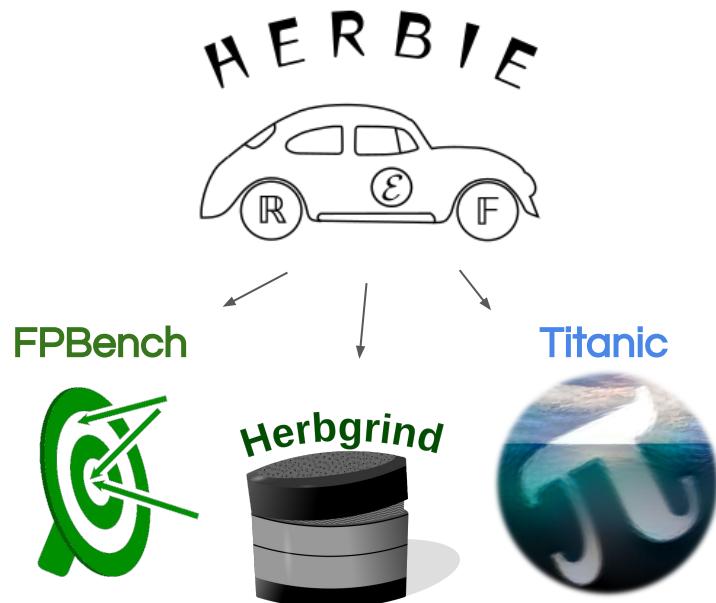
Blake Counter
CAD Researcher



Vision: Numerical Assistants



Vision: Numerical Assistants



Vision: Numerical Assistants

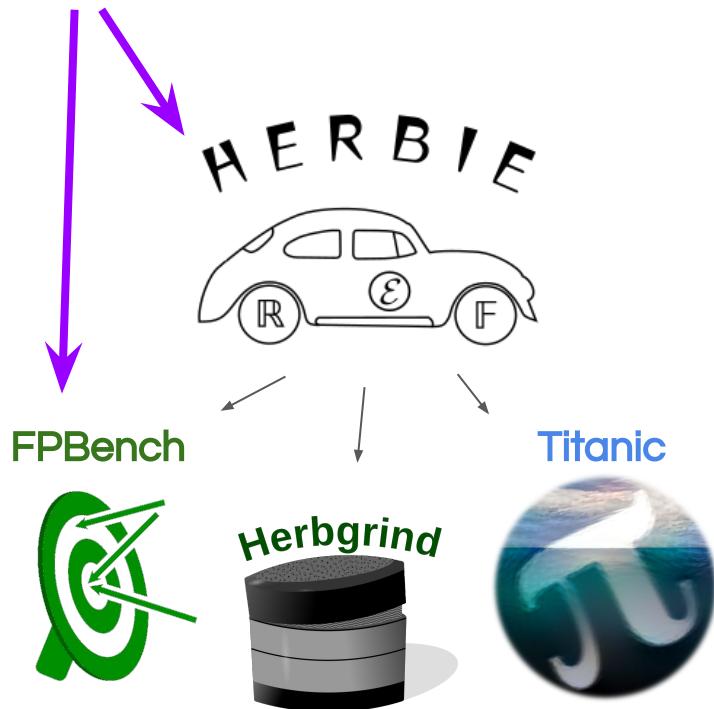


Developing since 2015:

1. Trust
2. Measurement
3. Community
4. Generality

Vision: Numerical Assistants

First, quick background



Developing since 2015:

1. Trust
2. Measurement
3. Community
4. Generality

Herbie

Math ▾

$$\frac{x - \sin x}{x - \tan x}$$



Herbie

Math

$$\frac{x - \sin x}{x - \tan x}$$

Given input expression
over the reals



Herbie

Given input expression
over the reals

$$\frac{x - \sin x}{x - \tan x}$$

Math

↓

if $x \leq -0.03256340944027832 \vee \neg(x \leq 0.027721191804569126)$:

$$\frac{x}{x - \tan x} - \frac{\sin x}{x - \tan x}$$

else :

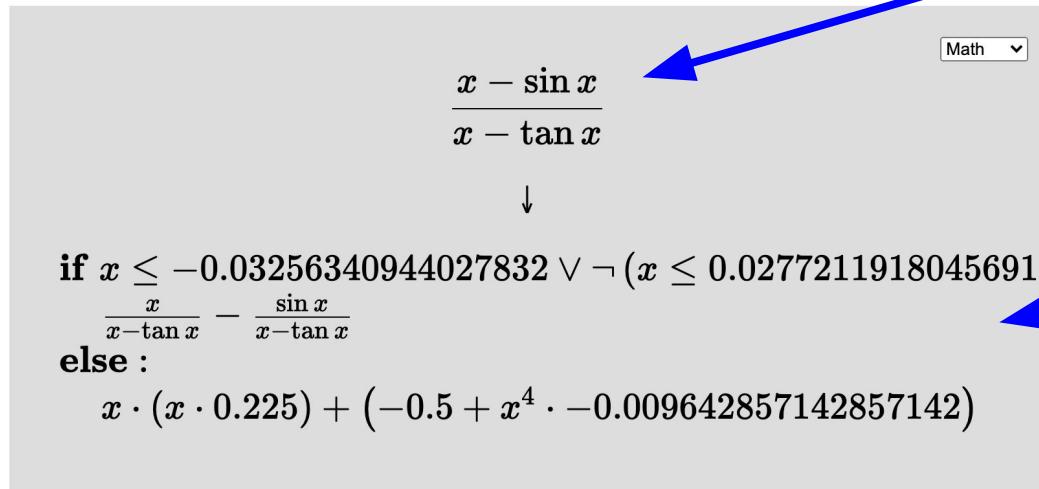
$$x \cdot (x \cdot 0.225) + (-0.5 + x^4 \cdot -0.009642857142857142)$$

Synthesize a floating
point implementation

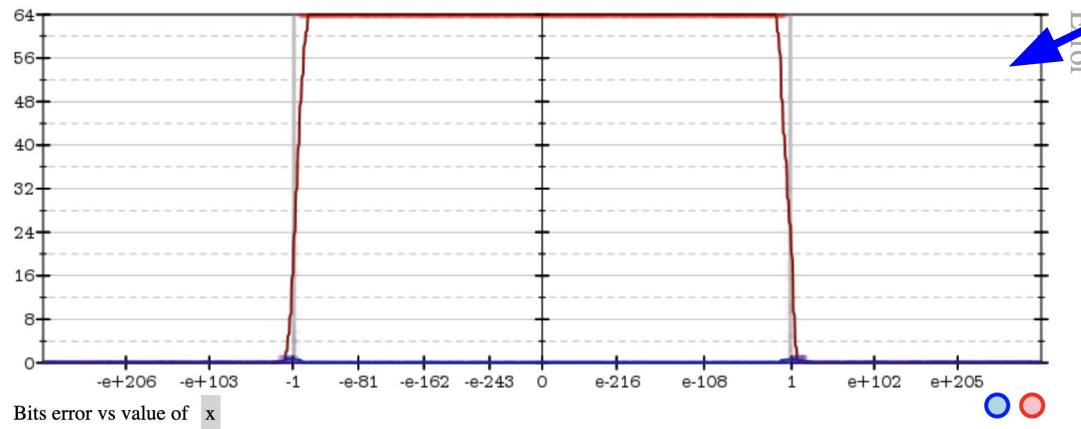


Herbie

Given input expression over the reals



Synthesize a floating point implementation



Less error than naive translation



Herbie

“The drunk numerical analyst”



edwardkmett 35 points · 4 years ago



Having applied a few dozen of these changes now I can see how `herbie` will start to fit into my workflow.

If `hlint` is like having a drunken undergraduate do a code review, `herbie` is like having a drunken numerical analyst pore over every line of code you've written offering up suggestions for how to improve stability without much considering the surrounding context. While you probably have access to the former, the latter is a much rarer talent pool.



Herbie

“The drunk numerical analyst”



edwardkmett 35 points · 4 years ago



Having applied a few dozen of these changes now I can see how `herbie` will start to fit into my workflow.

If `hlint` is like having a drunken undergraduate do a code review, `herbie` is like having a drunken numerical analyst pore over every line of code you've written offering up suggestions for how to improve stability without much considering the surrounding context. While you probably have access to the former, the latter is a much rarer talent pool.

Live demo: herbie.uwplse.org



FPBench

Benchmarks

- Mine papers, enable apples-to-apples

Compilers

- Generate C, JavaScript, Scala, TeX, ...

Standards

- Support tool interchange, composition



FPBench

Benchmarks

- Mine papers, enable apples-to-apples

Compilers

- Generate C, JavaScript, Scala, TeX, ...

Standards

- Support tool interchange, composition

Live tour: fpbench.org



1. Trust

Why should users trust Herbie?

1. Trust

Why should users trust Herbie?

Compilers forbidden from touching floating point
at all ... and they even get that wrong!

Compiler and C library bugs revealed by MPFR

- a bug in 32-bit sparc gcc 2.95.2, when a "double" is passed as last argument of a C function, which produced Bus errors. Re
- a bug in GCC on m68040-unknown-netbsd1.4.1, where DBL_MIN gives $(1-2^{(-52)})/2^{1022}$, reported by Paul Zimmerman
- [optimization bug of GCC 3.3 on Alpha with long double](#), reported by Paul Zimmermann with revision 2542 of MPFR
- [bug in LONG_MIN / 1](#) under FreeBSD (this is a bug of the C library of FreeBSD 5.20 on Alpha with GCC 3.3.3), reported
- a compiler bug found on Linux/m68k with GCC 4.0.2, reported by Vincent Lefèvre, found in revision 3945 of MPFR (file to
- [27116](#) bug of GCC 4.2 reported by [Martin Michlmayr](#) when compiling MPFR 2.2.0
- [bug of the Sun C compiler](#) under Solaris/x86, reported by Emil Miklic (affects Sun C 5.8 2005/10/13, Sun C 5.8 Patch 121)
- [36296](#) bug of GCC 4.3.0 reported by Paul Zimmermann when compiling MPFR 2.3.1 (no real bug, but false positive warning)
- [36299](#): a false warning issued by GCC 4.2+, reported by Vincent Lefèvre in revision 5585 of MPFR
- [39867](#) bug of GCC 4.4.0 reported by Philippe Théveny when testing the MPFR trunk (does not affect MPFR 2.4.1, but MPFR 2.4.1 fails)
- [40757](#) revealed when testing MPFR 2.4.1 on some Solaris machines with GCC 4.4.0 (this is in fact a bug of the Solaris memory manager)
- [bug with gcc 3.3.2 in AIX 6.1](#), where "make check" from mpfr 2.4.2 fails, but succeeds with gcc 3.3.2 in AIX 5.3 [reported by Daniel Richard G.]
- [bug with GCC 4.5.2 on powerpc-ibm-aix4.3.2.0](#) [reported by Daniel Richard G.]
- [bug with the Sun C compiler](#) with the -xO3 optimization level on sparc/Solaris, reported by Maciej Blizinski on August 3, 2005
- a bug with GCC 4.3.2 (and 4.4.1) was found while testing MPFR 3.1.0-rc1 on gcc54.fsffrance.org (UltraSparc IIe under Debian)
- a [bug](#) in the initialisation of Thread Local Storage variables in shared libraries on FreeBSD 8.2 (still present in GCC trunk as of 2011)
- [#50683](#), a [bug](#) with Thread Local Storage variables with GCC 4.6.1 on sparc64 was found with MPFR 3.1.0 on Debian [reported by Daniel Richard G.]
- [#50691](#), a bug with Thread Local Storage variables with GCC 4.4.6 on hppa-unknown-linux-gnu was found with MPFR 3.1.0 on HPUX [reported by Daniel Richard G.]

1. Trust

Why should users trust Herbie?

Compilers forbidden from touching floating point at all ... and they even get that wrong!

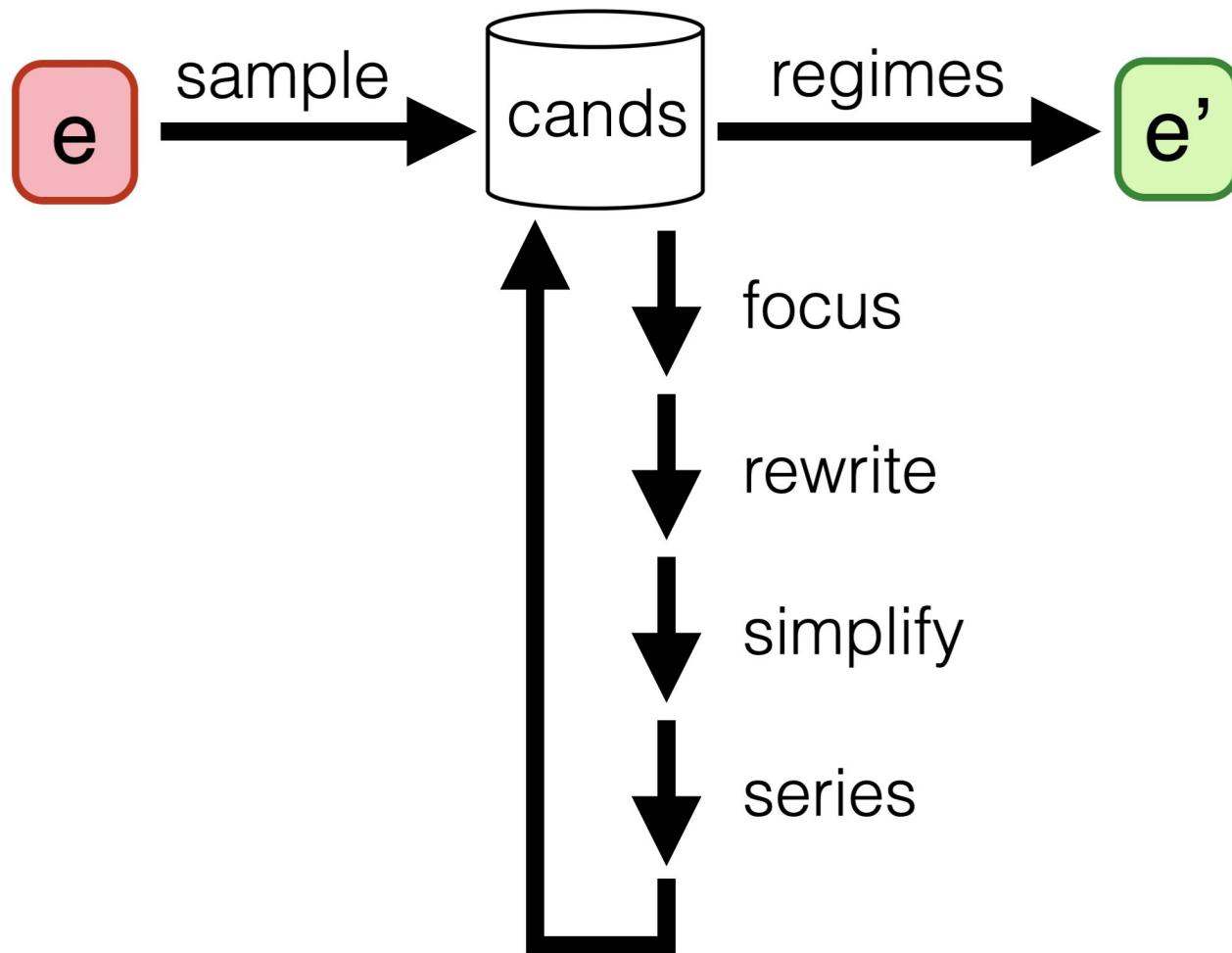
Compiler and C library bugs revealed by MPFR

- a bug in 32-bit sparc gcc 2.95.2, when a "double" is passed as last argument of a C function, which produced Bus errors. Re
- a bug in GCC on m68040-unknown-netbsd1.4.1, where DBL_MIN gives $(1-2^{(-52)})/2^{1022}$, reported by Paul Zimmerman
- [optimization bug of GCC 3.3 on Alpha with long double](#), reported by Paul Zimmermann with revision 2542 of MPFR
- [bug in LONG_MIN/1](#) under FreeBSD (this is a bug of the C library of FreeBSD 5.20 on Alpha with GCC 3.3.3), reported
- a compiler bug found on Linux/m68k with GCC 4.0.2, reported by Vincent Lefèvre, found in revision 3945 of MPFR (file to
- [27116](#) bug of GCC 4.2 reported by [Martin Michlmayr](#) when compiling MPFR 2.2.0
- [bug of the Sun C compiler](#) under Solaris/x86, reported by Emil Miklic (affects Sun C 5.8 2005/10/13, Sun C 5.8 Patch 121
- [36296](#) bug of GCC 4.3.0 reported by Paul Zimmermann when compiling MPFR 2.3.1 (no real bug, but false positive warnin
- [36299](#): a false warning issued by GCC 4.2+, reported by Vincent Lefèvre in revision 5585 of MPFR
- [39867](#) bug of GCC 4.4.0 reported by Philippe Théveny when testing the MPFR trunk (does not affect MPFR 2.4.1, but MPF
- [40757](#) revealed when testing MPFR 2.4.1 on some Solaris machines with GCC 4.4.0 (this is in fact a bug of the Solaris men
- [bug with gcc 3.3.2 in AIX 6.1](#), where "make check" from mpfr 2.4.2 fails, but succeeds with gcc 3.3.2 in AIX 5.3 [reported i
- [bug with GCC 4.5.2 on powerpc-ibm-aix4.3.2.0](#) [reported by Daniel Richard G.]
- [bug with the Sun C compiler](#) with the -xO3 optimization level on sparc/Solaris, reported by Maciej Blizinski on August 3, 2
- a bug with GCC 4.3.2 (and 4.4.1) was found while testing MPFR 3.1.0-rc1 on gcc54.fsffrance.org (UltraSparc IIe under Del
- a [bug](#) in the initialisation of Thread Local Storage variables in shared libraries on FreeBSD 8.2 (still present in GCC trunk a
- [#50683](#), a [bug](#) with Thread Local Storage variables with GCC 4.6.1 on sparc64 was found with MPFR 3.1.0 on Debian [rep
- [#50691](#), a bug with Thread Local Storage variables with GCC 4.4.6 on hppa-unknown-linux-gnu was found with MPFR 3.1

No really, why should users trust Herbie?

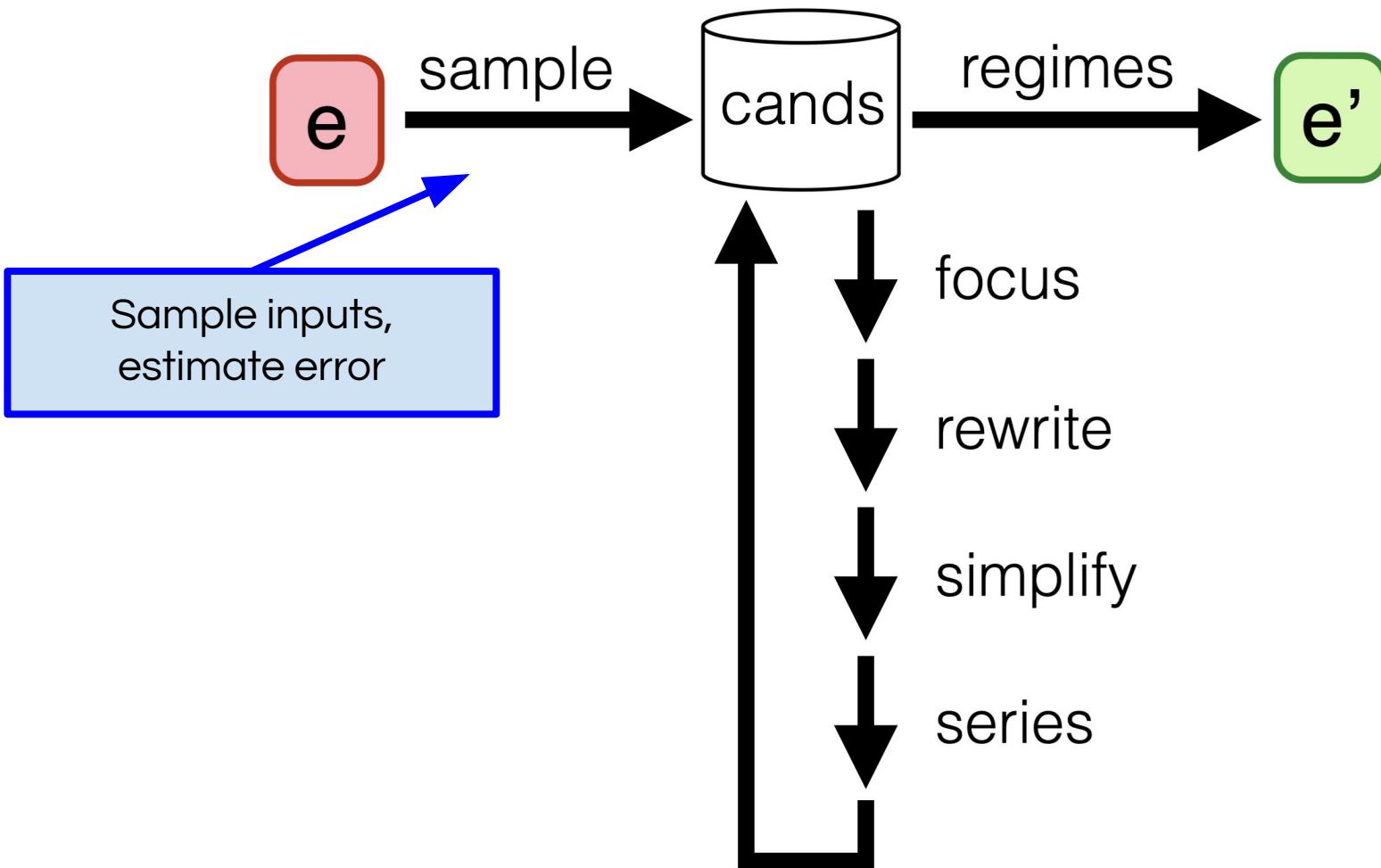
1. Trust

Why should users trust Herbie?



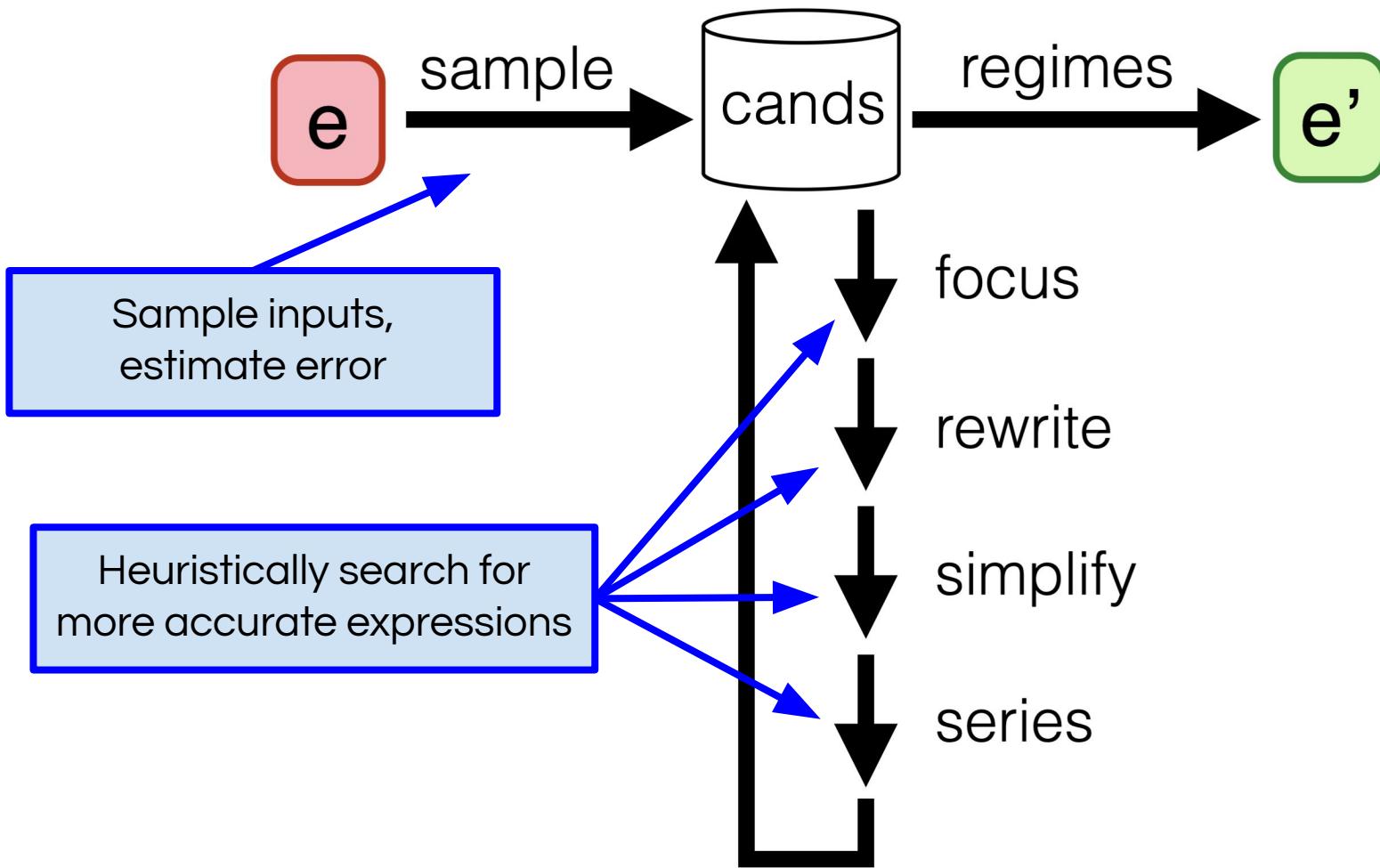
1. Trust

Why should users trust Herbie?



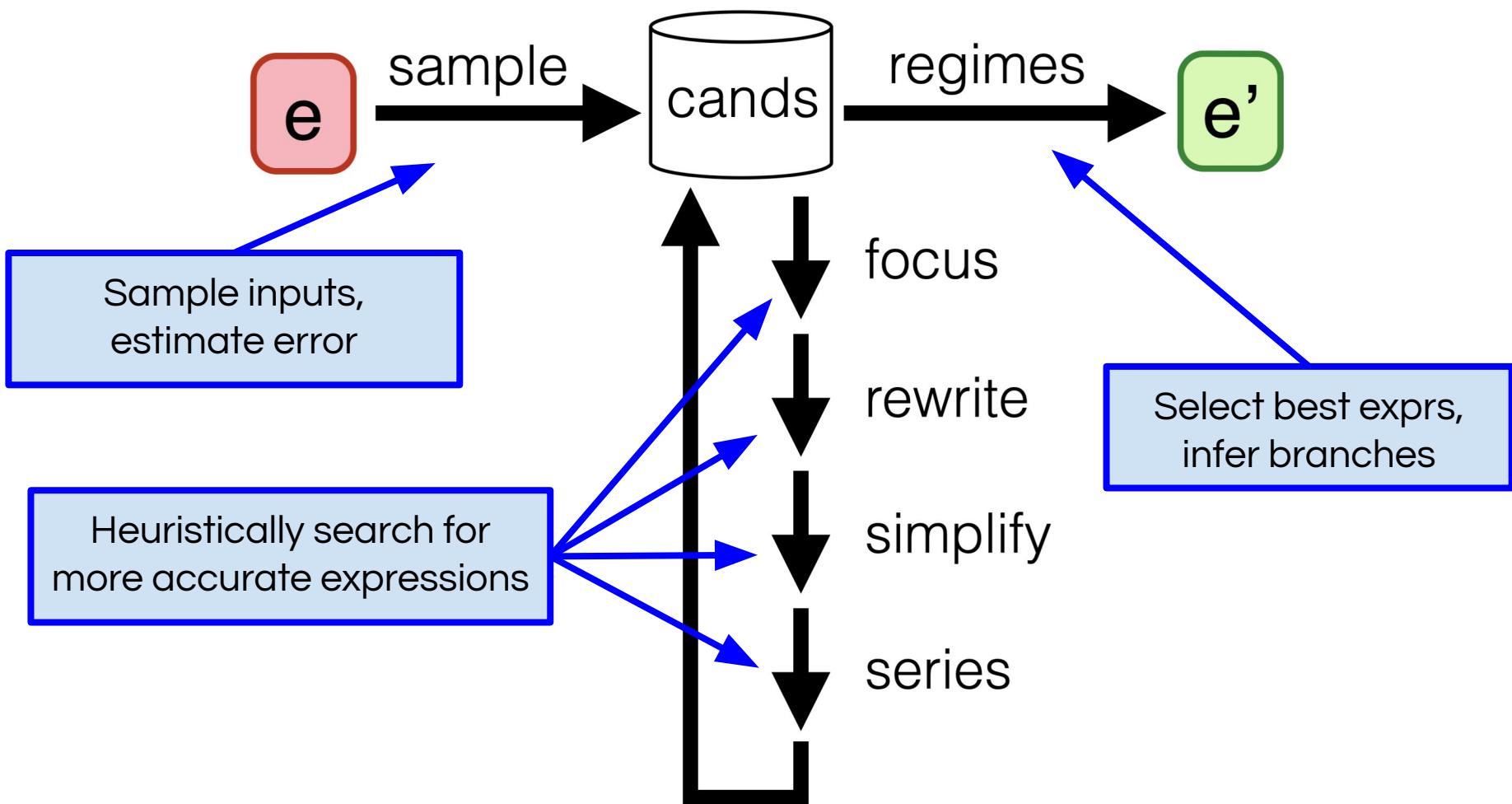
1. Trust

Why should users trust Herbie?



1. Trust

Why should users trust Herbie?



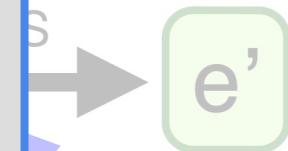
1. Trust

Why should users trust Herbie?

$$\frac{(-b) + \sqrt{b \cdot b - 4 \cdot (a \cdot c)}}{2 \cdot a}$$



```
if b <= -9.877954748696237 · 10+117 :  
    1 · (c/b - b/a)  
elif b <= 9.997495995248066 · 10-153 :  
    1  
    _____  
    ____ a  
    √b·b-4·(c·a)-b · 2  
else :  
    c/b · -1
```



Select best exprs,
infer branches

Unpredictable,
mysterious!

Sam
estim

Heur
more a

1. Trust

Why should users trust Herbie?

Average Error:

61.5 → 2.0 14.7s binary64

Time:

Precision:

Test set, not training set

1. Trust

Why should users trust Herbie?

Derivation

Split input into 3 regimes

if $b < -9.87795474869623693e117$

Initial program
51.1

$$\frac{(-b) + \sqrt{b \cdot b - 4 \cdot (a \cdot c)}}{2 \cdot a}$$

Simplified
51.1

$$\rightsquigarrow \frac{\sqrt{b \cdot b - 4 \cdot (a \cdot c)} - b}{a \cdot 2}$$

Taylor expanded
around $-\inf$
2.4

$$\rightsquigarrow 1 \cdot \frac{c}{b} - 1 \cdot \frac{b}{a}$$

Simplified
2.4

$$\rightsquigarrow 1 \cdot \left(\frac{c}{b} - \frac{b}{a} \right)$$

1. Trust

Why should users trust Herbie?

Split input into 3 regimes

`if b < -9.87795474869623693e117`

Initial program
51.1

$$\frac{(-b) + \sqrt{b \cdot b - 4 \cdot (a \cdot c)}}{2 \cdot a}$$

Simplified
51.1

$$\rightsquigarrow \frac{\sqrt{b \cdot b - 4 \cdot (a \cdot c)} - b}{a \cdot 2}$$

Taylor expanded
around -inf
2.4

$$\rightsquigarrow 1 \cdot \frac{c}{b} - 1 \cdot \frac{b}{a}$$

Simplified
2.4

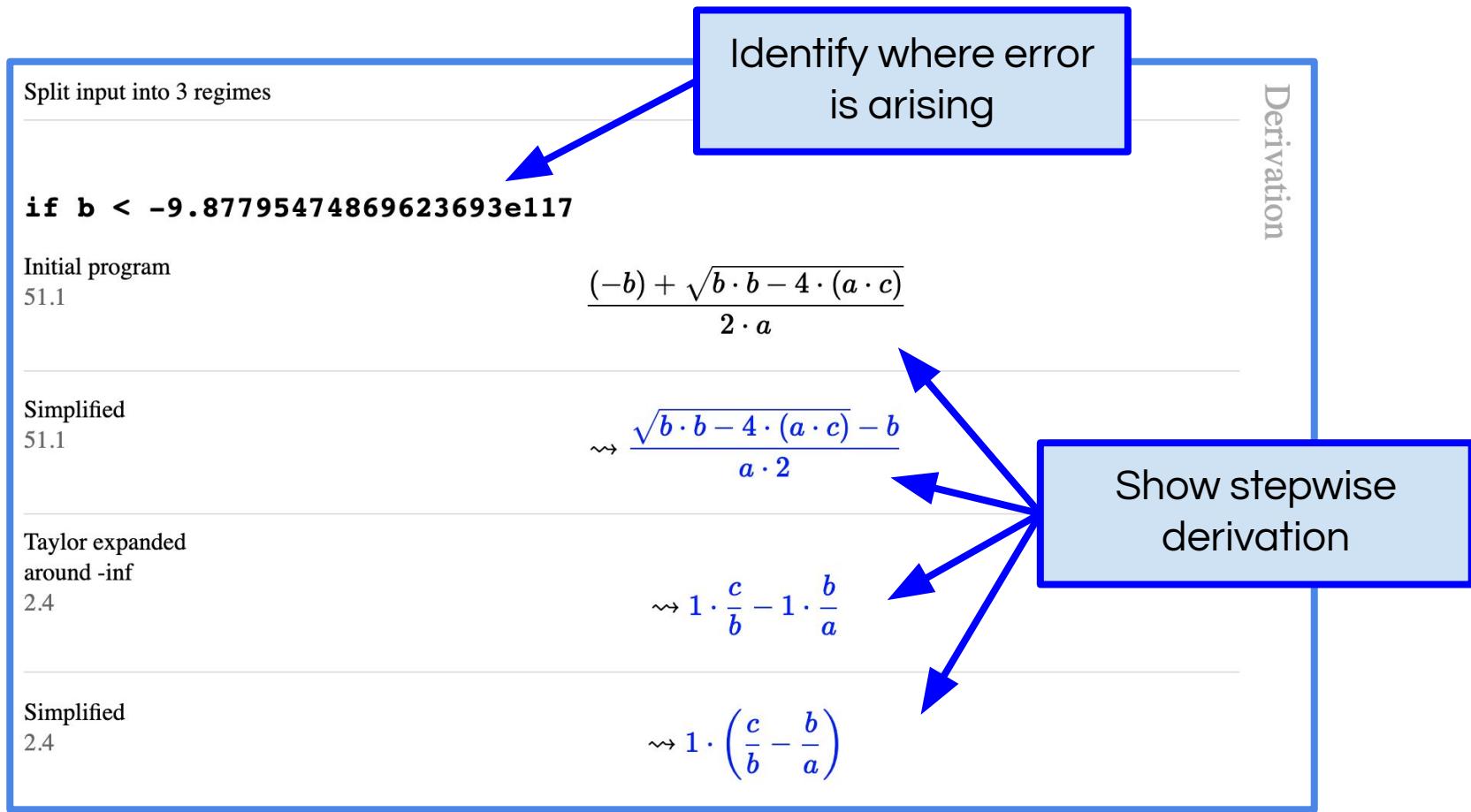
$$\rightsquigarrow 1 \cdot \left(\frac{c}{b} - \frac{b}{a} \right)$$

Identify where error
is arising

Derivation

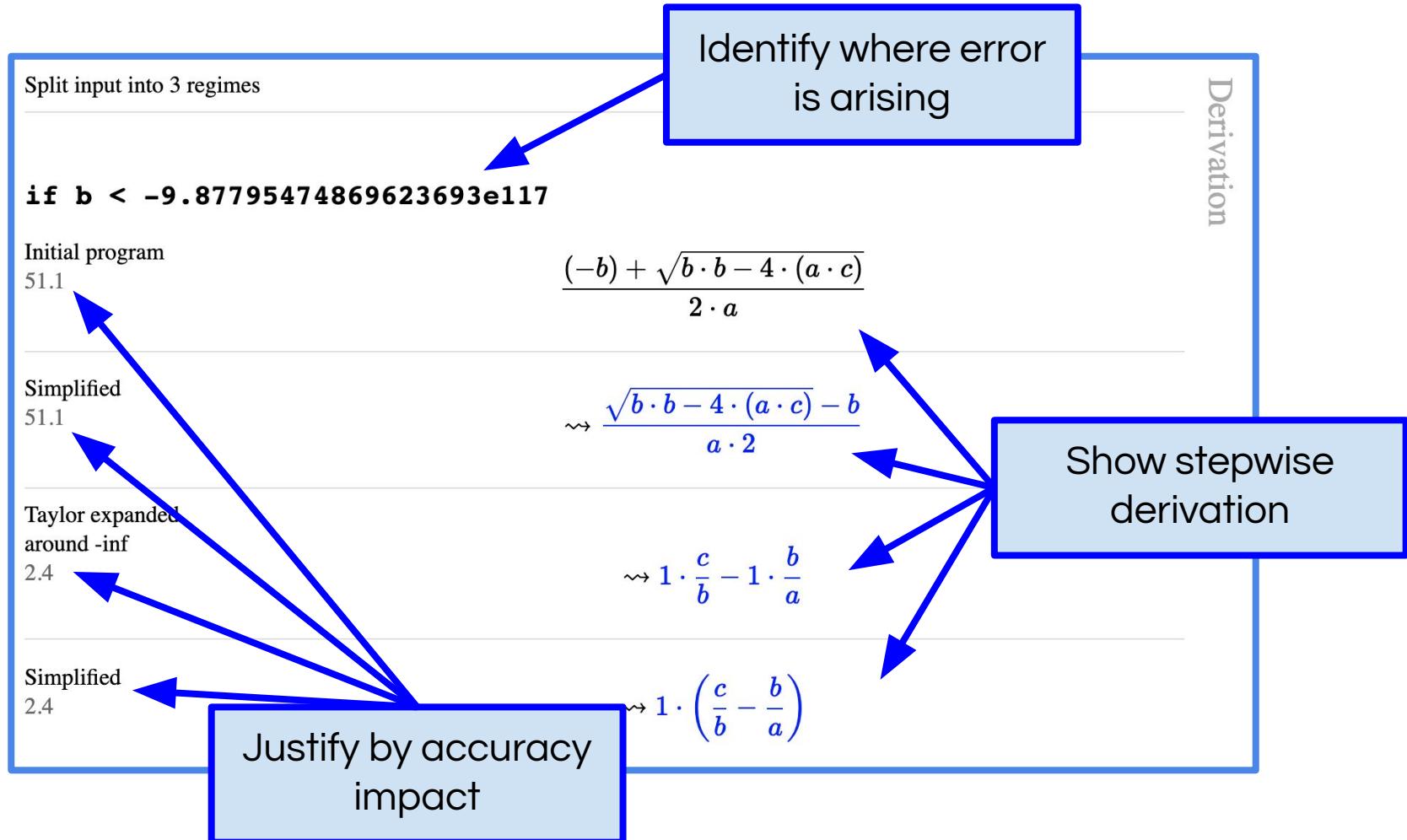
1. Trust

Why should users trust Herbie?



1. Trust

Why should users trust Herbie?



2. Measurement

That which is measured, improves.

2. Measurement

That which is measured, improves.

Herbie is a complex pipeline.

Each component individually robust.

- *Actually works against debugging!!*

Need to have per-module specs and metrics.

2. Measurement

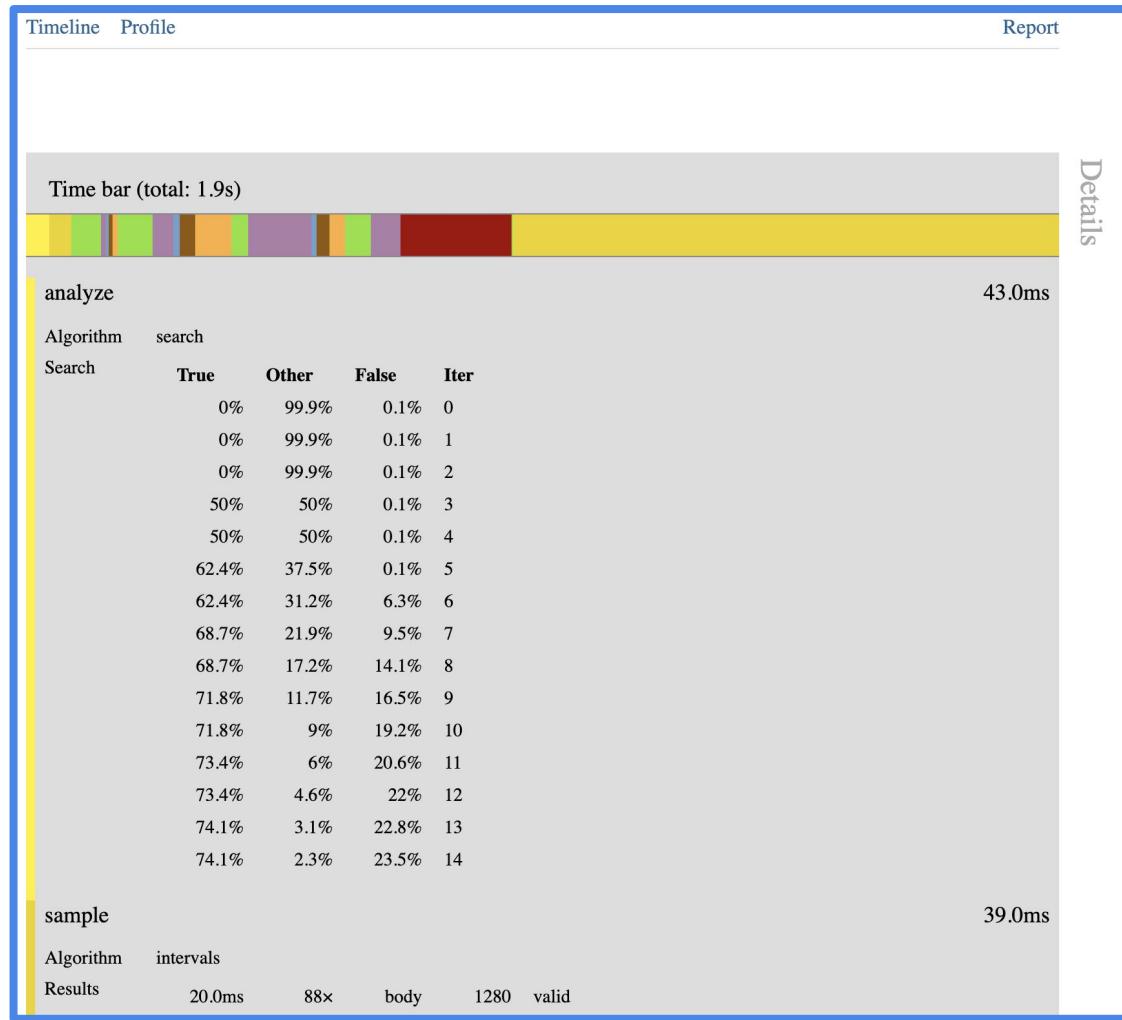
That which is measured, improves.

```
5.164 * [progress]: [Phase 1 of 3] Setting up.  
0.001 * * * [progress]: [1/2] Preparing points  
0.080 * * * [progress]: [2/2] Setting up program.  
0.083 * [progress]: [Phase 2 of 3] Improving.  
0.083 * [simplify]: Simplifying using #<procedure:simplify-batch-egg> :  
  (- (exp (* a x)) 1.0)  
0.137 * * [simplify]: iteration 0 : 4778 enodes (cost 6 )  
0.137 * * [simplify]: iteration 1 : 4778 enodes (cost 6 )  
0.137 * [simplify]: Simplified to:  
  (- (pow (exp a) x) 1.0)  
0.141 * * [progress]: iteration 1 / 4  
0.141 * * * [progress]: picking best candidate  
0.146 * * * [pick]: Picked #<alt (i» (a x) (- (exp (* a x)) 1.0))>  
0.146 * * * [progress]: localizing error  
0.151 * * * [progress]: generating rewritten candidates  
0.151 * * * * [progress]: [ 1 / 2 ] rewriting at (2)  
0.155 * * * * [progress]: [ 2 / 2 ] rewriting at (2 1)  
0.157 * * * [progress]: generating series expansions  
0.157 * * * * [progress]: [ 1 / 2 ] generating series at (2)  
0.157 * [approximate]: Taking taylor expansion of (- (exp (* a x)) 1.0) in (a x) around 0  
0.157 * [taylor]: Taking taylor expansion of (- (exp (* a x)) 1.0) in x  
0.157 * [taylor]: Taking taylor expansion of (exp (* a x)) in x  
0.157 * [taylor]: Taking taylor expansion of (* a x) in x  
0.157 * [taylor]: Taking taylor expansion of a in x  
0.157 * [taylor]: Taking taylor expansion of x in x  
0.157 * [taylor]: Taking taylor expansion of 1.0 in x  
0.157 * [taylor]: Taking taylor expansion of (- (exp (* a x)) 1.0) in a  
0.157 * [taylor]: Taking taylor expansion of (exp (* a x)) in a  
0.157 * [taylor]: Taking taylor expansion of (* a x) in a  
0.157 * [taylor]: Taking taylor expansion of a in a  
0.157 * [taylor]: Taking taylor expansion of x in a  
0.157 * [taylor]: Taking taylor expansion of 1.0 in a  
0.157 * [taylor]: Taking taylor expansion of (- (exp (* a x)) 1.0) in a  
0.157 * [taylor]: Taking taylor expansion of (exp (* a x)) in a  
0.157 * [taylor]: Taking taylor expansion of (* a x) in a  
0.157 * [taylor]: Taking taylor expansion of a in a  
0.157 * [taylor]: Taking taylor expansion of x in a  
0.157 * [taylor]: Taking taylor expansion of 1.0 in a  
0.157 * [taylor]: Taking taylor expansion of 0 in x  
0.158 * [taylor]: Taking taylor expansion of x in x  
0.158 * [taylor]: Taking taylor expansion of (* 1/2 (pow x 2)) in x  
0.158 * [taylor]: Taking taylor expansion of 1/2 in x  
0.158 * [taylor]: Taking taylor expansion of (pow x 2) in x  
0.158 * [taylor]: Taking taylor expansion of x in x  
0.158 * [taylor]: Taking taylor expansion of (* 1/6 (pow x 3)) in x  
0.158 * [taylor]: Taking taylor expansion of 1/6 in x  
0.158 * [taylor]: Taking taylor expansion of (pow x 3) in x  
0.158 * [taylor]: Taking taylor expansion of x in x  
0.159 * [taylor]: Taking taylor expansion of (* 1/24 (pow x 4)) in x  
0.159 * [taylor]: Taking taylor expansion of 1/24 in x
```

No “just enough” level of detail is right for logs...

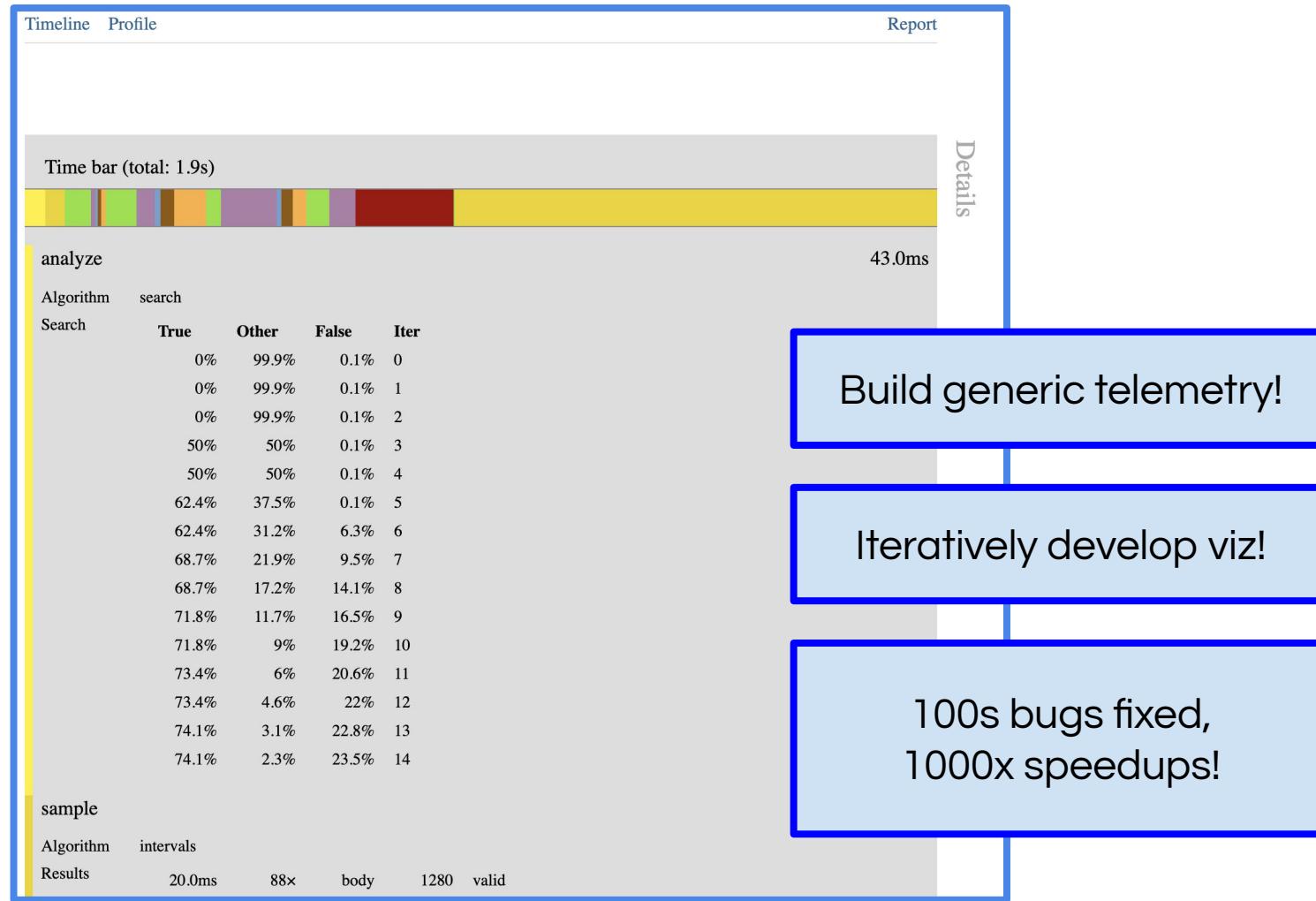
2. Measurement

That which is measured, improves.



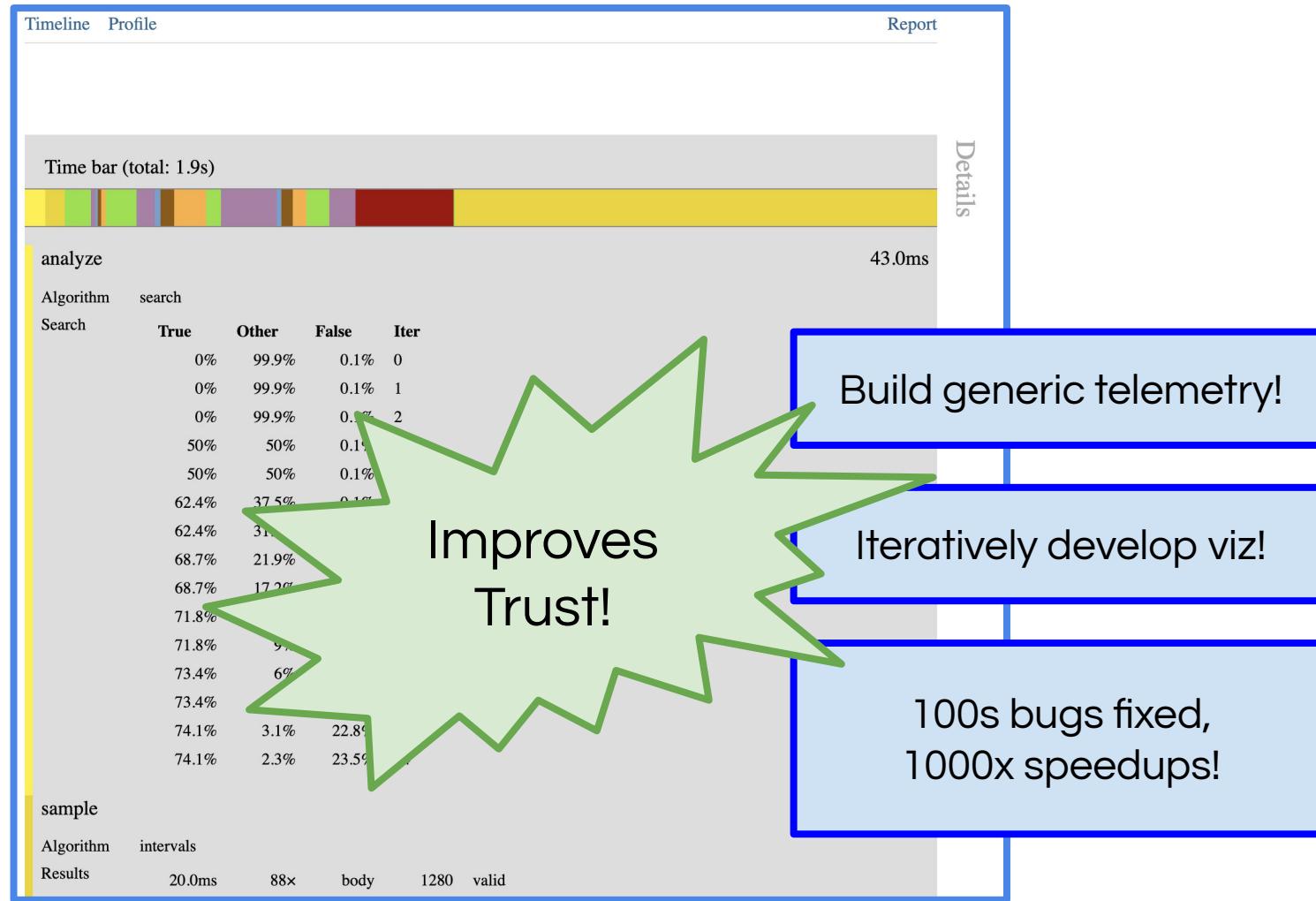
2. Measurement

That which is measured, improves.



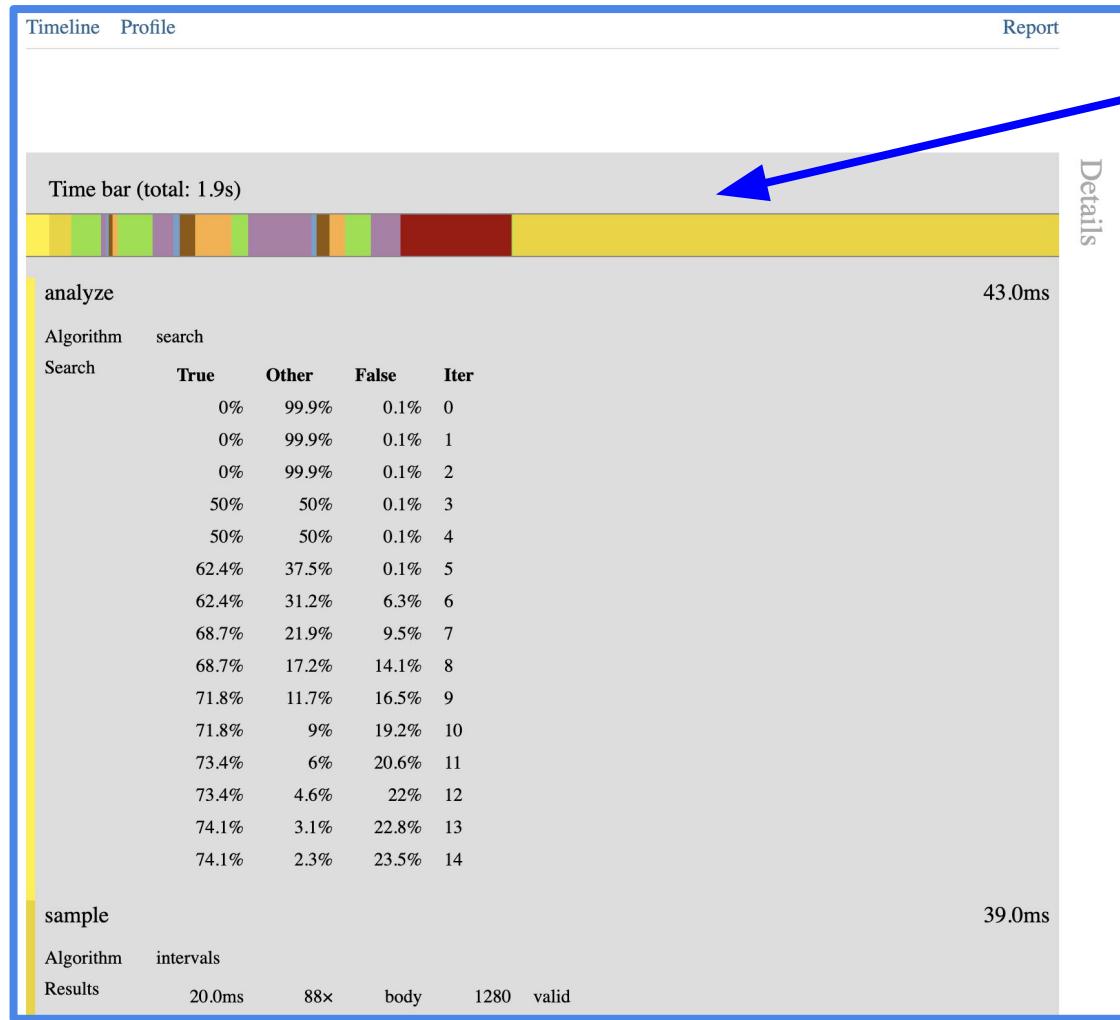
2. Measurement

That which is measured, improves.



2. Measurement

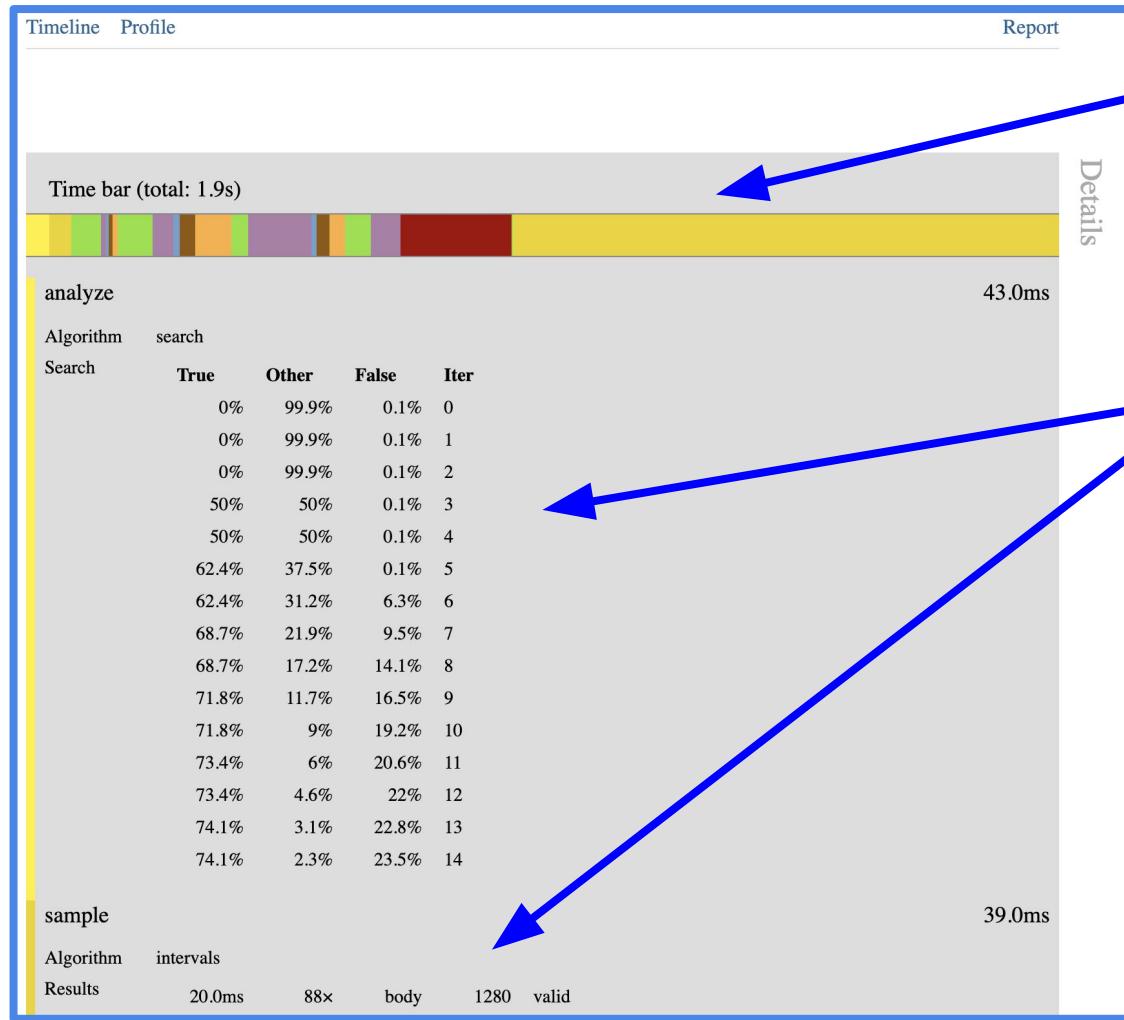
That which is measured, improves.



Global overviews

2. Measurement

That which is measured, improves.

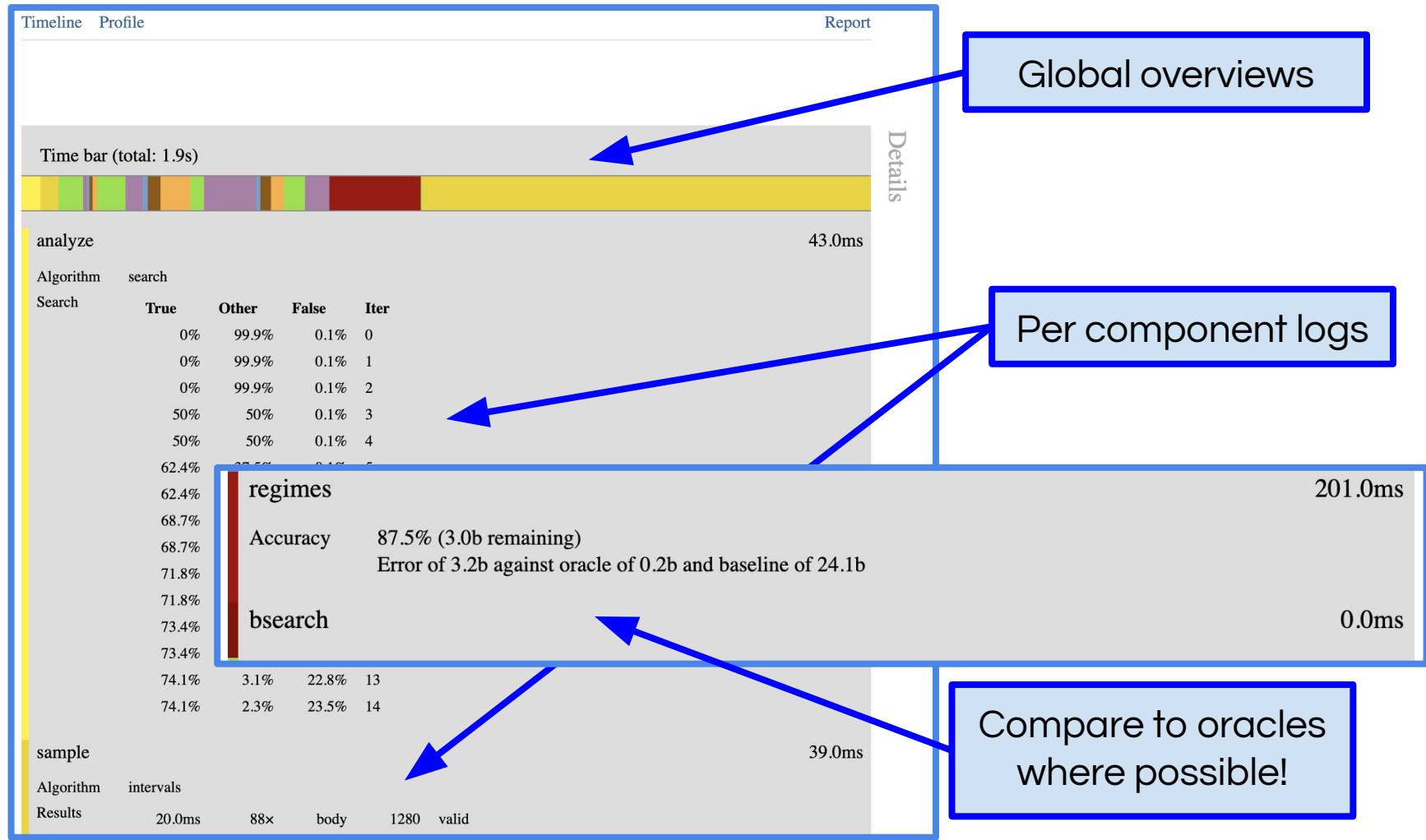


Global overviews

Per component logs

2. Measurement

That which is measured, improves.



3. Community

We are all in this together.

3. Community

We are all in this together.

Herbie is just one piece of the puzzle.

The workbench demands cooperating tools:

- Rigorous error bounds, precision tuning, performance optimization, ...

How should they interact? Need standards!



3. Community

We are all in this together.

Herbie is just one piece of the puzzle.

The workbench demands cooperating tools:

Numerics research is a community effort.

We need more open communications!

How We need more collaboration opportunities!

We need more shared infrastructure!

Our thinking
in 2016

View today

3. Community

We are all in this together.



FPTalks 2020
The leading edge of floating-point research
[Home](#) [Benchmarks](#) [Compilers](#) [Standards](#)

The floating-point research community.
Online. June 24, 2020.

FPTalks 2020 was held online over Zoom. All talks were live-streamed and recorded on Youtube. Each talk was 10 minutes long, followed by audience questions.

ada
Applications Driving Architectures

FPTalks 2020 was supported in part by the Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA.

Session 1 (Session Stream) **8:00–9:00 PDT**
Welcome
Pavel Panchekha, University of Utah
[Video](#), [Slides](#), [FPBench](#)

3. Community

We are all in this together.



FPTalks 2020
The leading edge of floating-point research
[Home](#) [Benchmarks](#) [Compilers](#) [Standards](#)

The floating-point res
Online. June

FPTalks 2020 was held online over Zoom
recorded on Youtube. Each talk was 10 m
questions.

ada
Applications Driving Architectures

Session 1 (Session Stream)
Welcome
Pavel Panchekha, University of Utah
[Video](#), [Slides](#), [FPBench](#)



FPBench Tools
Export and transform FPcores
[Home](#) [Benchmarks](#) [Compilers](#) [Standards](#)

FPBench ships two compiler tools for FPCore: an [exporter](#) and a
[transformation tool](#). Both are [available on Github](#).

Installing the FPBench tools

The FPBench tools require [Racket](#). Use the [official installer](#) to install Racket,
or use distro-provided packages provided they are version 7.0 or later of
Racket (earlier versions are not supported).

Test that Racket is installed correctly and has a correct version:

```
$ racket
Welcome to Racket v7.7.
> (exit)
```

Now that Racket is installed, [download](#) the FPBench tools, and enter the

3. Community

We are all in this together.



FPTalks 2020

The leading edge of floating-point research

Home Benchmarks Compilers Standards

The floating-point research community
Online. June 16-18, 2020

FPTalks 2020 was held online over Zoom. All talks were recorded on YouTube. Each talk was 10 minutes long, followed by 10 minutes of questions.

ada
Applications Driving Architectures

Session 1 (Session Stream)

Welcome
Pavel Panchekha, University of Utah
[Video](#), [Slides](#), [FPBench](#)



FPBench Tools

Export and transform FPCores

Home Benchmarks Compilers Standards

FPBench ships two compiler tools for floating-point cores: a transformation tool and a code generation tool. Both are available as command-line programs.

Installing the FPBench tools

The FPBench tools require Racket. Use the `raco setup` command or use distro-provided packages provided by your distribution. Racket (earlier versions are not supported).

Test that Racket is installed correctly:

```
$ racket
Welcome to Racket v7.7.
> (exit)
```

Now that Racket is installed, download the FPBench tools from [fpbench.org](#).



Community Tools

Floating-point Research Tools

Home Benchmarks Compilers Standards

The floating-point research community has developed many tools that authors of numerical software can use to test, analyze, or improve their code. They are listed here in alphabetical order and categorized into three general types: 1) [dynamic tools](#), 2) [static tools](#), 3) [commercial tools](#), and 4) [miscellaneous](#). You can find tutorials for some of these tools at [fpanalysistools.org](#), and many of the reduced-precision tools were compared in a [recent survey](#) by authors from the Polytechnic University of Milan.

Tools on the list are not endorsed, and need not endorse or support the FPBench project. The list is intended to be exhaustive.

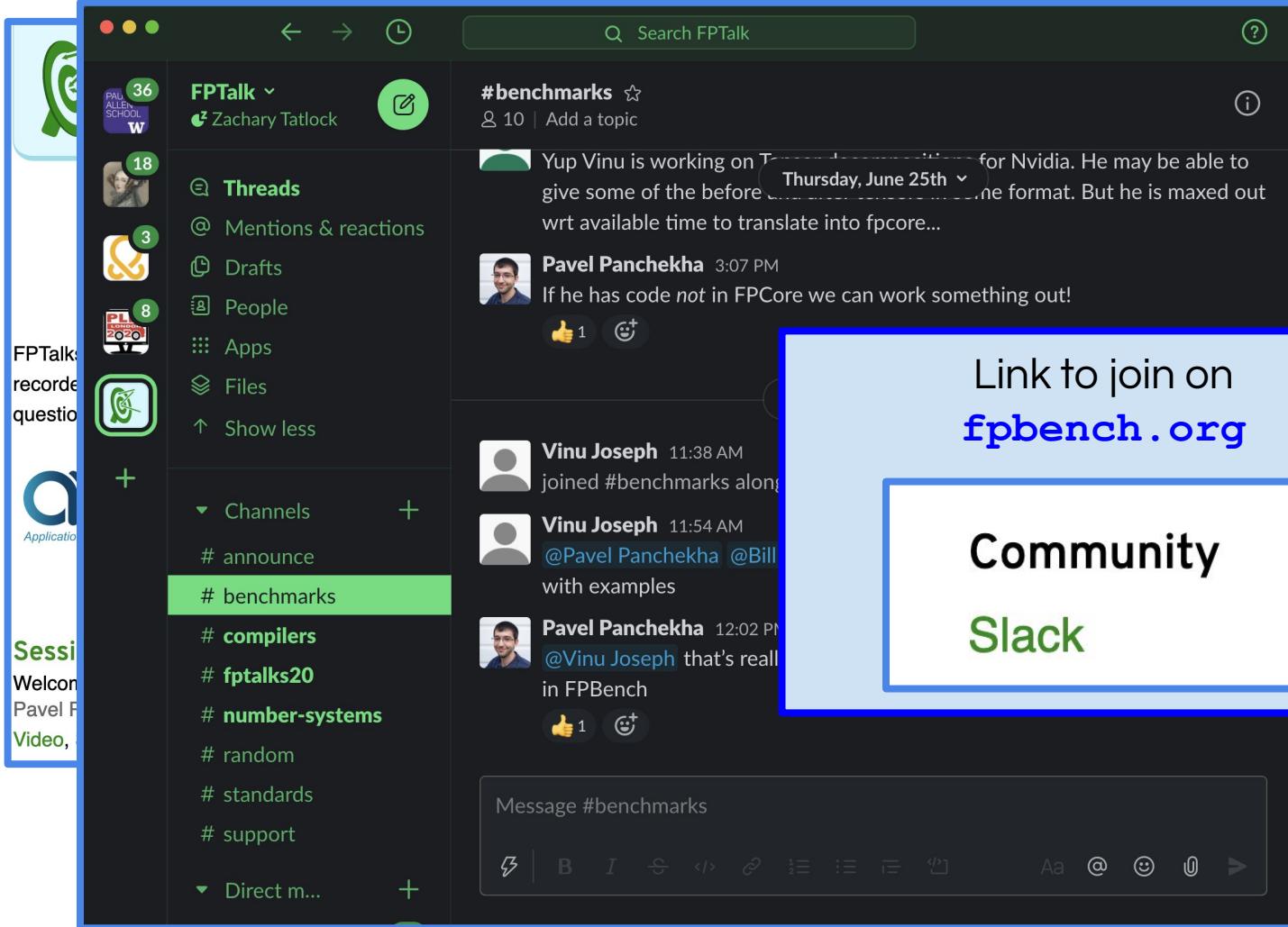
3. Community

We are all in this together.

The screenshot shows the FPTalk application interface. On the left, there's a sidebar with various icons and counts: 36 notifications for 'PAL ALLEN SCHOOL W', 18 for another section, 3 for 'Drafts', 8 for 'PFTALKS 20', and a large green icon for 'FPTalk'. Below these are buttons for '+', 'Channels', '# announce', and a list of channels including '# benchmarks' (which is highlighted in green), '# compilers', '# fptalks20', '# number-systems', '# random', '# standards', '# support', and 'Direct m...'. To the right of the sidebar is a main pane with a search bar at the top. A topic titled '#benchmarks' is selected, showing a message from 'Zachary Tatlock' (@ 10) about Vinu Joseph working on something for Nvidia. Pavel Panchekha responds, suggesting they can work something out if the code is not in FPCore. Vinu Joseph joins the channel. Pavel Panchekha then mentions that @Pavel Panchekha and @Bill Zorn will send over a GitHub link with examples. At the bottom, there's a message input field for '#benchmarks' with various text and emoji icons.

3. Community

We are all in this together.



The screenshot shows a Slack interface for the FPTalk channel. On the left, there's a sidebar with various icons and a list of channels. The channel list includes:

- # benchmarks (highlighted in green)
- # compilers
- # fptalks20
- # number-systems
- # random
- # standards
- # support
- Direct m...

The main area shows a message from Pavel Panchekha (@Pavel_Panchevka) at 3:07 PM: "Yup Vinu is working on T... for Nvidia. He may be able to give some of the before Thursday, June 25th ... format. But he is maxed out wrt available time to translate into fpcore..." followed by a reply from Vinu Joseph (@Vinu_Joseph) at 11:38 AM: "joined #benchmarks along with examples". Below this, Pavel Panchekha (@Pavel_Panchevka) at 12:02 PM says: "@Vinu_Joseph that's real... in FPBench". A blue box highlights the text "Link to join on fpbench.org". Another blue box highlights the word "Community" in the channel list and the word "Slack" in the message area.

Link to join on fpbench.org

Community

Slack

4. Generality

Supporting a rapidly evolving landscape

4. Generality

Supporting a rapidly evolving landscape

Explosion of numerical representations.

How do we adapt code? Optimize? Need tools!

4. Generality

Supporting a rapidly evolving landscape

Explosion of numerical representations.

How do we adapt code? Optimize? Need tools!

```
(FPCore (x)
  :name "float32 ulp"
  :precision binary64
  (! :precision binary32 (- (! :round toPositive (cast x))
    (! :round toNegative (cast x)))))
```

```
(FPCore (x)
  :name "round with addition"
  :spec (round x)
  (let ([n (! :precision binary64 6755399441055744)])
    (! :precision binary64 :round nearestEven (- (+ x n) n))))
```

“Precision scopes”
specify MPMF!

4. Generality

Supporting a rapidly evolving landscape

```
(FPCore (n)
  :name "arclength"
  :cite (precimonious-2013)
  :precision (posit 1 16)
  :pre (>= n 0)

  (let* ([dppi (acos -1)]
         [h (/ dppi n)]
         [t1 0])

    (while* (<= i n)

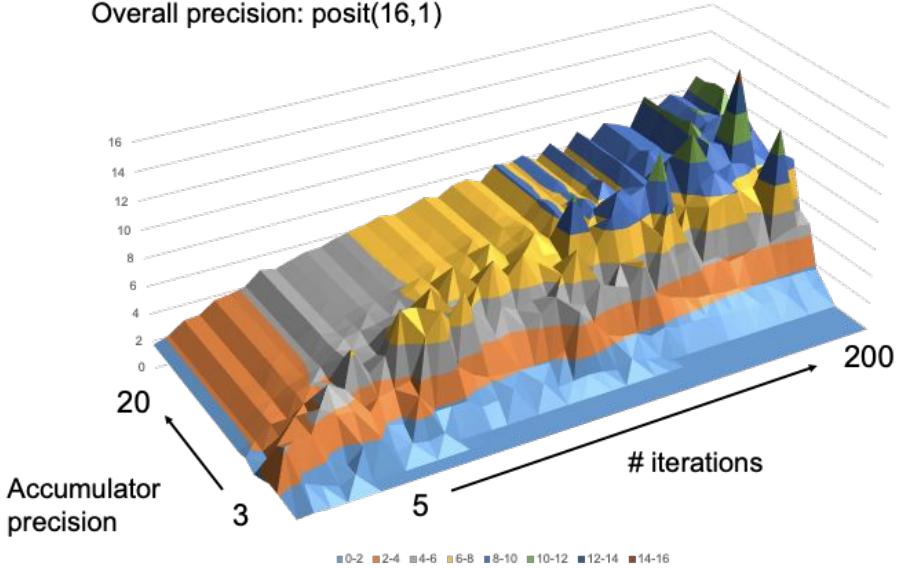
      ([t2 0
        (let ([x (* i h)])
          (while* (<= k 5)
            ([d1 1
              (! :precision binary32 (* d1 2))]
             [t1 x
               (+ t1 (/ (* sin (* d1 x)) d1))]
             [k 1 (+ k 1)])
            t1))

      [s1 0
        (let ([s0 (sqrt (+ (* h h) (* (- t2 t1) (- t2 t1))))])
          (! :precision (fixed -8 12) (+ s1 s0))]

      [t1 t1 t2]
      [i 1 (+ i 1)])]

    s1)))
```

Overall precision: posit(16,1)



Enables parameter sweeps:
iters, cost, accuracy

4. Generality

Supporting a rapidly evolving landscape

Increasing interest in numerics of tensor code.

4. Generality

Supporting a rapidly evolving landscape

Increasing interest in numerics of tensor code.

```
(FPCore matmul ((A am an) (B bm bn))
  :pre (== an bm)
  (tensor ([m am]
           [n bn])
    (for ([i bm])
      ([prod 0 (+ prod (* (ref A m i) (ref B i n))))]
       prod)))
  )
```

FPBench 2.0 adds tensor support, keeps minimal core

Where do we go from here?

Trust: ground truth for MPMF

Measurement: metrics for tensor benchmarks

Community: outreach from research to practice

Generality: extensibility for new formats

A brief announcement...

Herbie 1.4 just released!



Please take it for a spin :)

herbie.uwplse.org



Many Acknowledgements!

Alex Sanchez-Stern
Bill Zorn
David Thien
Oliver Flatt
Brett Saiki
Jason Qiu
Ian Briggs

Heiko Becker
Eva Darulova
Max Willsey
James Wilcox
Jack Firth
Mike Lam
... and more!



Sandia
National
Laboratories



Thank You!

Trust: ground truth for MPMF

Measurement: metrics for tensor benchmarks

Community: outreach from research to practice

Generality: extensibility for new formats



herbie.uwplse.org



fpbench.org