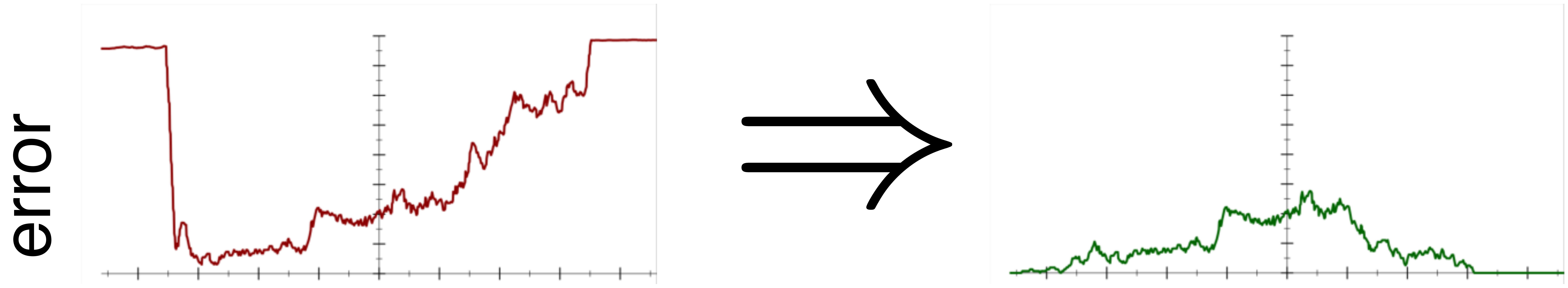
Towards Automated Floating-point Tools for End Users



Zachary Tatlock
CoNGA 2019

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING
PLSE
SRC JUMP
ada
Applications Driving Architectures

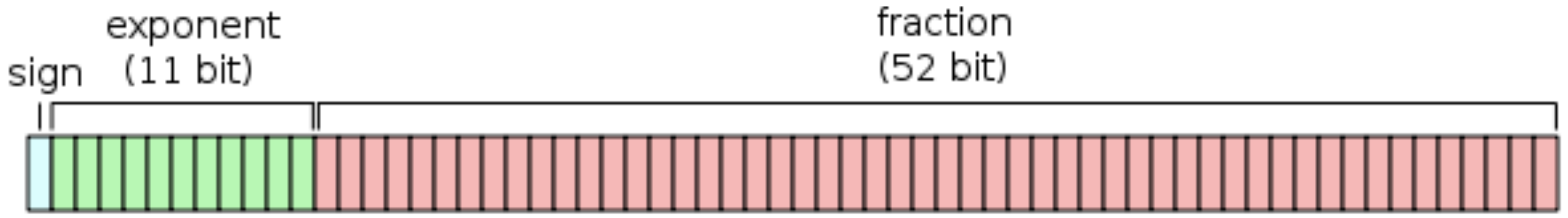
Growing Community of Contributors

- 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

LAMPS

Floating Point's Wild Success

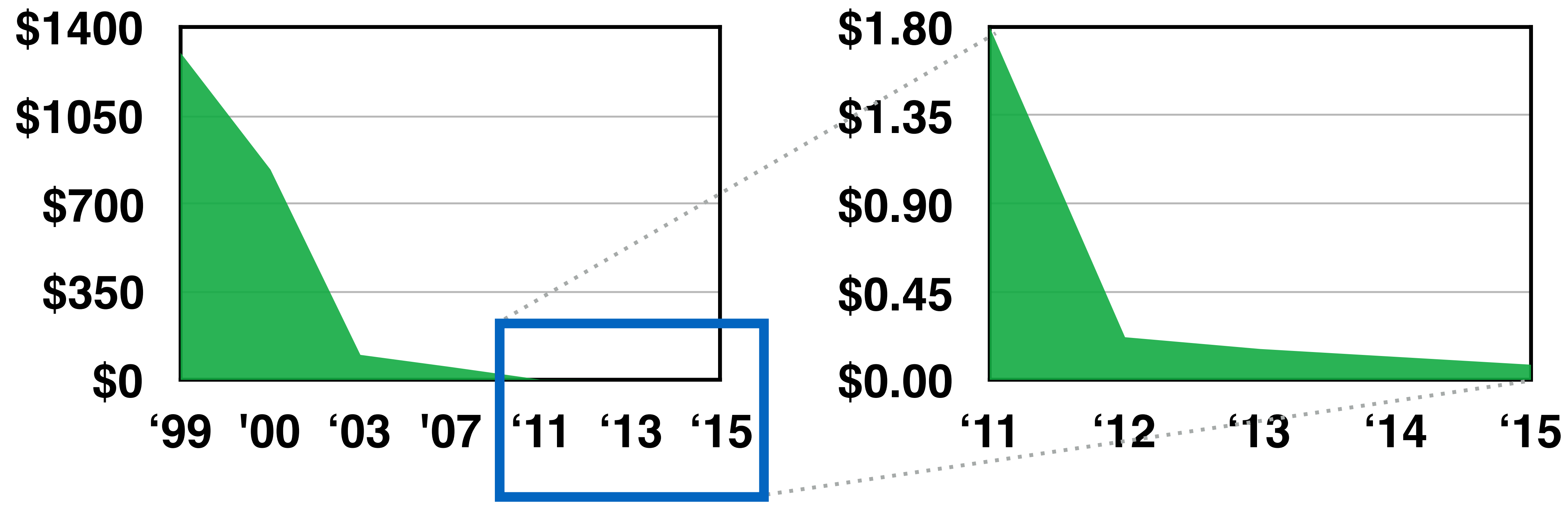
Flexibility



vast range:10^-324 to 10^308

Performance

cheap GFLOPS



Accuracy

ops w/ min error

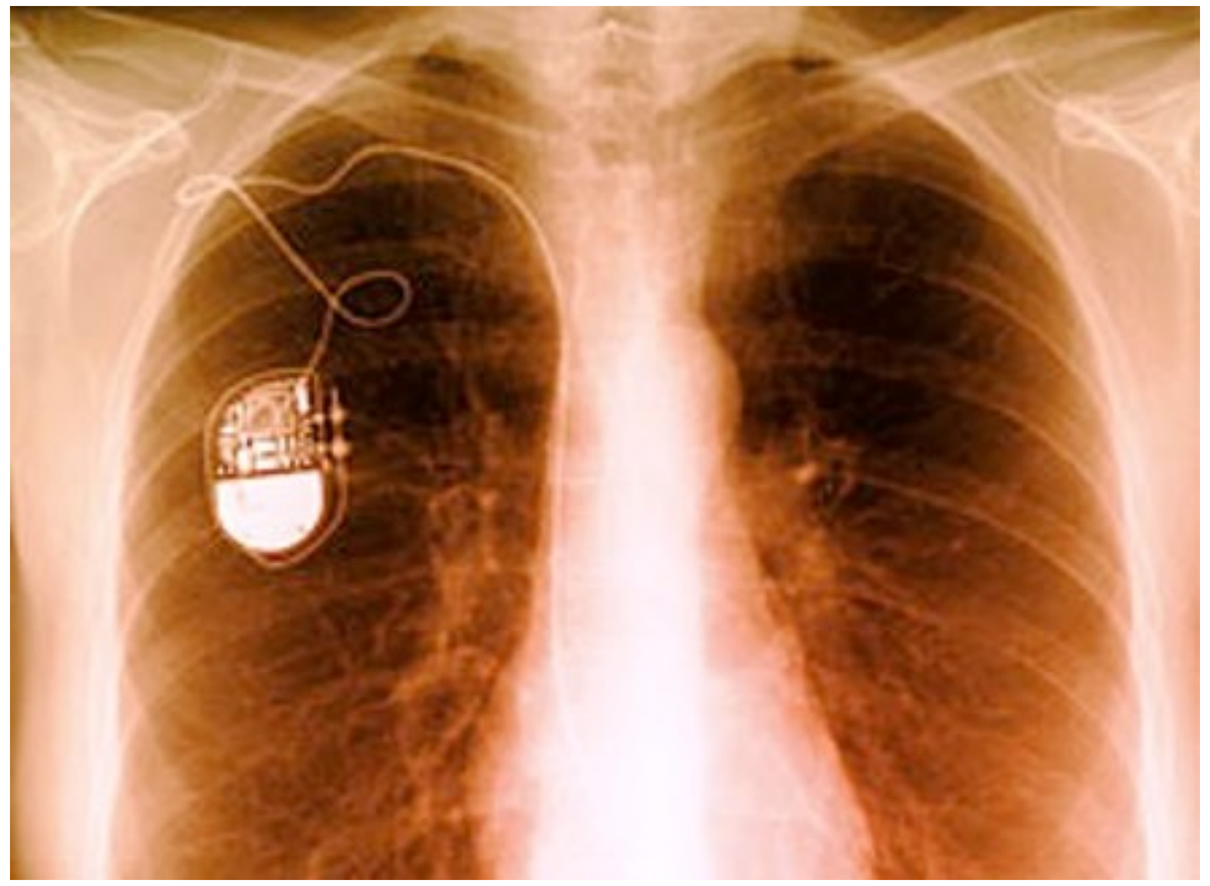
$$[\![x \star y]\!]_{\mathbb{F}} = \mathrm{Round}([\![x \star y]\!]_{\mathbb{R}})$$

Compute in $\mathbb{F}$

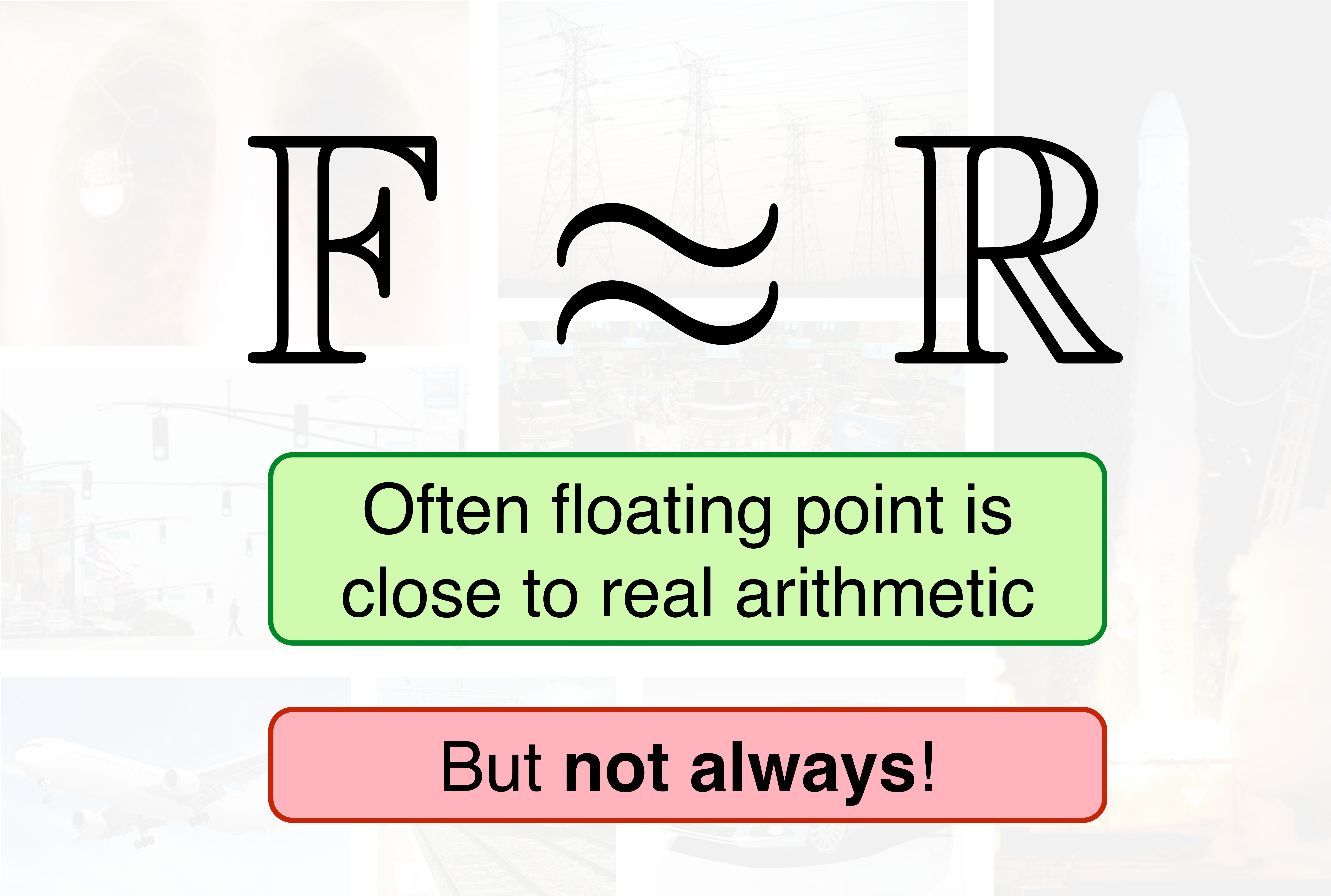
Round to $\mathbb{F}$

Compute in $\mathbb{R}$

Floating Point’s Wild Success



Floating Point’s Wild Success

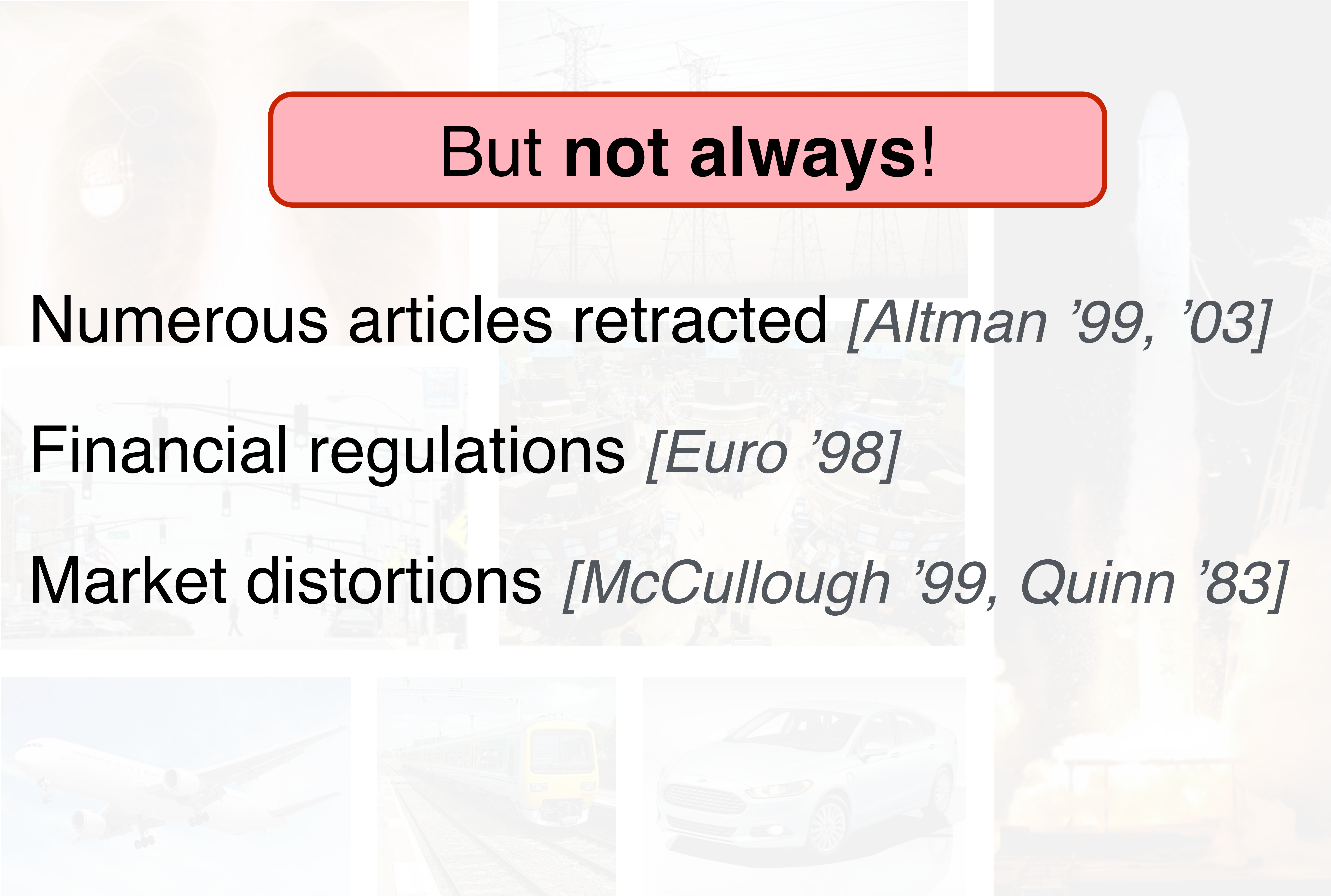


$$\mathbb{F} \approx \mathbb{R}$$

Often floating point is close to real arithmetic

But **not always**!

Floating Point’s Wild Success



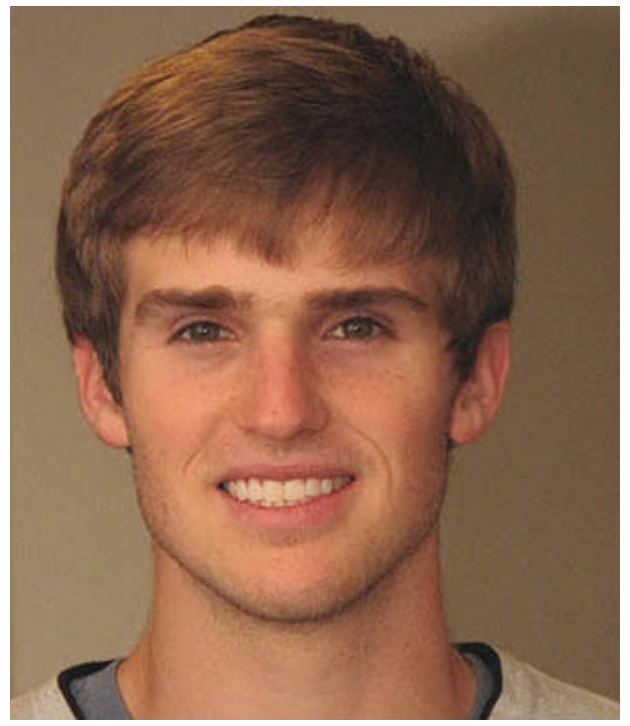
But not always!

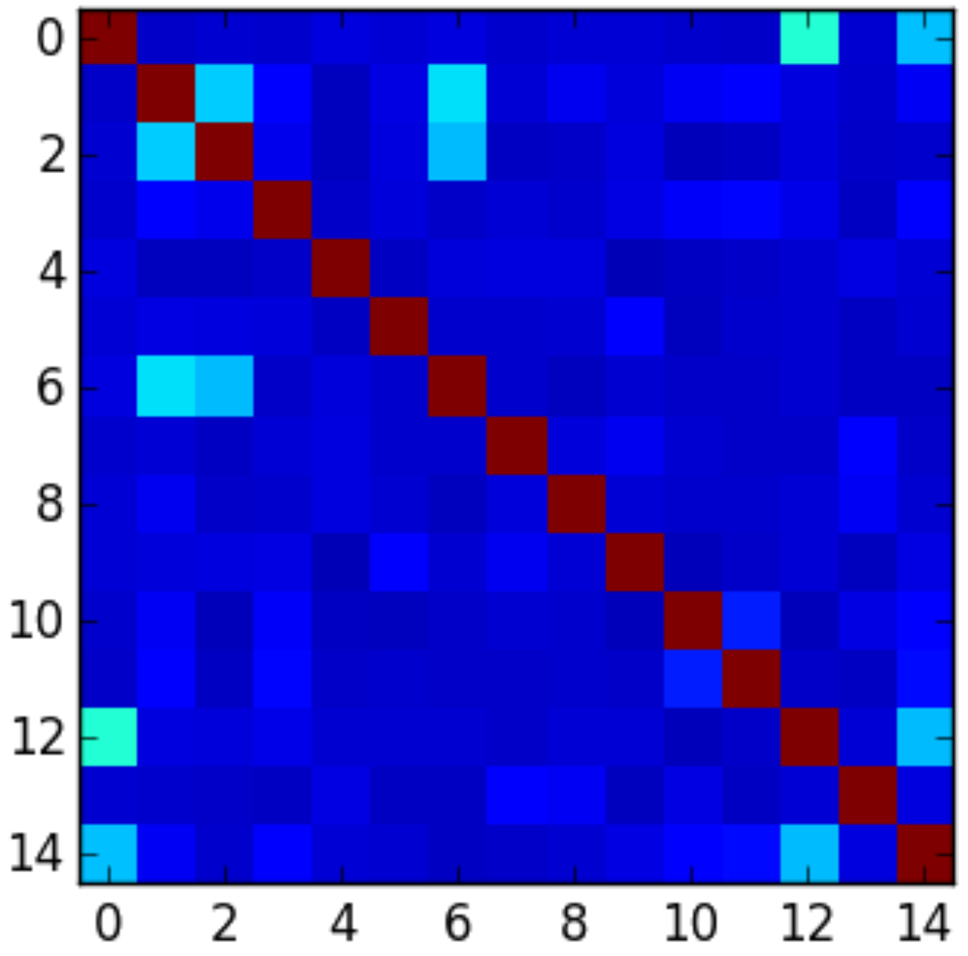
Numerous articles retracted *[Altman ’99, ’03]*

Financial regulations *[Euro ’98]*

Market distortions *[McCullough ’99, Quinn ’83]*

Error in Machine Learning

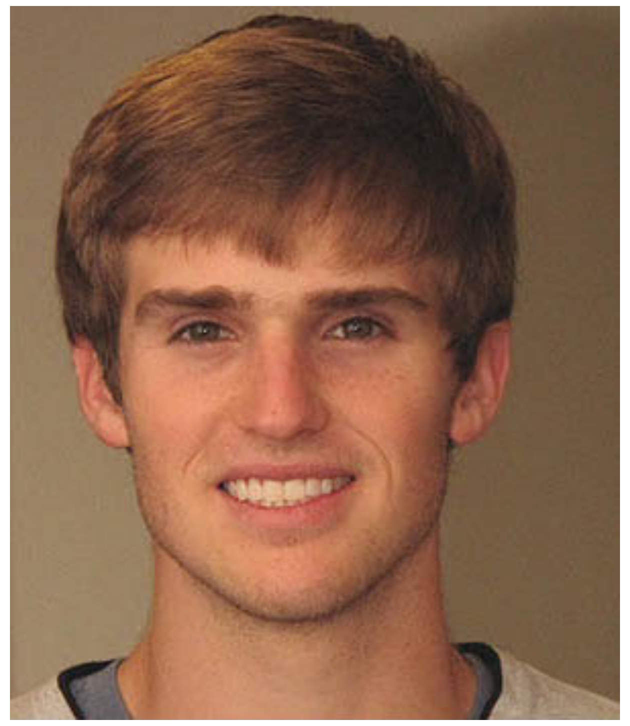




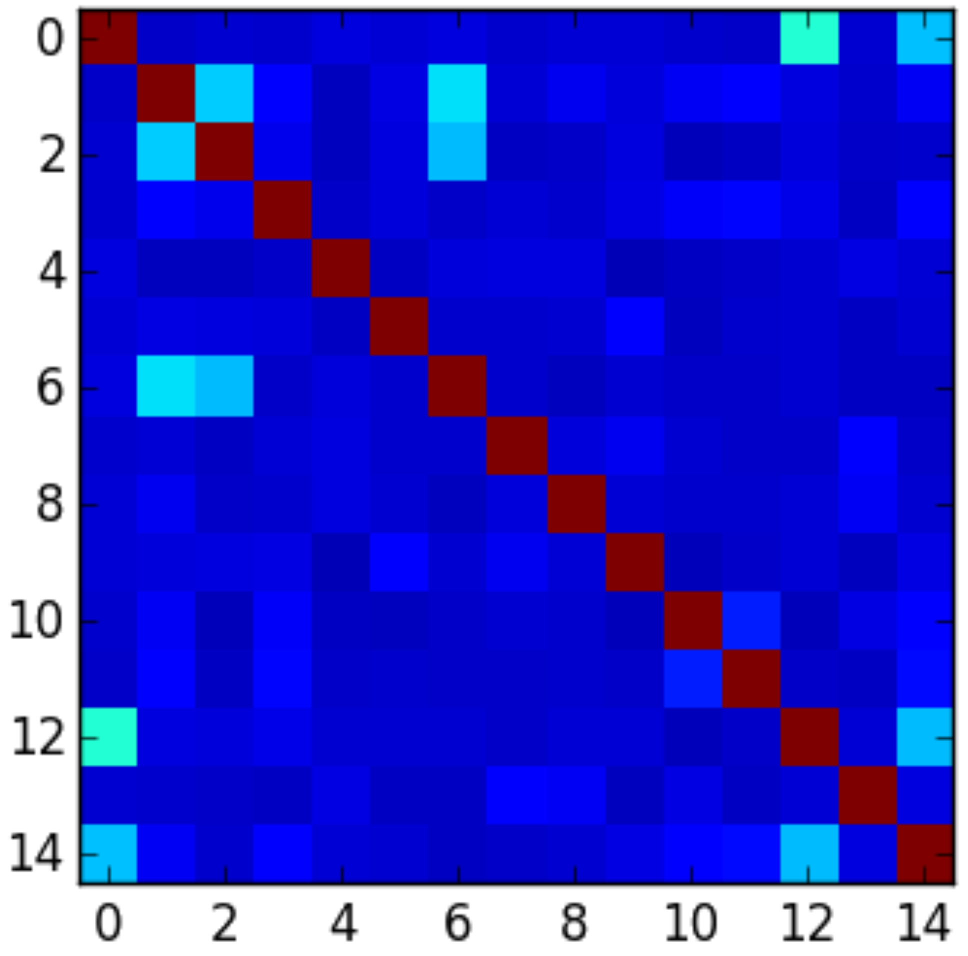
Harley Montgomery
AI Researcher

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.

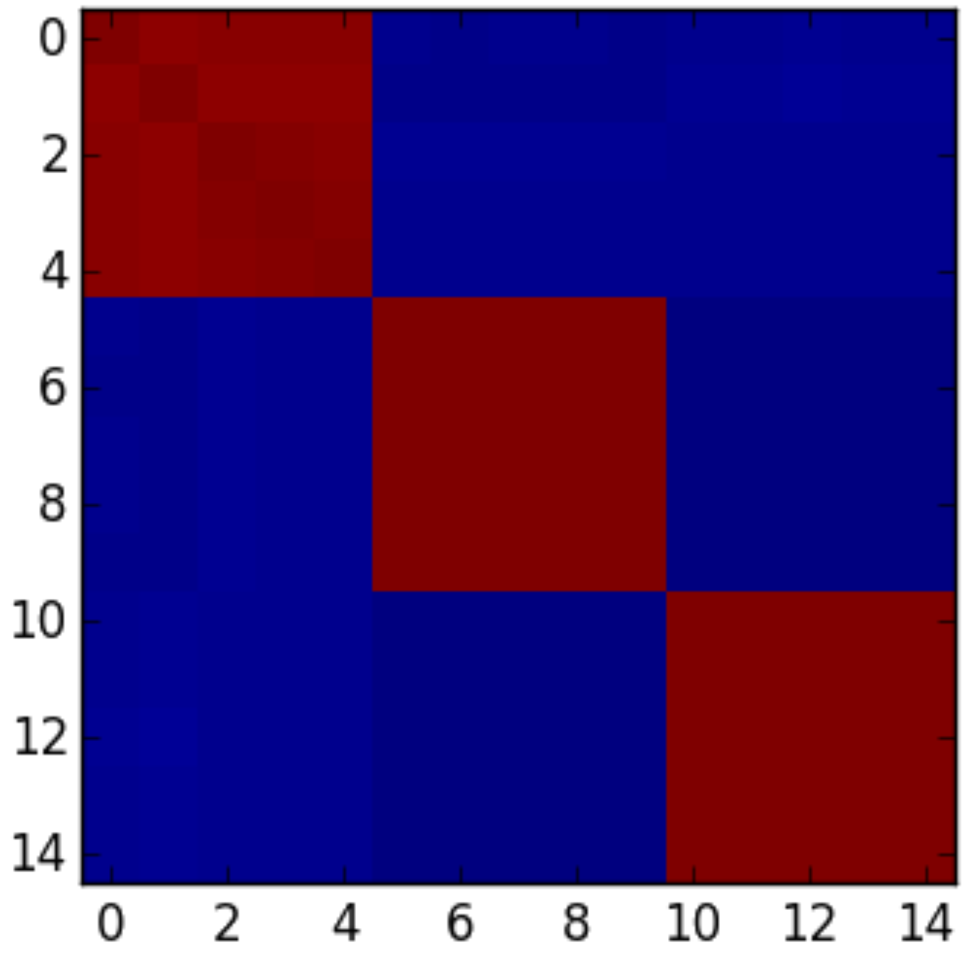
Error in Machine Learning



Harley Montgomery
AI Researcher



$\Rightarrow$



Bigger, darker blocks better

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

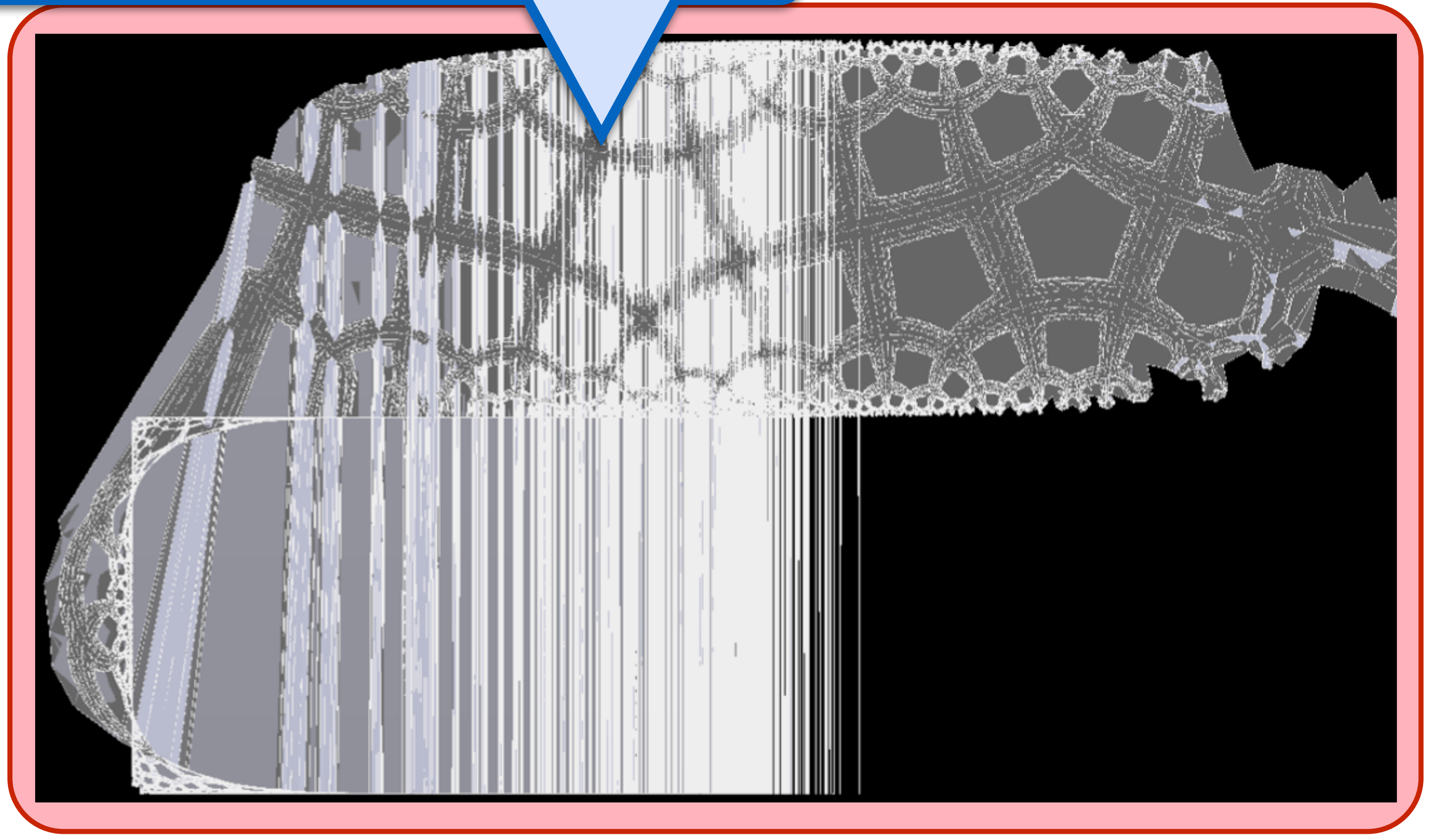
Rounding error

$$\exp\left(c_p \ln \frac{1+e^{-t}}{1+e^{-s}} + c_n \ln \frac{1-\frac{1}{1+e^{-s}}}{1-\frac{1}{1+e^{-t}}}\right)$$

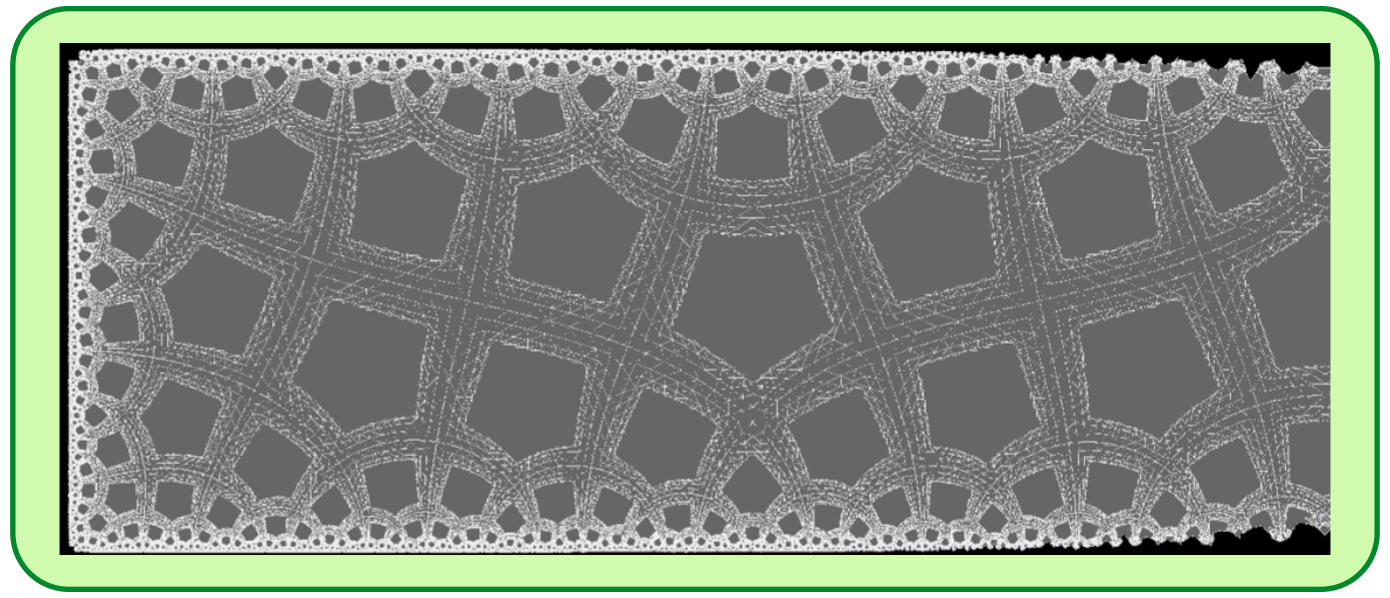
Rounding Error in CAD/CAM

Blake Courter
CAD Engineer

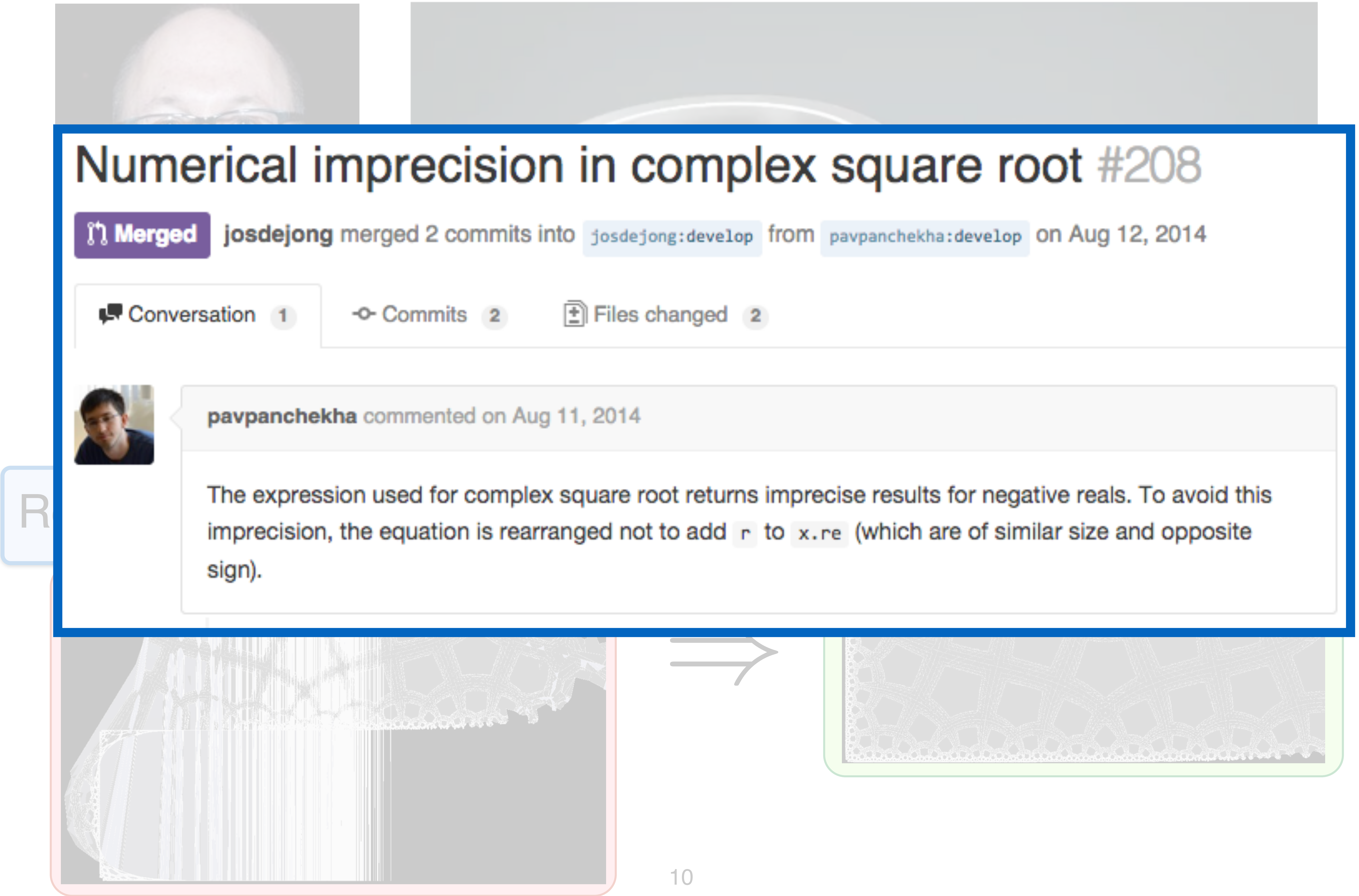
Rounding error



$\Rightarrow$



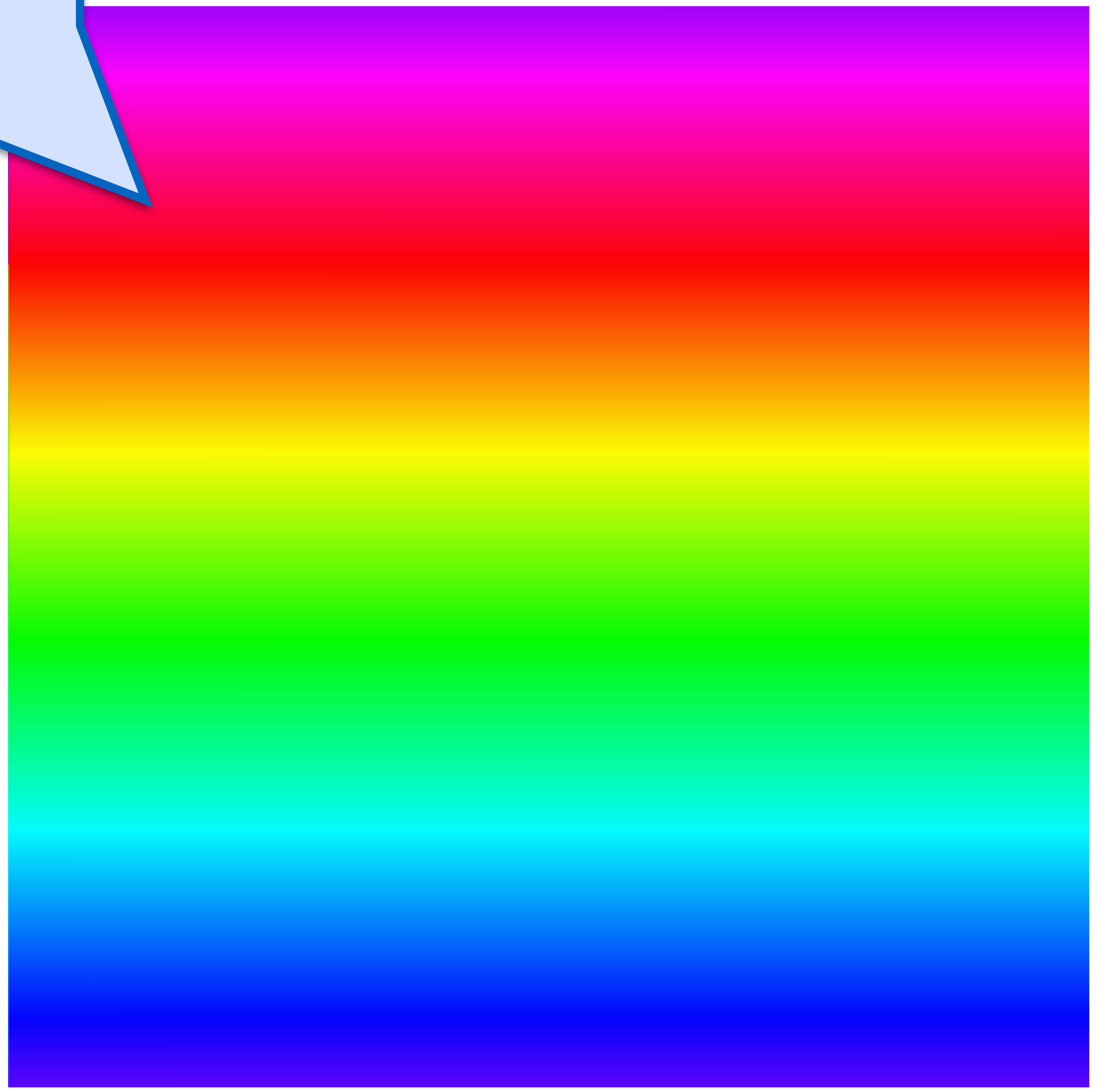
Rounding Error in CAD/CAM

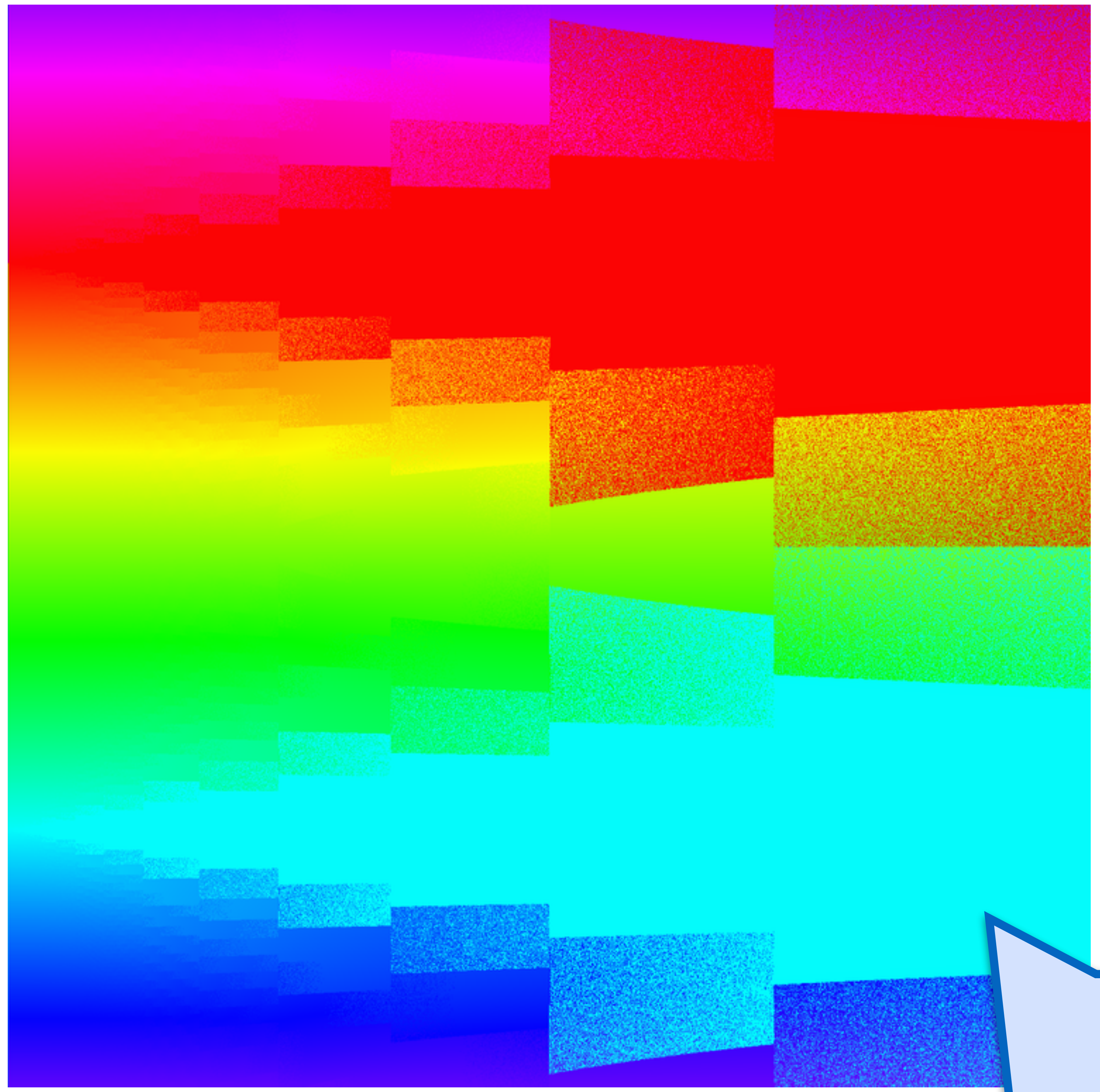


Error in Complex Plotting

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

Correct Output (smooth)





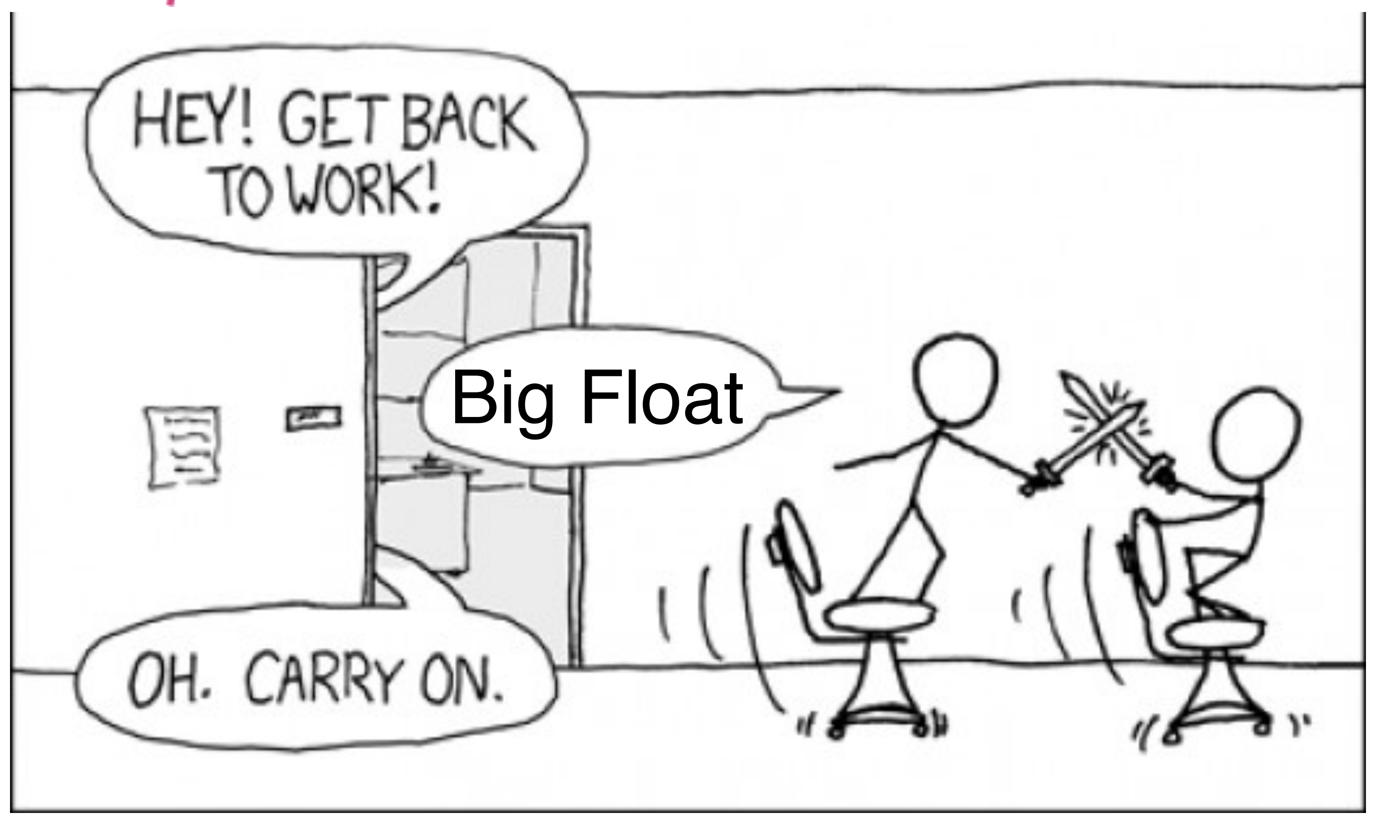
Floating Point (not smooth)

Existing options

– Unreliable

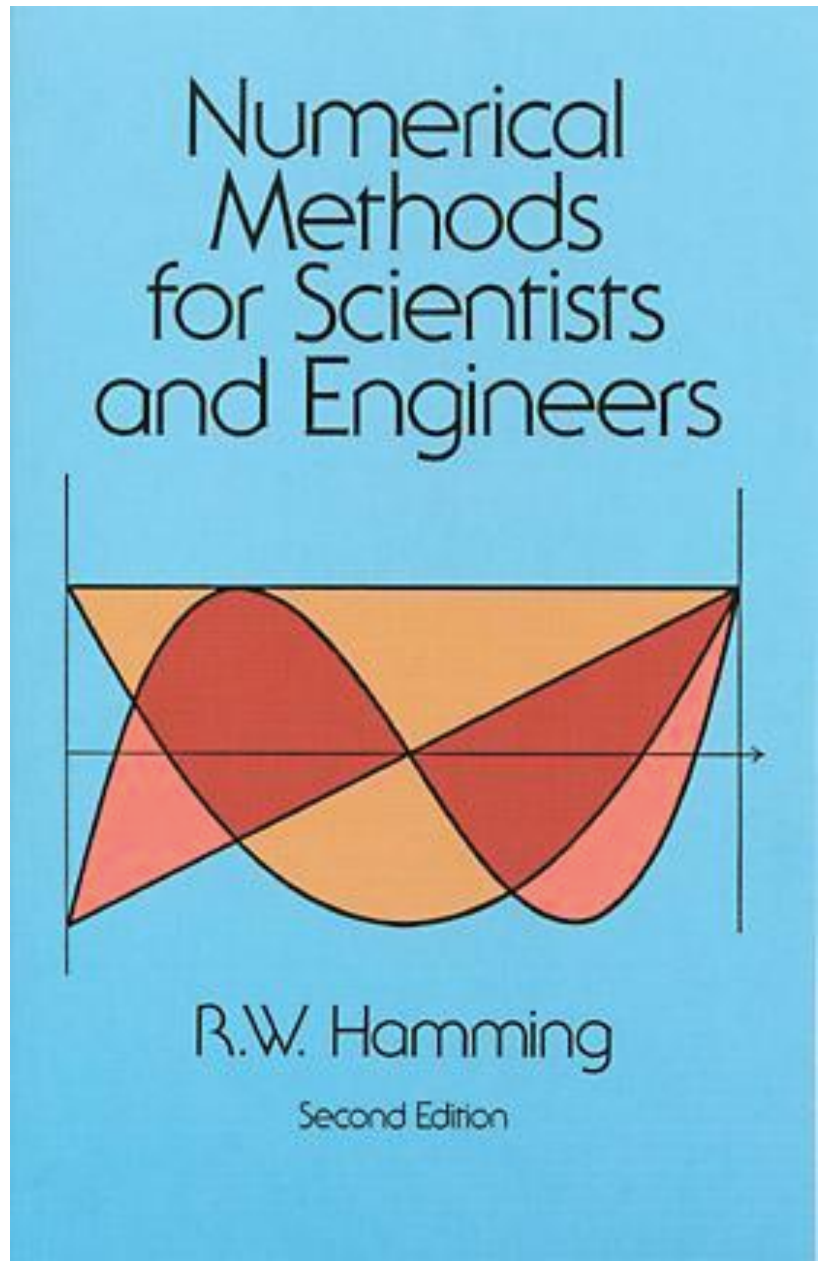
+ Fast Code

MPFR



+ 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

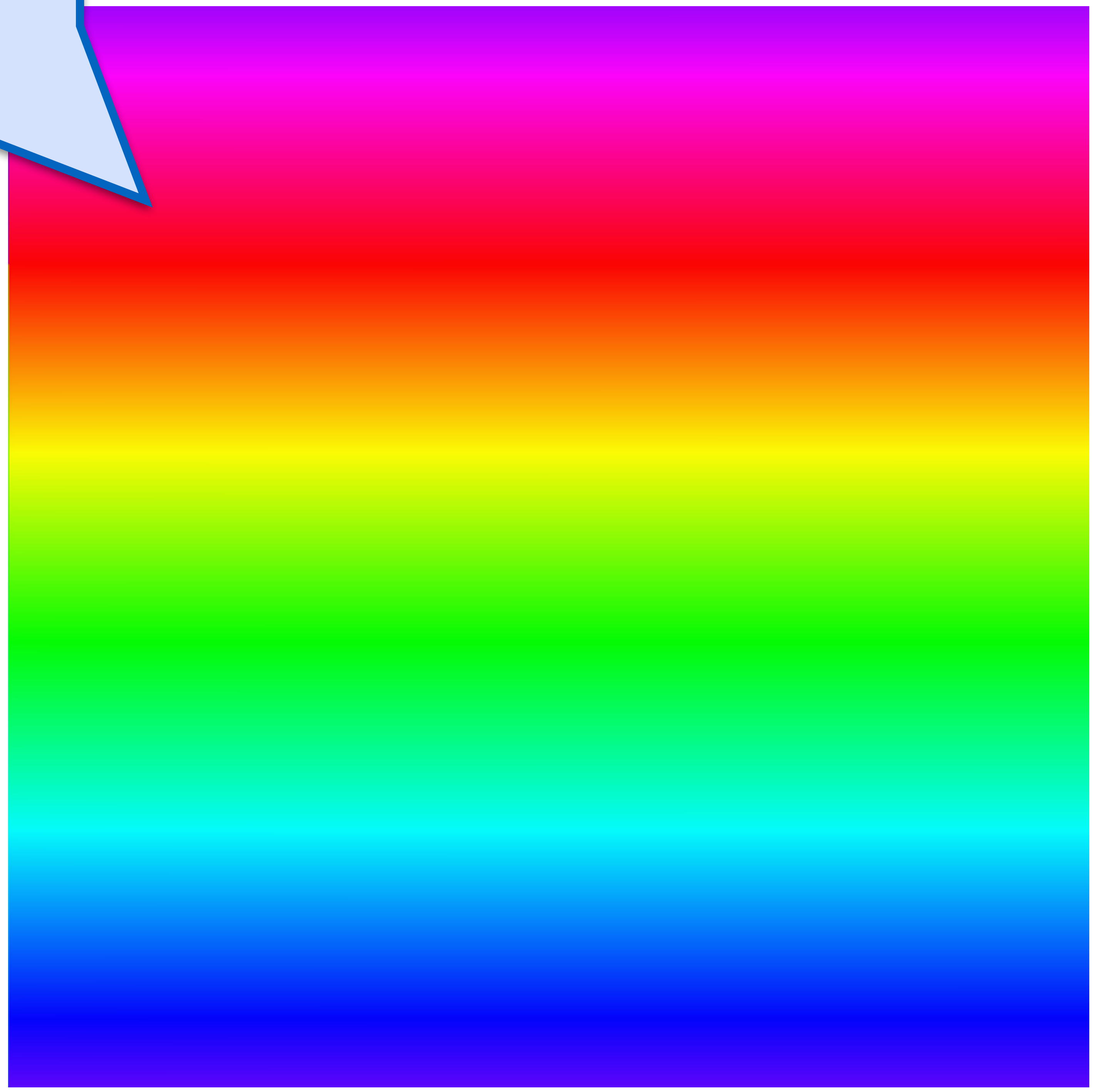
FPBench: A standard format for composing tools

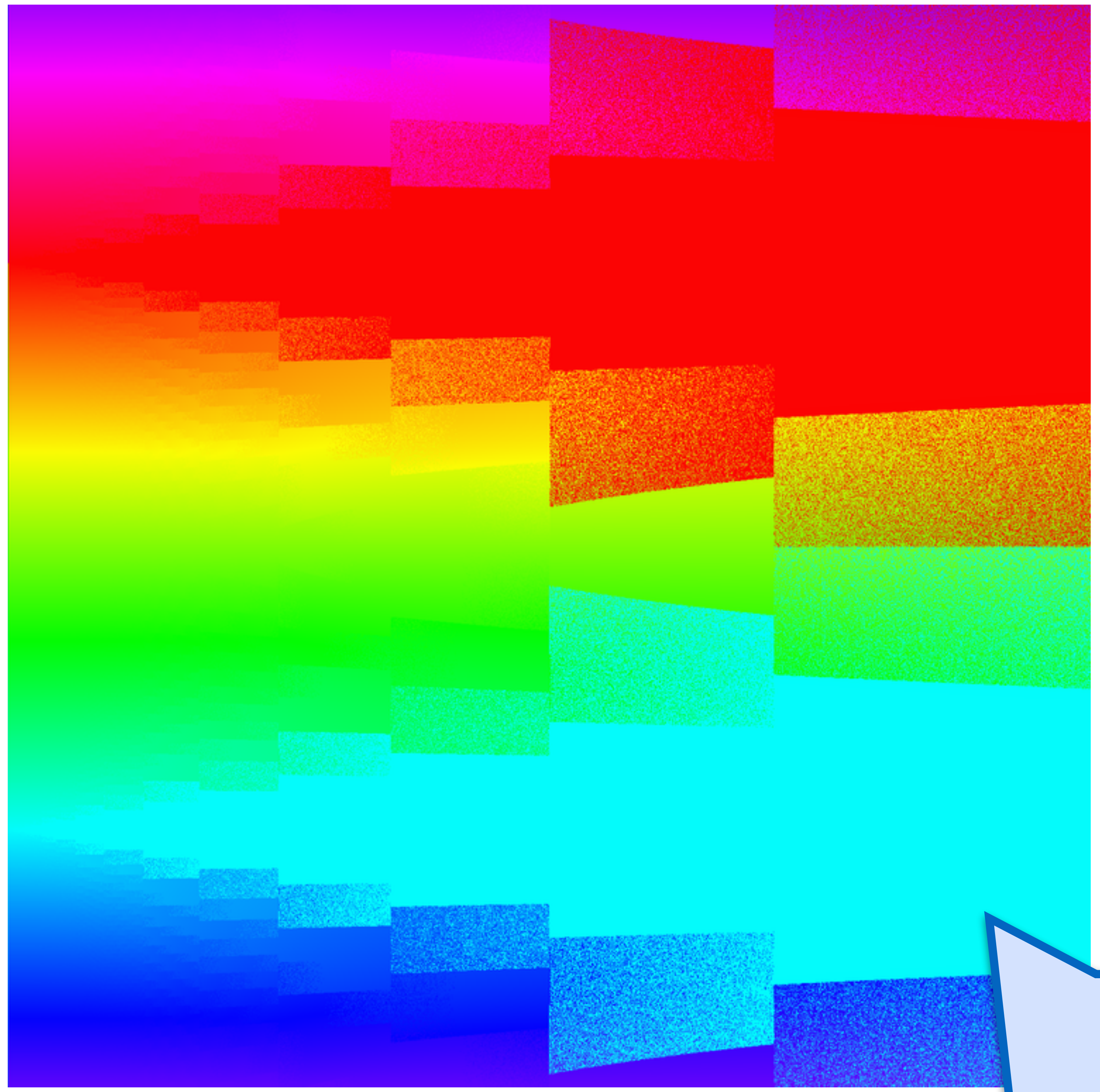
Titanic: A laboratory for exploring number systems

Error in Complex Plotting

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

Correct Output
(smooth)

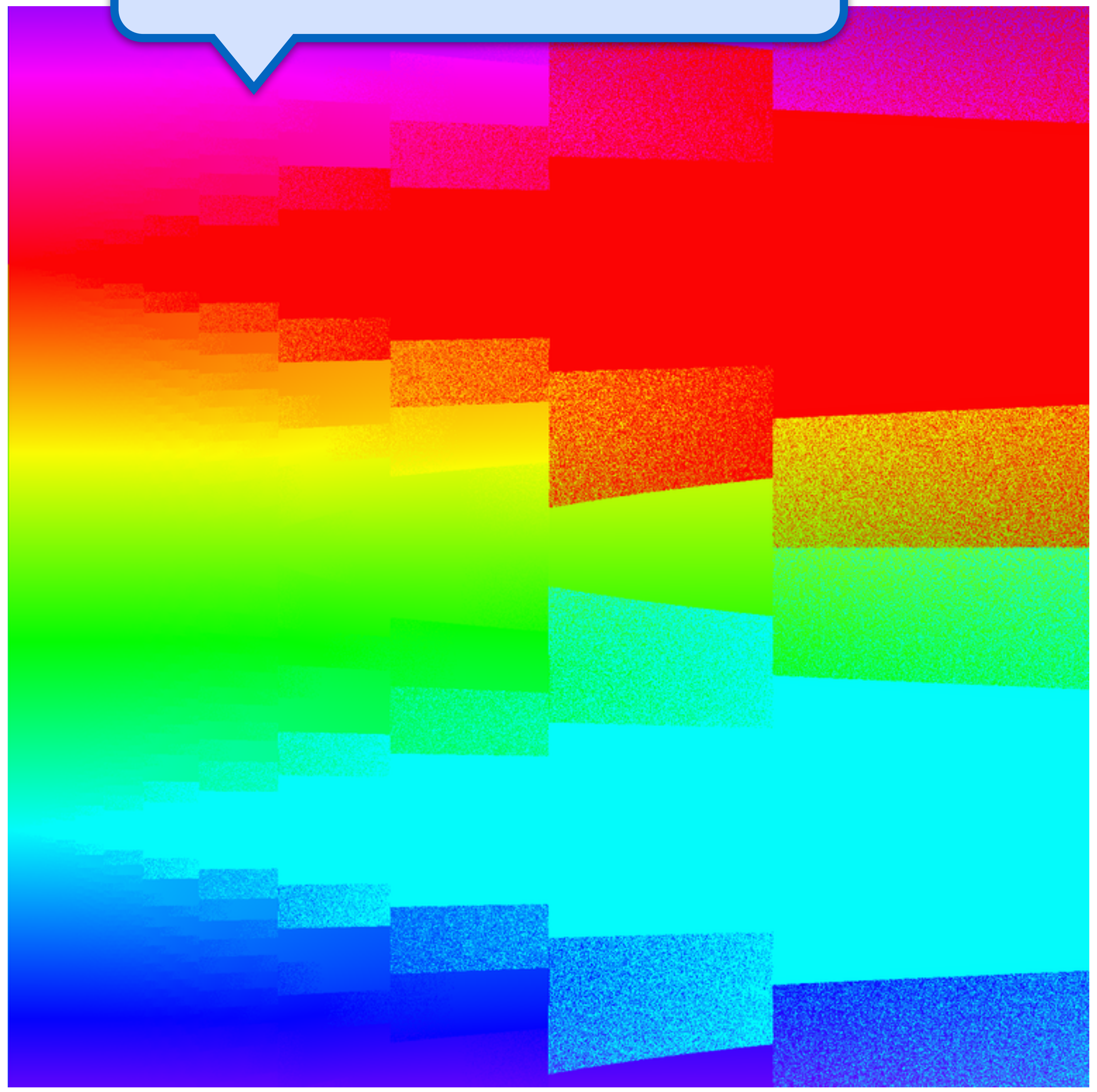




Floating Point
(not smooth)

Finding error in large programs

Erroneous output



- Generate input values
- Compute function
- Interpolate and resample
- Encode to colors
- Generate bitmap

Multiple libraries, data types, complex memory layout

Finding error in large programs

Compute function

Source of error

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

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

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

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

Input ranges

$$\frac{\left(-b + \sqrt{b^2 - 4ac}\right)}{2a} \Big\} \quad b > 0$$

Expression computed

Symptom of Error

```
cvttsd2si       %xmm0, %eax
sall     $16, %eax
movl     %eax, %edx
movsd    -48(%rbp), %xmm1
movsd    .LC7(%rip), %xmm0
mulsd    %xmm1, %xmm0
cvttsd2si       %xmm0, %eax
sall     $8, %eax
addl     %eax, %edx
movsd    -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

Herbgrind Example Output [PLDI 18]

Erroneous output

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

Root cause

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

3
$$\sqrt{b^2 - 4 \cdot 1 \cdot c} - b$$

where

4
$$-2\textsc{e}{-9} < b < 0.2 \qquad -2\textsc{e}{-9} < c < 0.2$$

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 $$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

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
movsd     %xmm5, (%rsp)
call      __muldc3
movsd     16(%rsp), %xmm4
```

Floating-point instructions

\$5	←	mul	\$1	\$3
\$5	←	mul	\$5	\$4
\$4	←	mul	\$2	\$2
\$5	←	add	\$4	\$5
\$4	←	sqrt	\$5	
\$4	←	sub	\$4	\$2
\$5	←	add	\$1	\$1
\$0	←	div	\$4	\$5

Executing one instruction

	\$4	← sub	\$4	\$2
Float	0.0		7.389	7.389

Executing one instruction

		$4	← sub	$4	$2
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 $$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

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 $$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

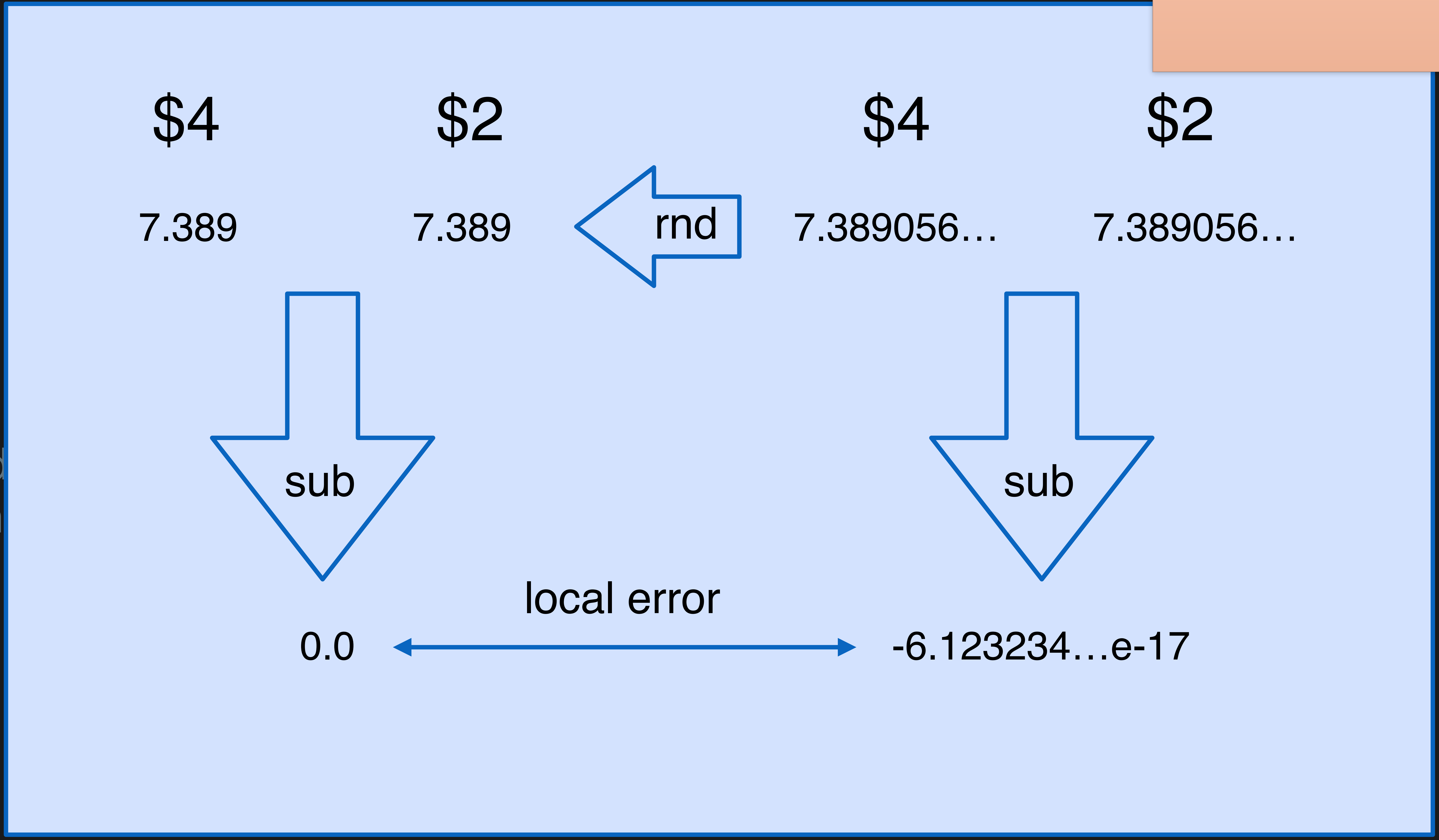
Executing one instruction

		$4		$4	$2
			sub		
Shadow Memory	Float	0.0		7.389	7.389
	Real	-6.123234…e-17		7.389056…	7.389056…

Executing one instruction

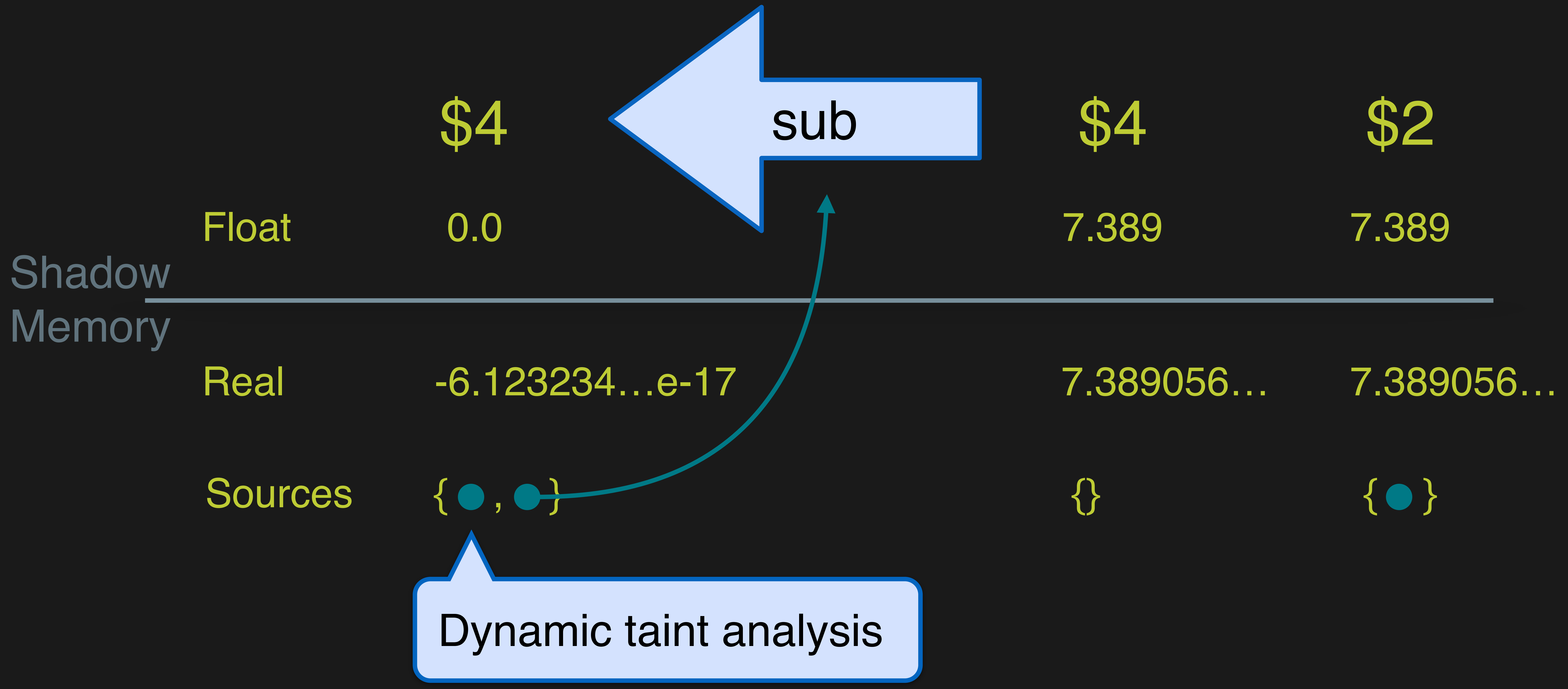
Emphasize local error as key idea

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



Shad
Mem

Executing one instruction



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 $$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

Herbgrind Example Output

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

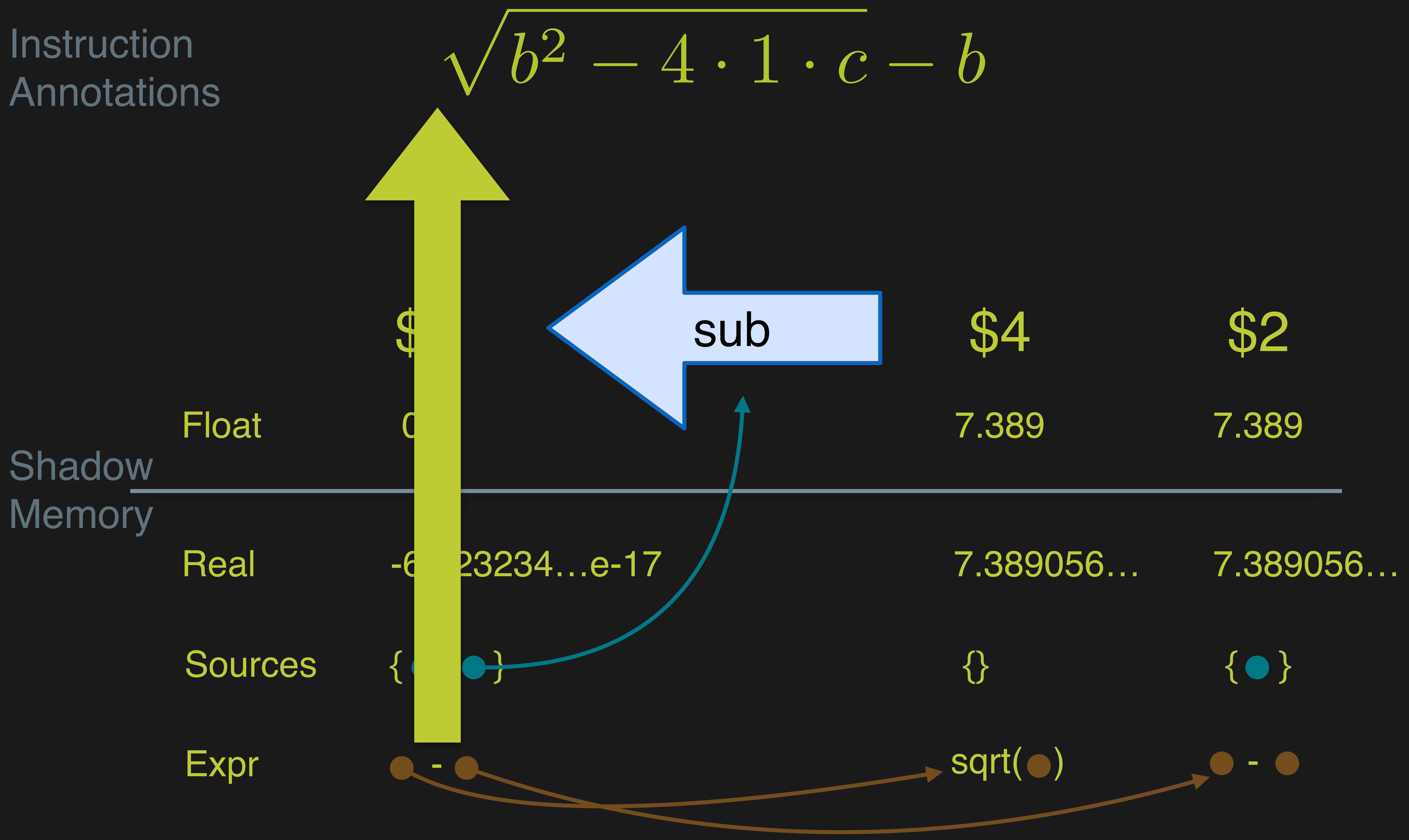
2 **Influenced** by main.cpp:12 in sqrt(complex)

3
$$\boxed{\sqrt{b^2 - 4 \cdot 1 \cdot c} - b}$$

where

4
$$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

Executing one instruction



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

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

where

4

$$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

Herbgrind Example Output

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

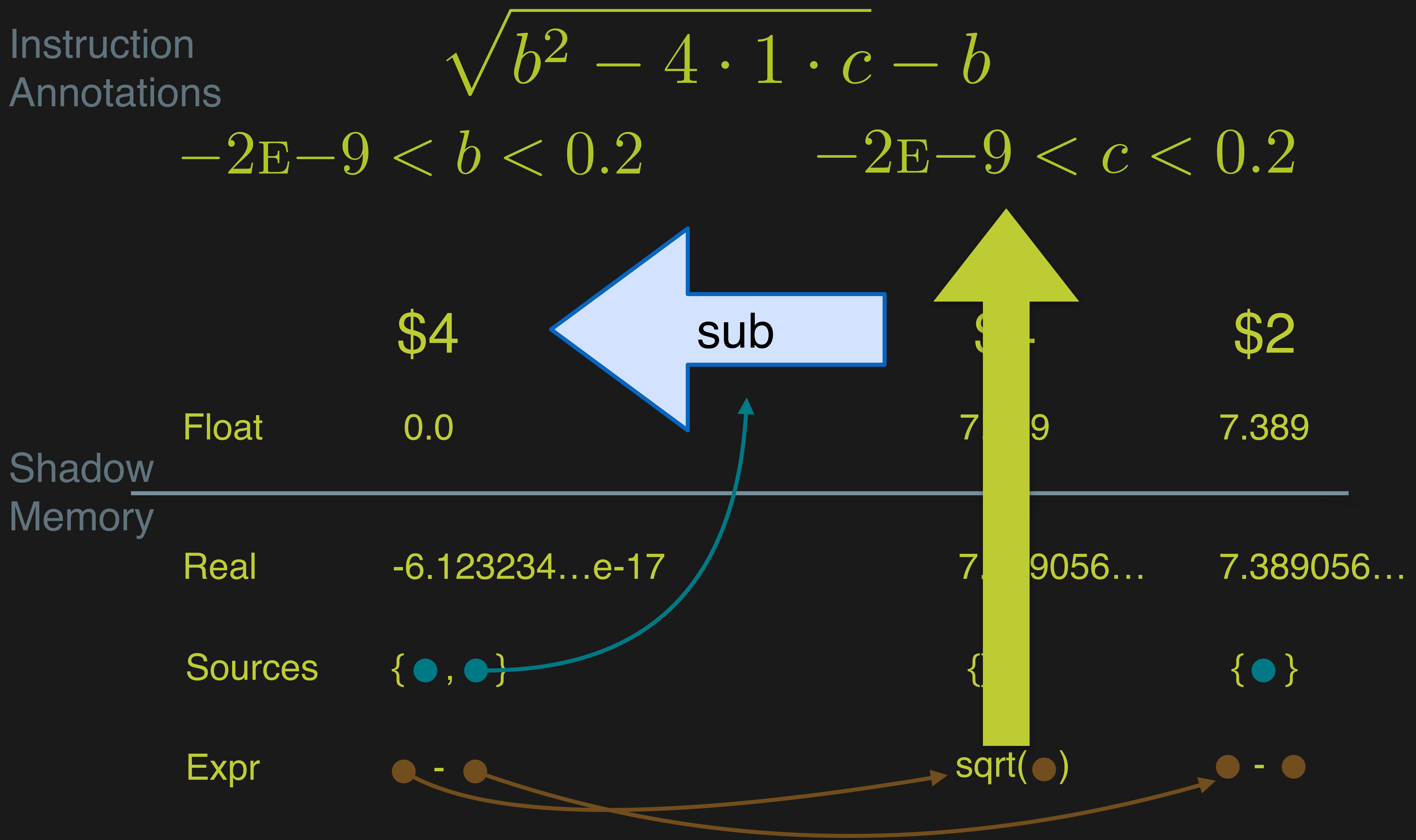
2 **Influenced** by main.cpp:12 in sqrt(complex)

3 $$\sqrt{b^2 - 4 \cdot 1 \cdot c} - b$$

where

4 $-2\text{E}{-9} < b < 0.2$ $\quad$ $-2\text{E}{-9} < c < 0.2$

Executing one instruction



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. $-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$

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. $$-2\text{E}{-9} < b < 0.2 \qquad -2\text{E}{-9} < c < 0.2$$

1 2 : Vt : Emacs: *eshell* AC | 06:22

U:@--- demo.c All of 324 (7,0) <N> (C +4 Helm MRev Projectile[pavpan] Abbrev)

```
#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;
}
```

U:@**- *eshell* All of 41 (3,13) <I> (EShell +4 Helm MRev Projectile[pavpan])

```
Welcome to the Emacs shell

~/mpi-talk $
```

1 2 : Vt : Emacs: ~/mpi-talk/demo.gh AC | 06:24

U:@--- demo.c All of 324 (6,0) <N> (C +4 Helm MRev Projectile[pavpan] Abbrev)

```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;
}
```

-:@--- demo.gh All of 480 (13,0) <N> (Fundamental +4 Helm MRev ARev Projectile[pavpan])

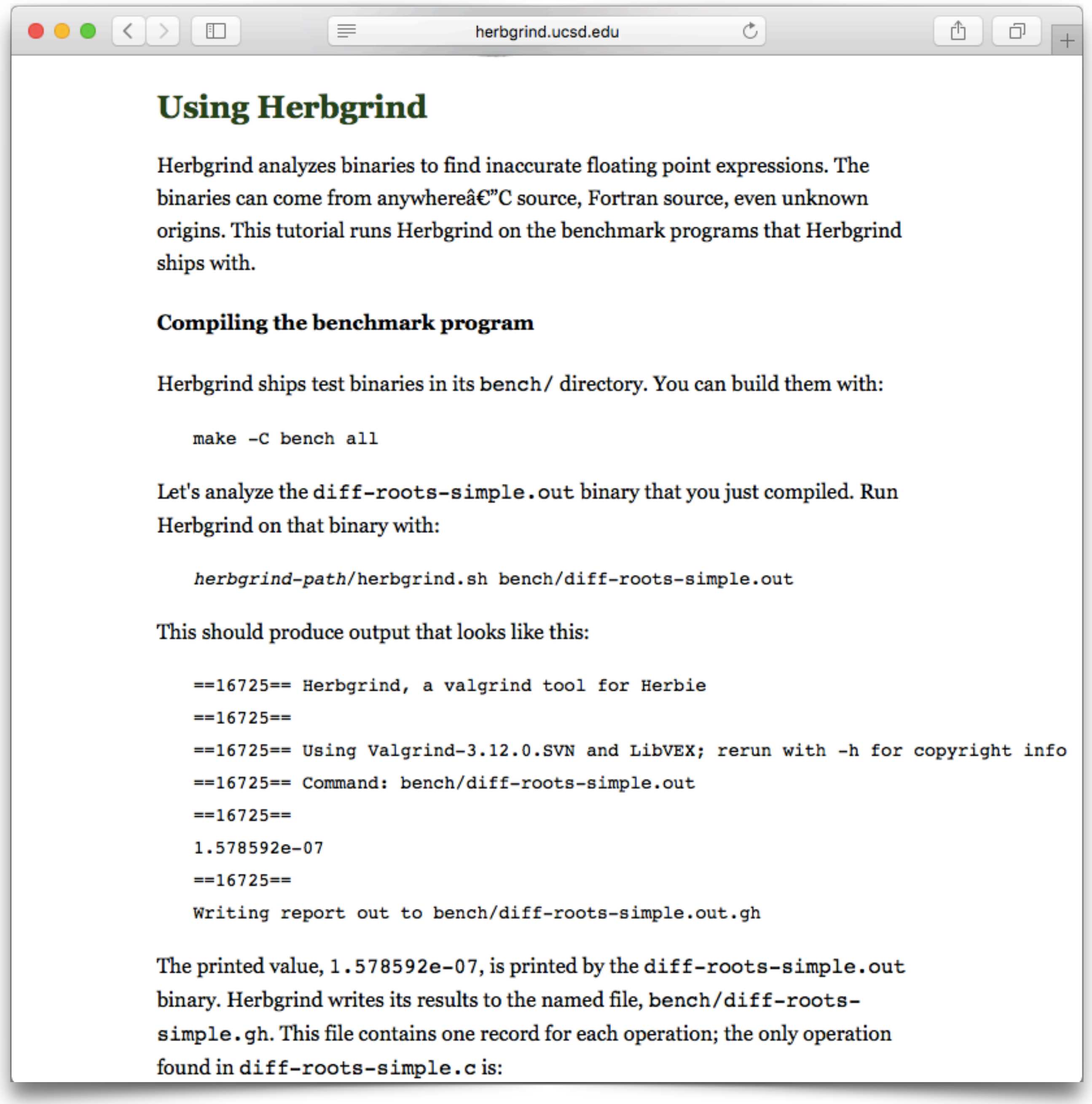
```
Result @ demo.c:13 in main (addr 400616)
47.750810 bits average error
64.000000 bits max error
Aggregated over 20 instances
Influenced by erroneous expression:

    (FPCore (x)
      (/ (- (sqrt (- (* x x) (* (* 4.000000 2.000000) 3.000000))) x) (+ 2.000000 2.000000)))
   demo.c:6 in solve_quadratic (addr 400599)
   47.750810 bits average error
   64.000000 bits max error
   32.000000 bits average local error
   64.000000 bits max local error
   Aggregated over 20 instances
```

To be continued…

Quit

Herbgrind Implementation

http://herbgrind.ucsd.edu



herbgrind.ucsd.edu

Using Herbgrind

Herbgrind analyzes binaries to find inaccurate floating point expressions. The binaries can come from anywhereâ€"C source, Fortran source, even unknown origins. This tutorial runs Herbgrind on the benchmark programs that Herbgrind ships with.

Compiling the benchmark program

Herbgrind ships test binaries in its bench/ directory. You can build them with:

```
make -C bench all
```

Let's analyze the `diff-roots-simple.out` binary that you just compiled. Run Herbgrind on that binary with:

```
herbgrind-path/herbgrind.sh bench/diff-roots-simple.out
```

This should produce output that looks like this:

```
==16725== Herbgrind, a valgrind tool for Herbie
==16725==
==16725== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright info
==16725== Command: bench/diff-roots-simple.out
==16725==
1.578592e-07
==16725==
Writing report out to bench/diff-roots-simple.out.gh
```

The printed value, `1.578592e-07`, is printed by the `diff-roots-simple.out` binary. Herbgrind writes its results to the named file, `bench/diff-roots-simple.gh`. This file contains one record for each operation; the only operation found in `diff-roots-simple.c` is:

~ 11 KLOC in C

Herbgrind Case Studies

GROMACS
FAST. FLEXIBLE. FREE.

Found confirmed bug in dihedral angle routine.

Triangle

Few false positives; detected most tricks.

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*

Outline

Herbgrind: Finding error in large applications

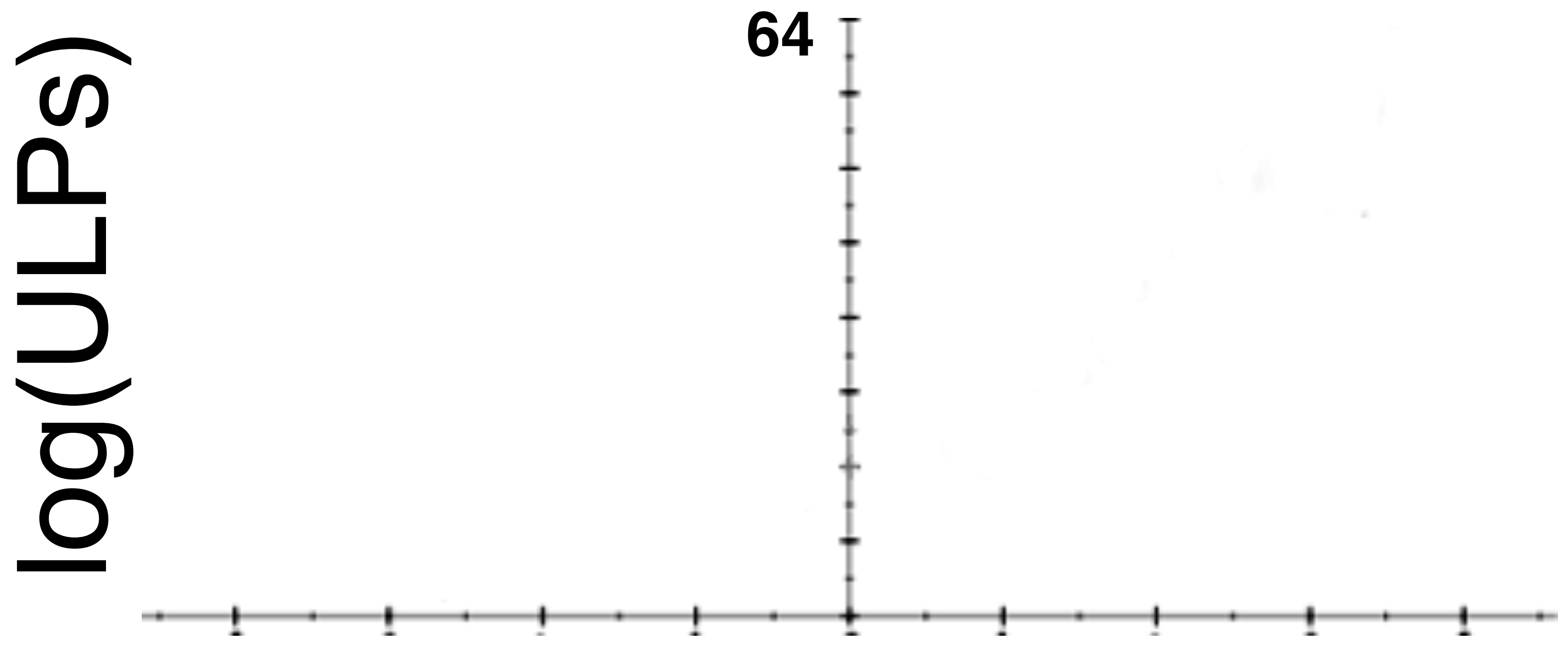
Herbie: Automatically improving accuracy

FPBench: A standard format for composing tools

Titanic: A laboratory for exploring number systems

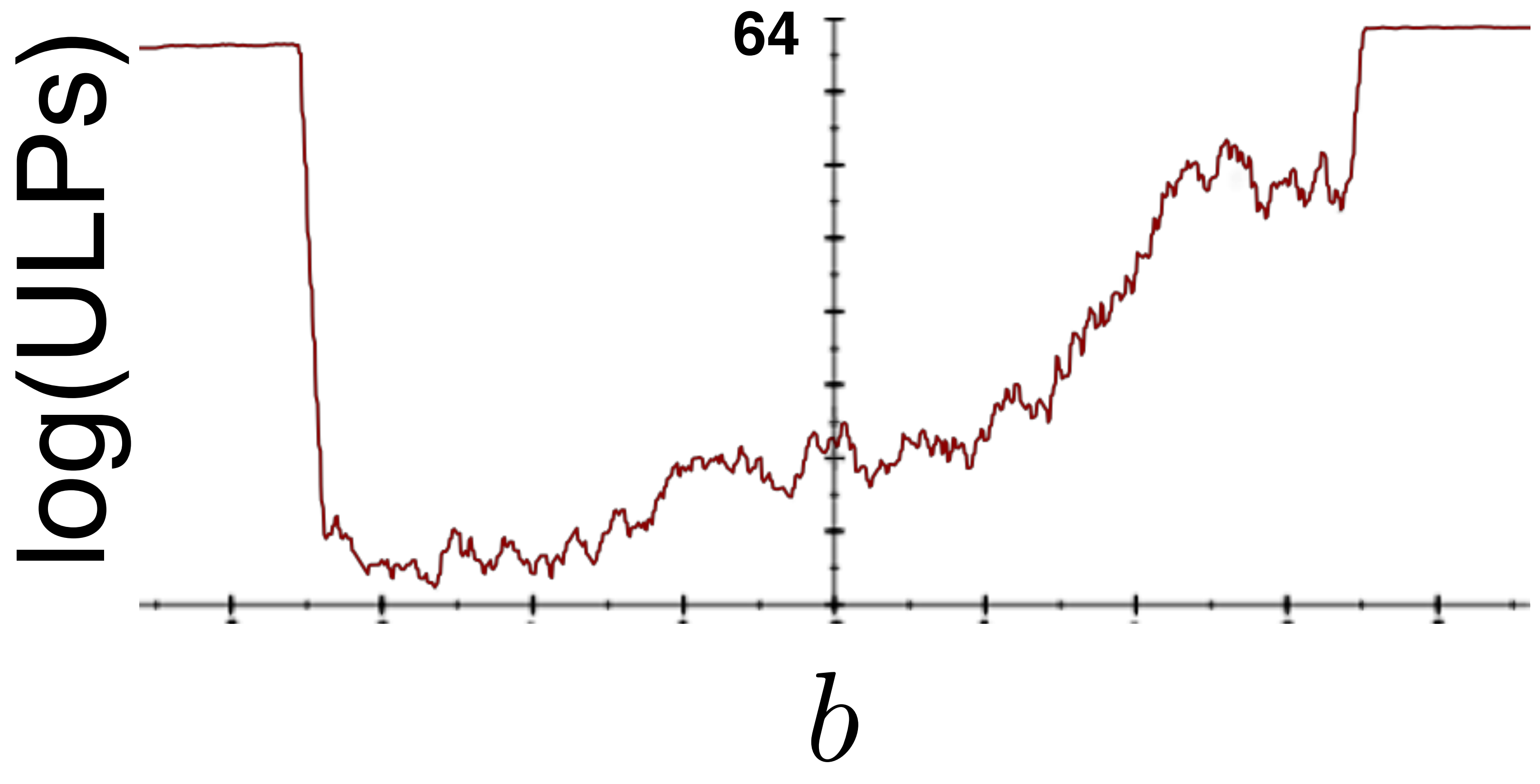
Rounding Error in Quadratic

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



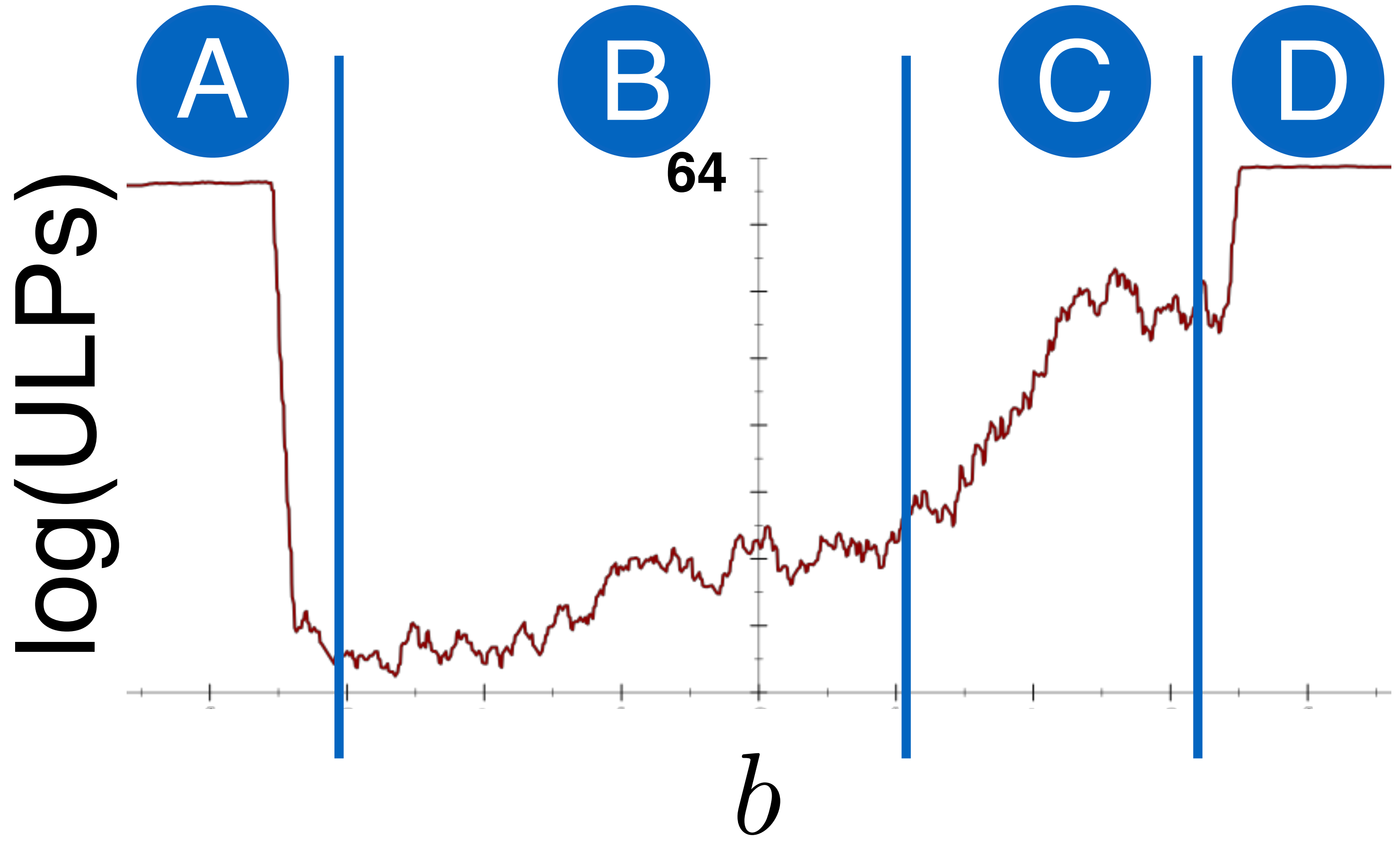
Rounding Error in Quadratic

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



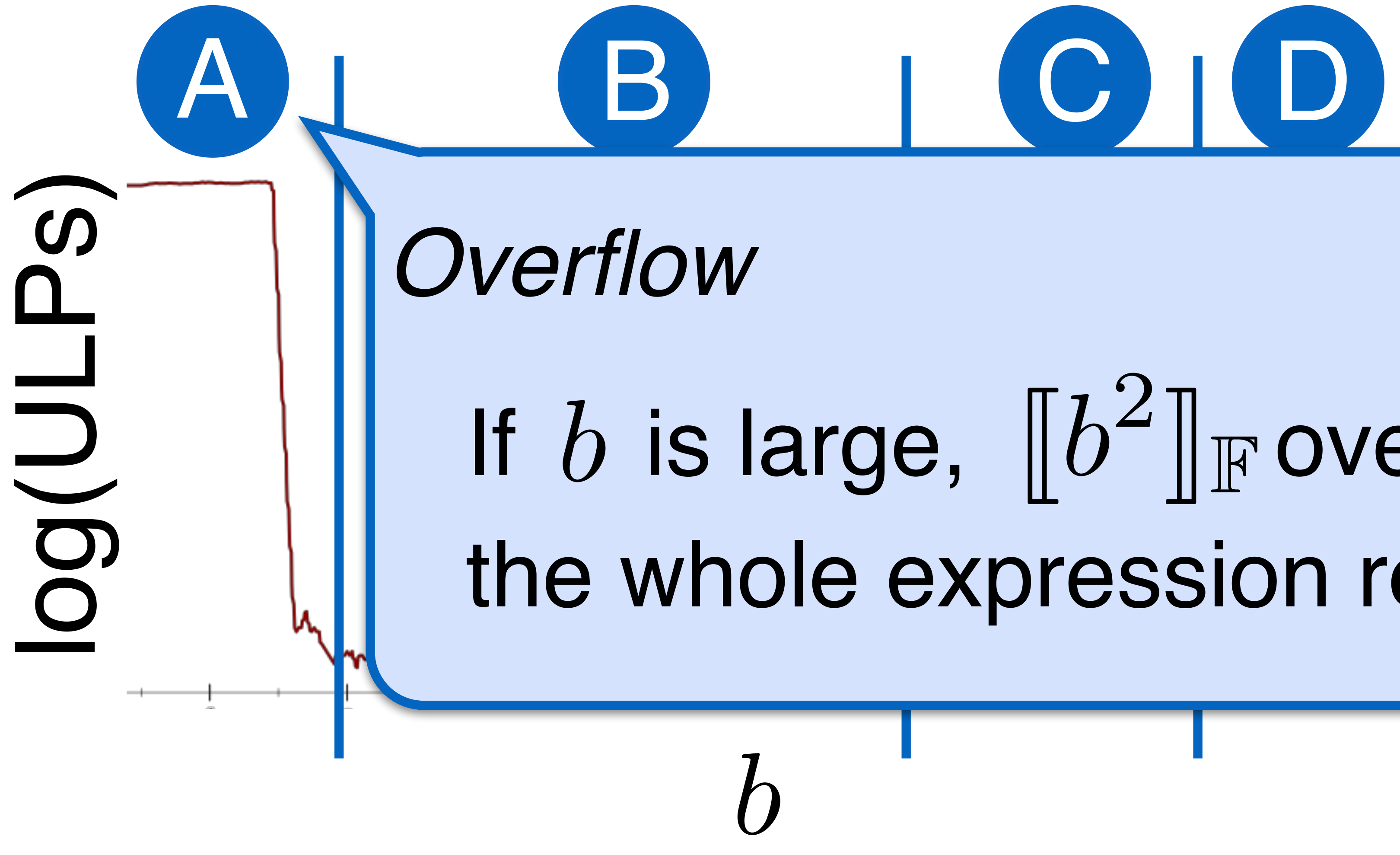
Rounding Error in Quadratic

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



Rounding Error in Quadratic

$$\frac{-b+\sqrt{b^2-4ac}}{2a} \Rightarrow \left\{ \vphantom{\begin{array}{c} a \\ b \\ c \\ d \end{array}} \right.$$

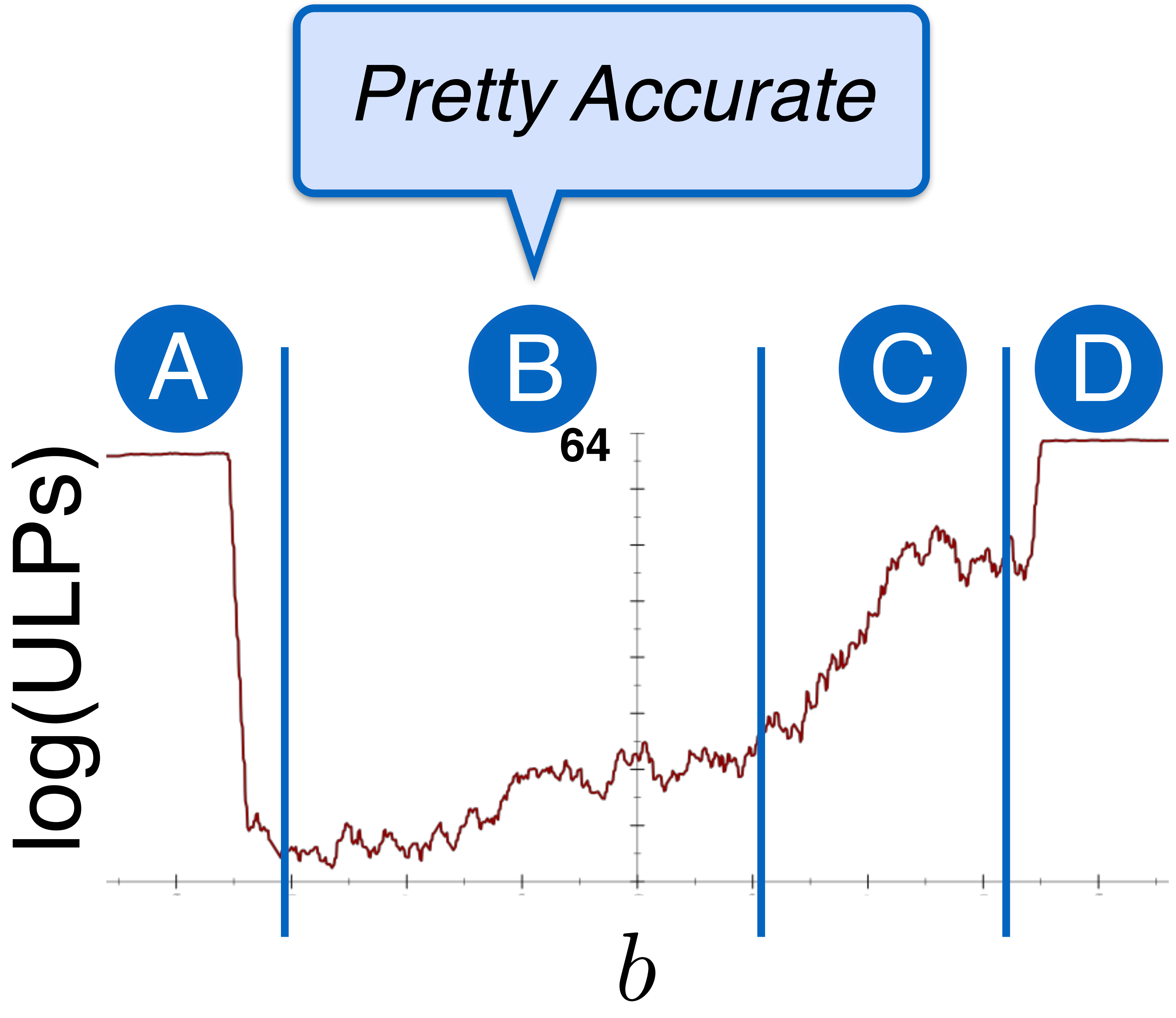


Overflow

If b is large, $[\![b^2]\!]_{\mathbb{F}}$ overflows and the the whole expression returns ∞.

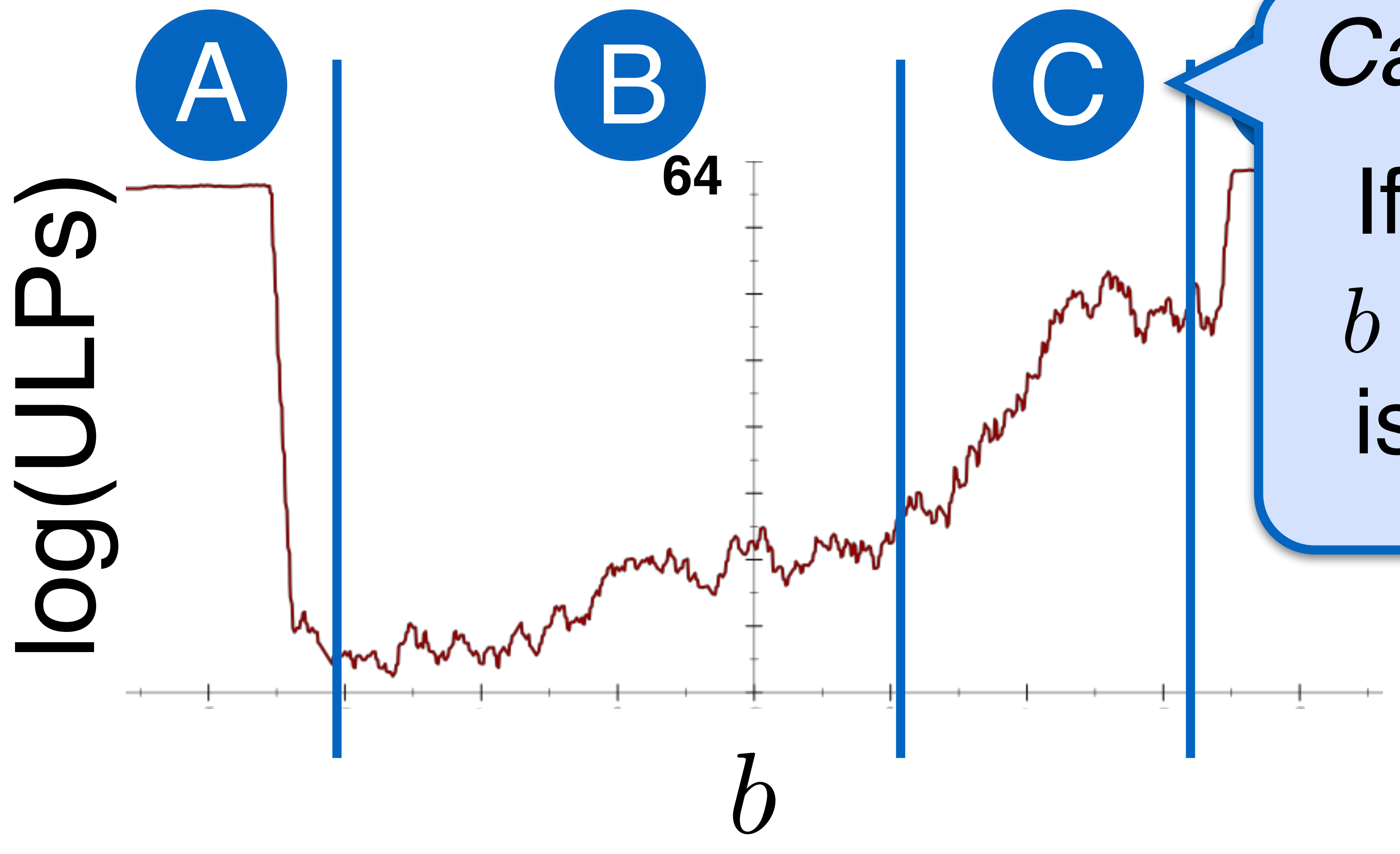
Rounding Error in Quadratic

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



Rounding Error in Quadratic

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

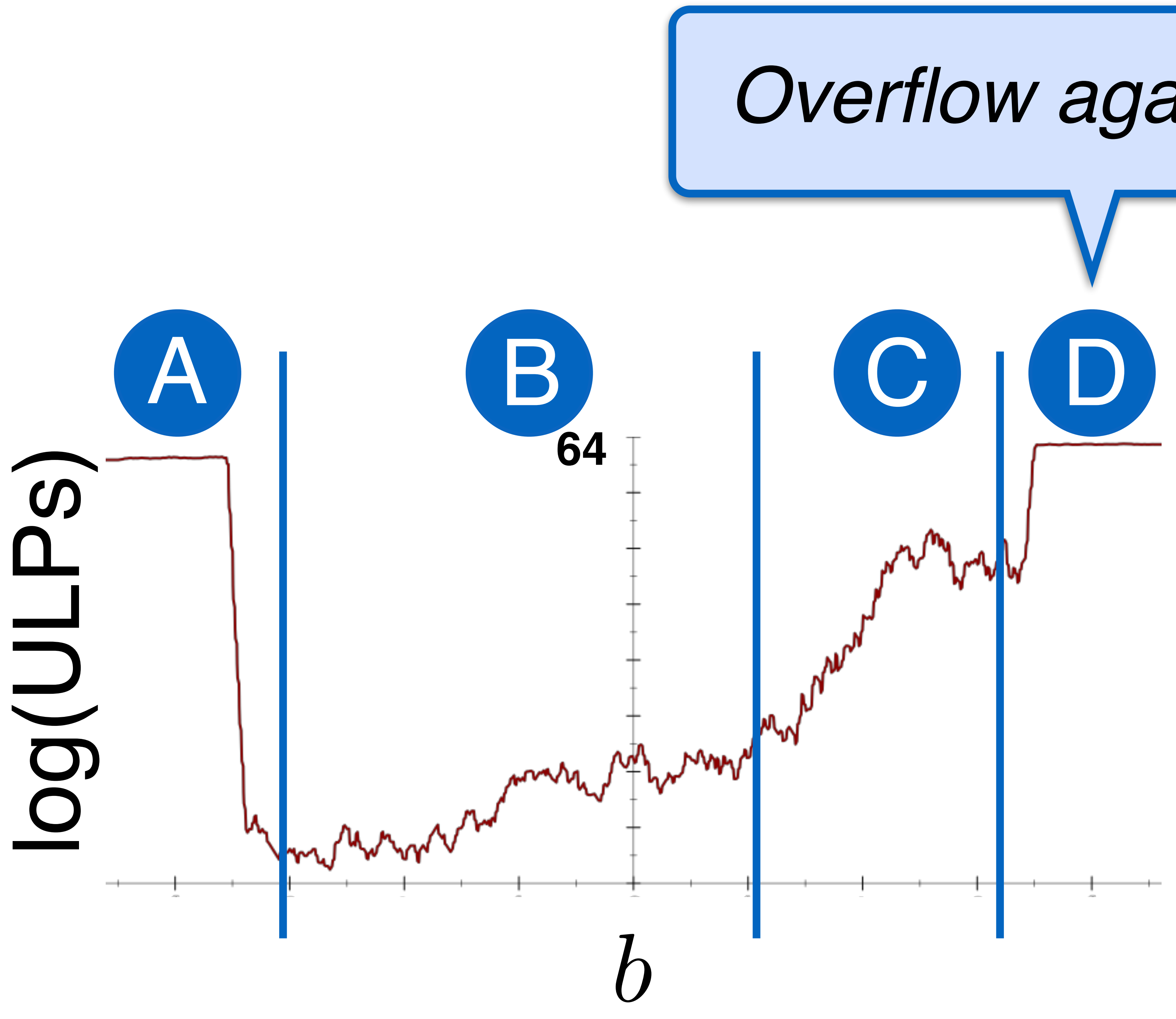


Catastrophic Cancellation

If b is large, but a and c are small, $b \approx \sqrt{b^2 - 4ac}$ and the difference is rounded off.

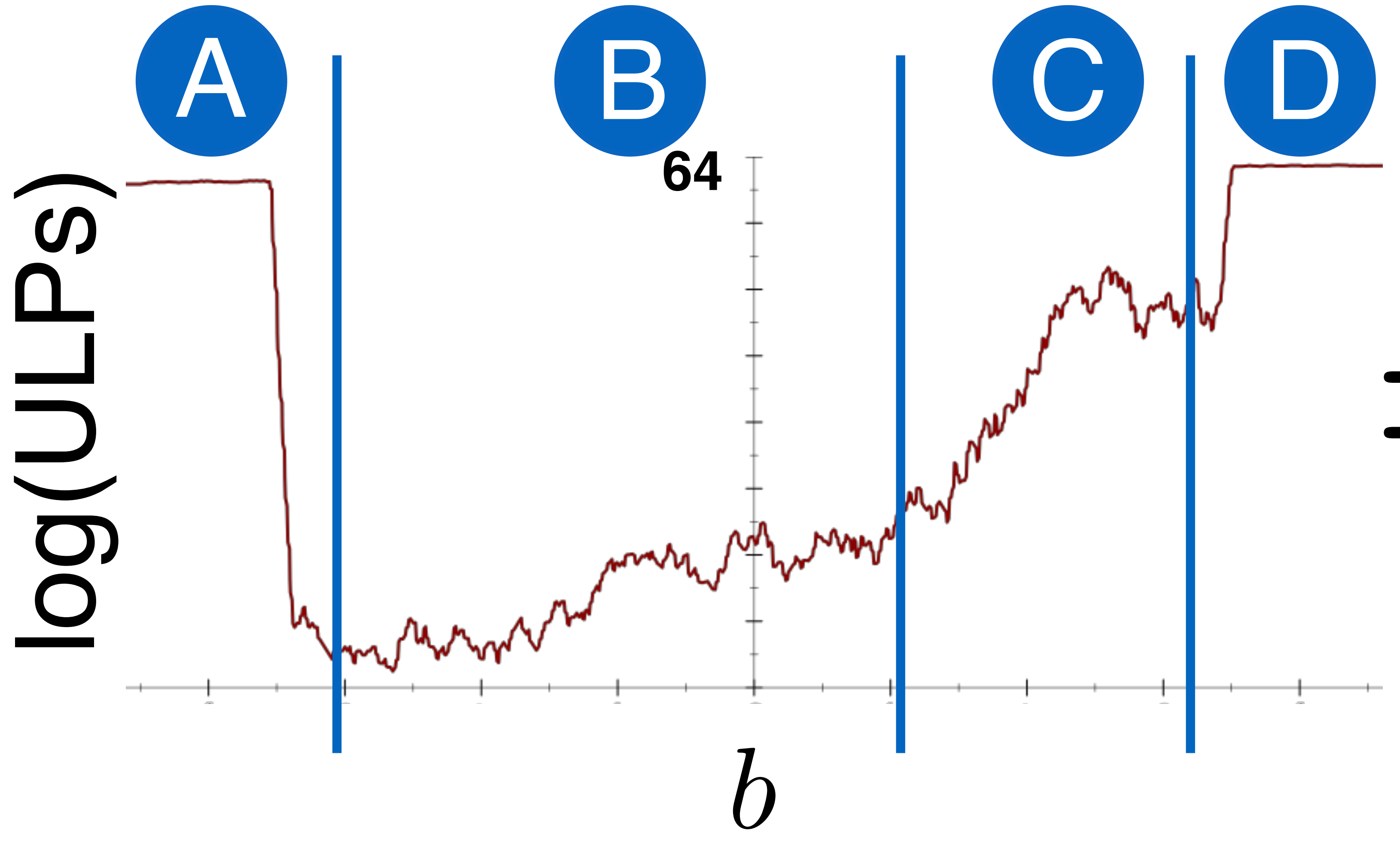
Rounding Error in Quadratic

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

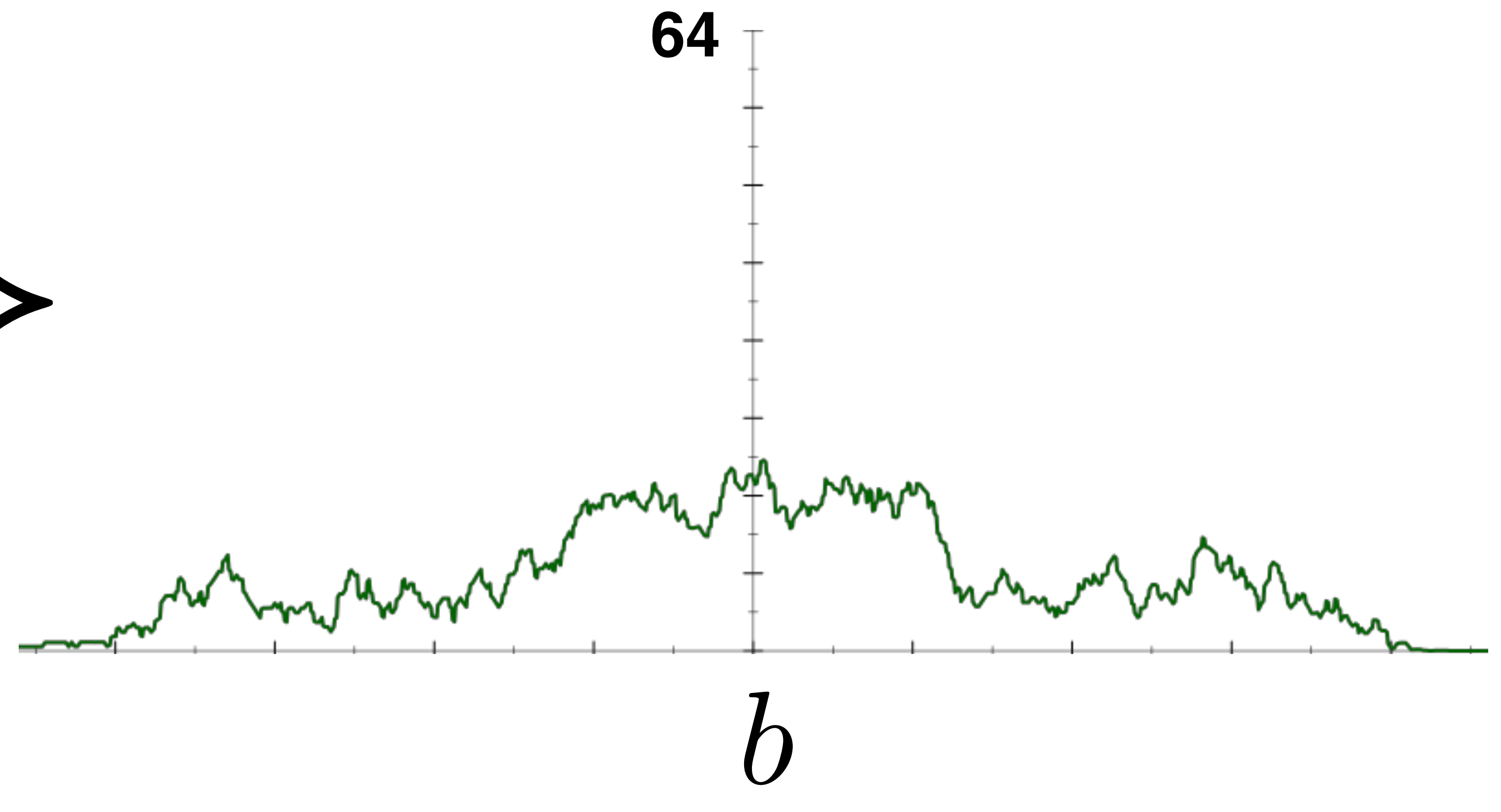


Rounding Error in Quadratic

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \Rightarrow \quad \begin{cases} \frac{c}{b} - \frac{b}{a} & \text{if } b \in \text{A} \\ \frac{-b+\sqrt{b^2-4ac}}{2a} & \text{if } b \in \text{B} \\ \frac{2c}{-b-\sqrt{b^2-4ac}} & \text{if } b \in \text{C} \\ -\frac{c}{b} & \text{if } b \in \text{D} \end{cases}$$



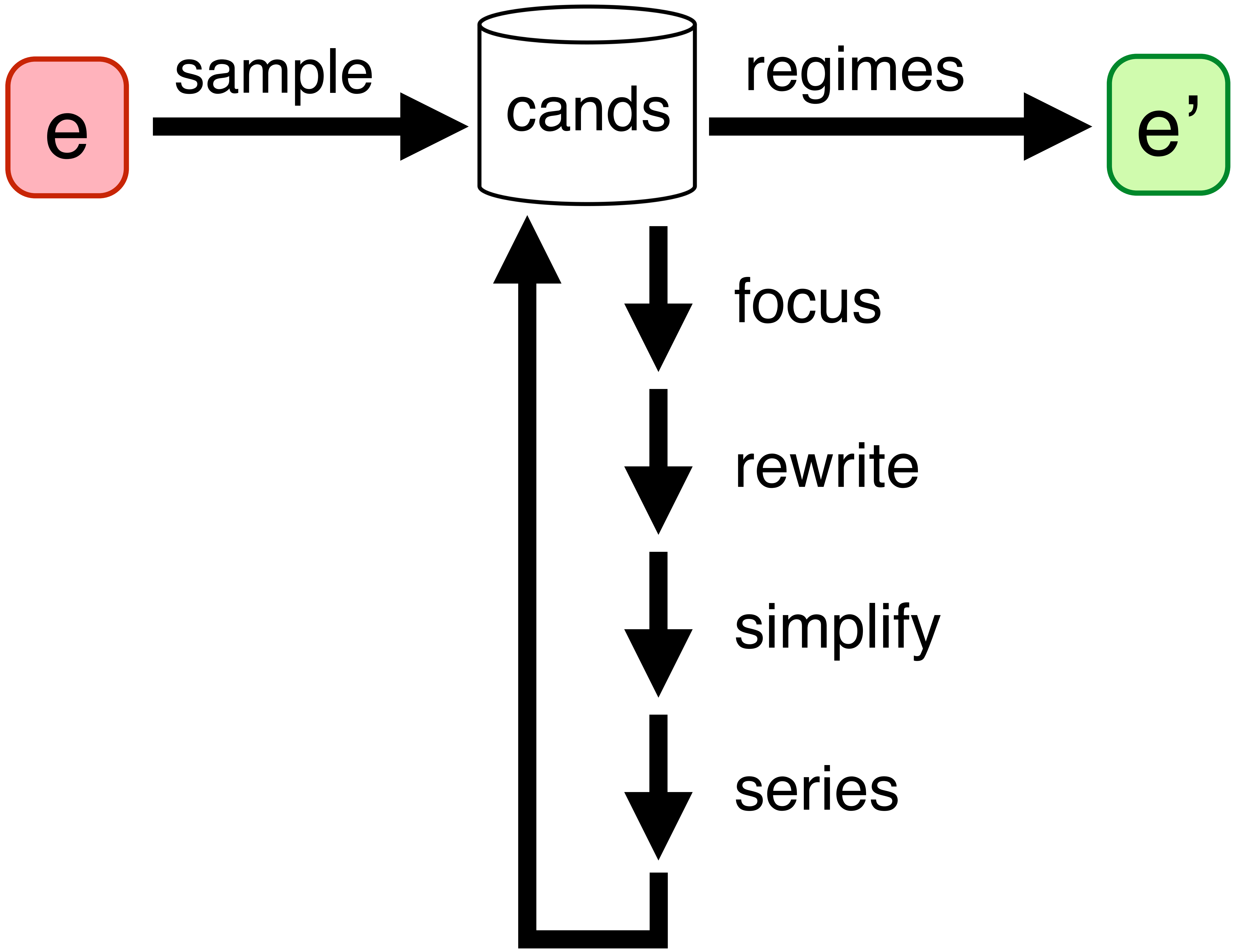
$\Rightarrow$



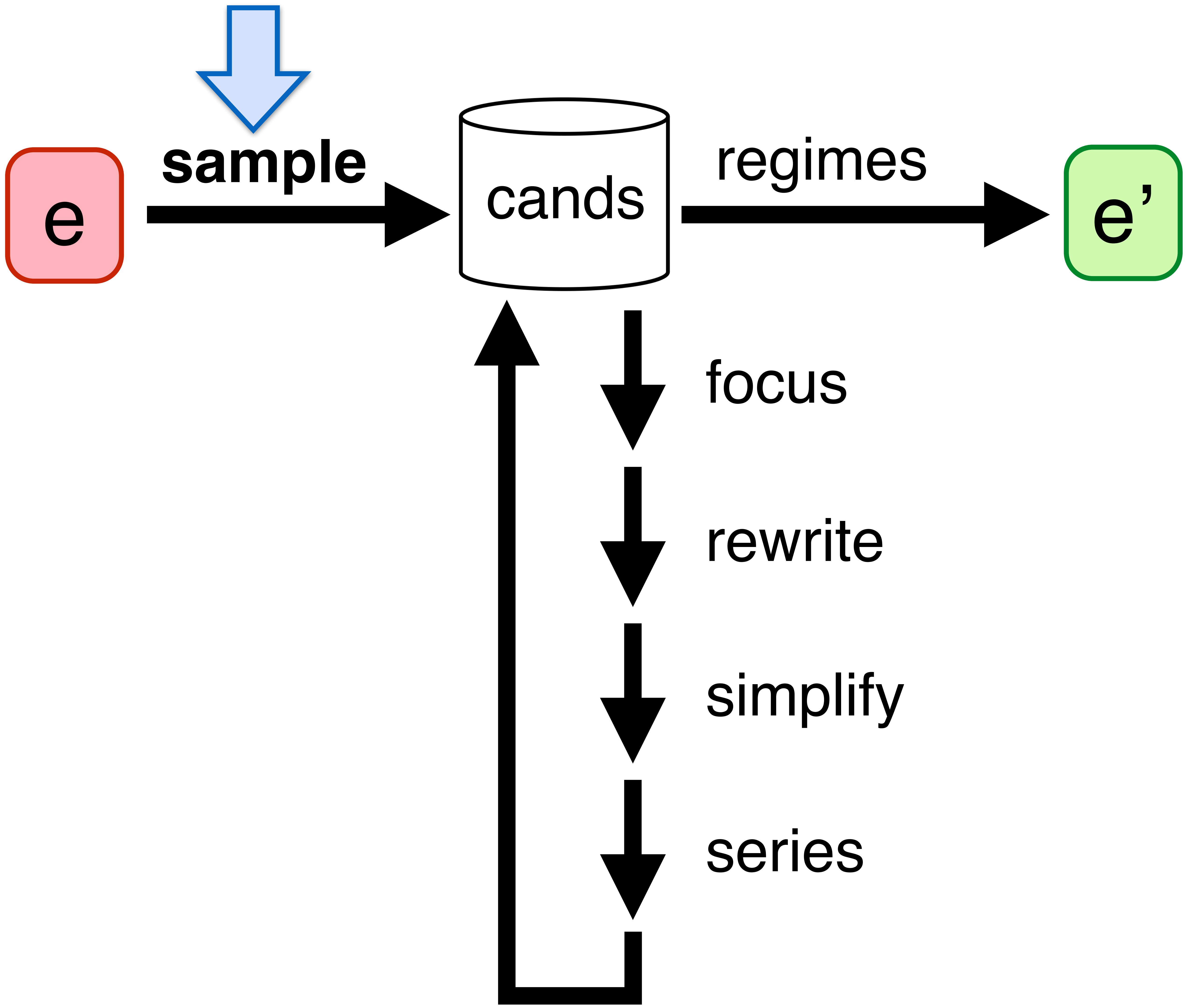
Goal: Automatically improve floating point accuracy

[*Distinguished Paper,* PLDI 15]

Herbie Architecture



Herbie Architecture



Determine ground truth

HERBIE

$X = \mathsf{sample}(\mathsf{domain}(e))$

64 random bits

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

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

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

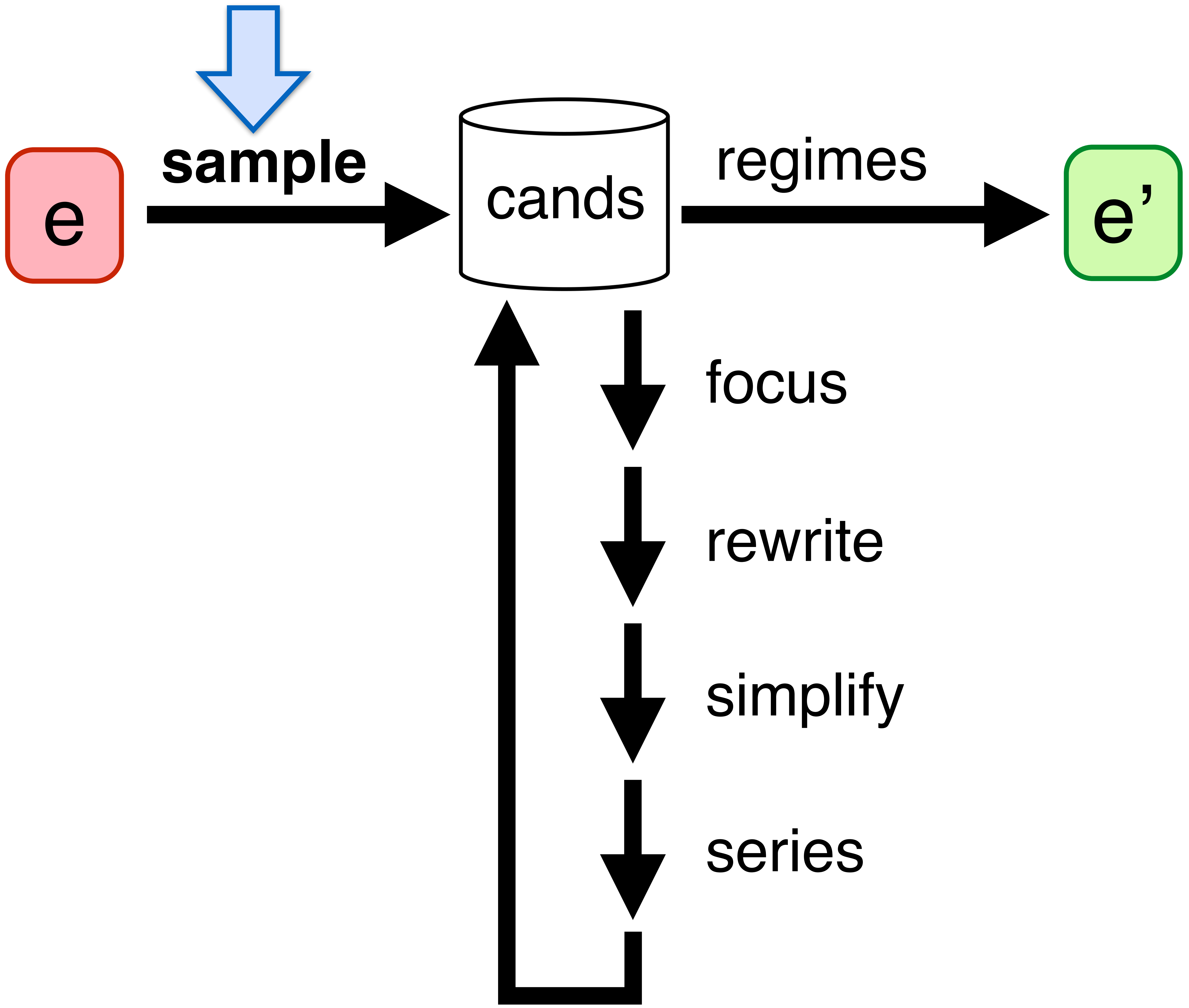
Get 64-bit prefix
with MPFR.
Subtle! See paper.

Compute in $\mathbb{F}$

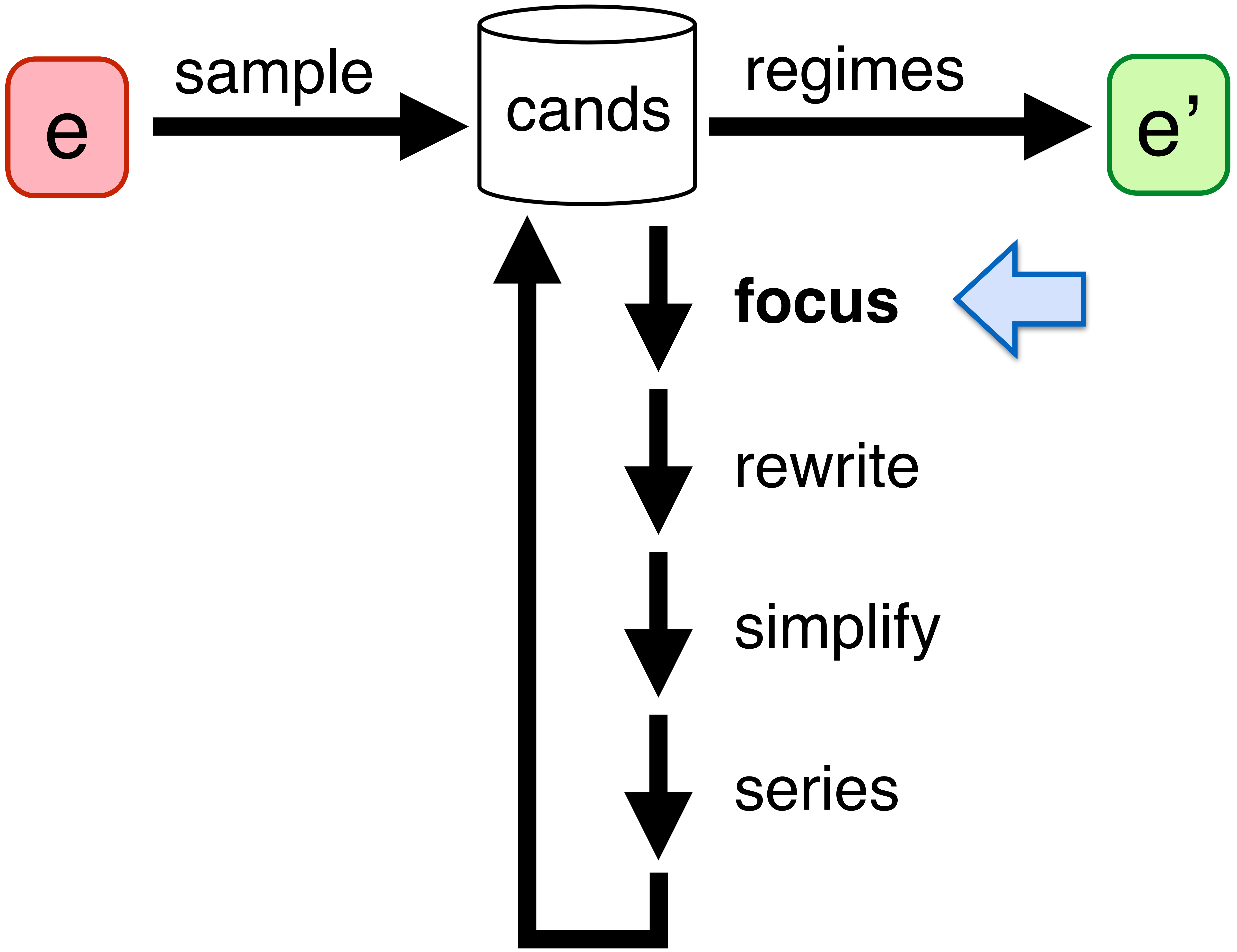
error

e.g. $\{13.2\text{b}, 51.7\text{b}, 1\text{b}, \ldots\}$

Herbie Architecture



Herbie Architecture



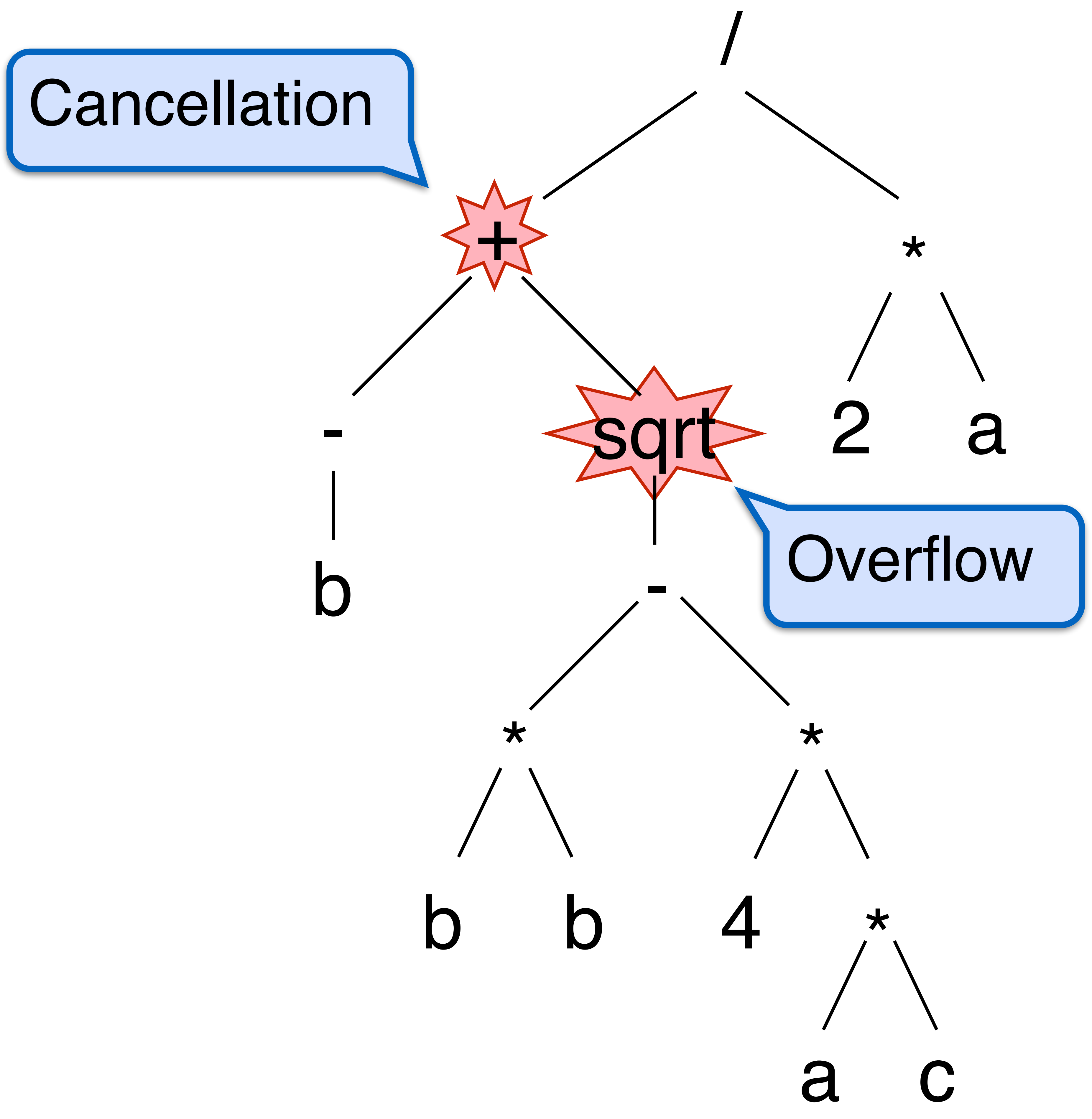
Focus: Localize error

Compute local errors

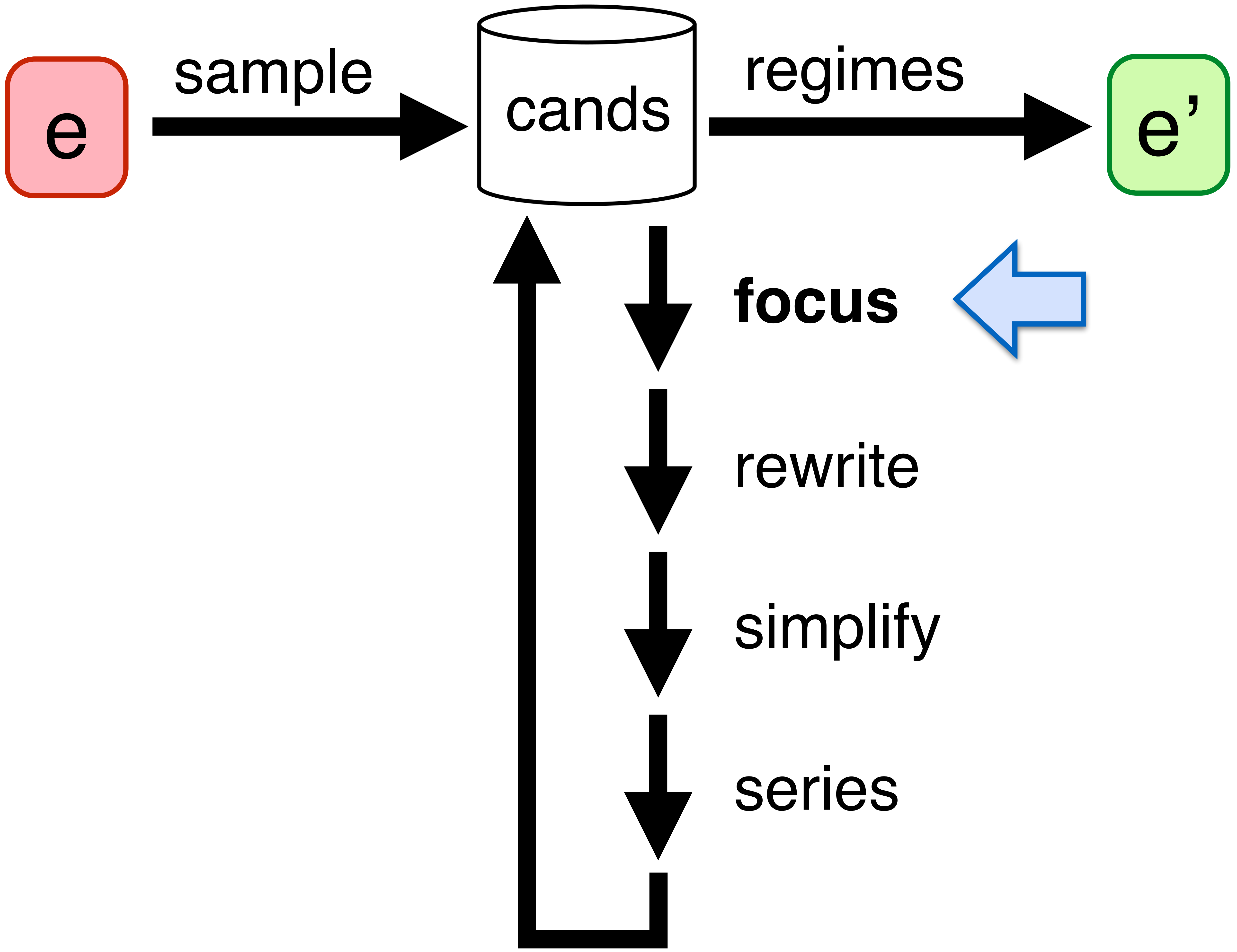
- start at leaves
- work bottom-up

Focus on biggest

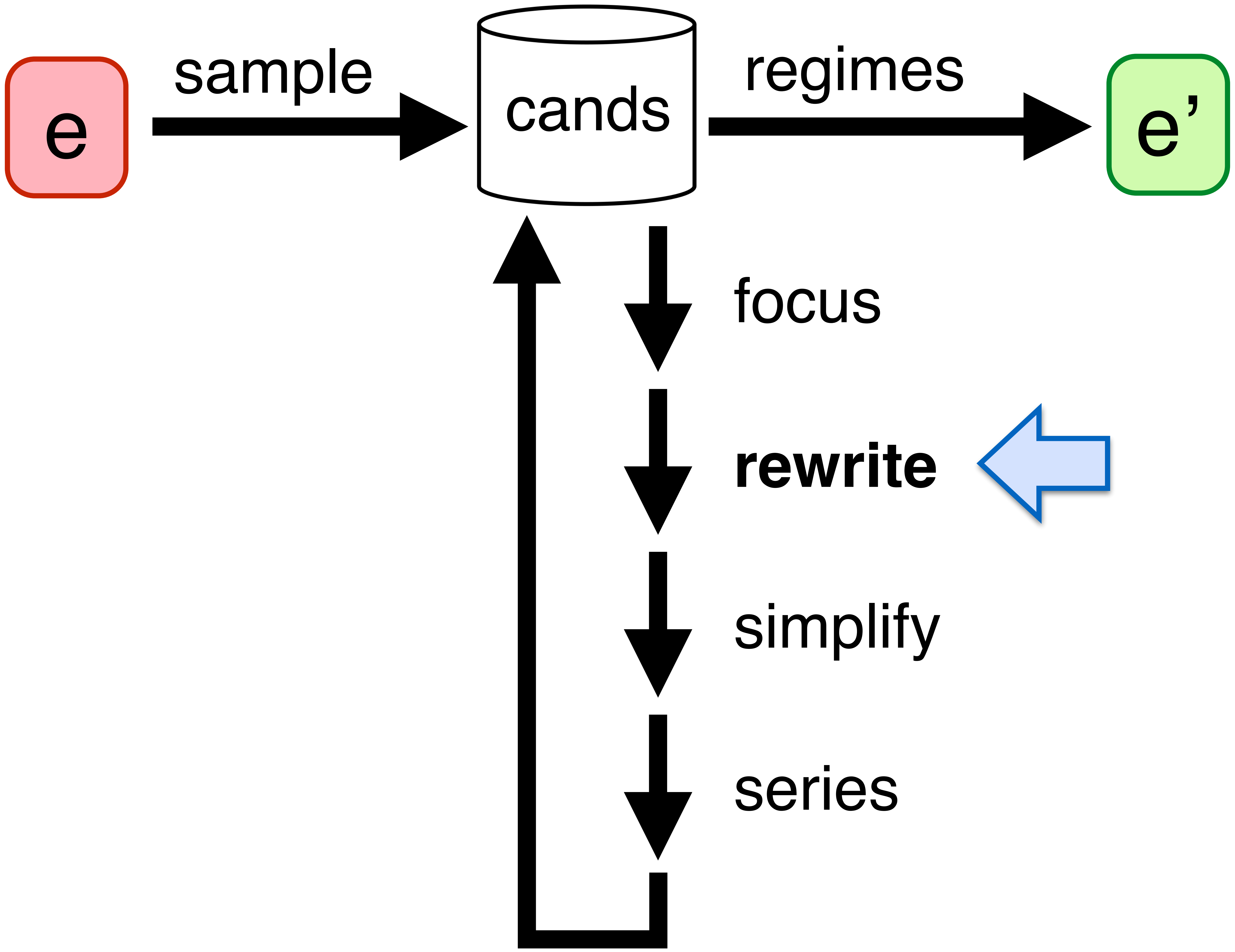
- High local error = likely root cause
- Needs fixing; focus for rewrites



Herbie Architecture



Herbie Architecture



Rewrite: Generate Candidates

Apply rewrites to ✷

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

$-x \rightsquigarrow 0 - x$
$x + y \rightsquigarrow \dfrac{x^2 - y^2}{x - y}$
$(x - y) + z \rightsquigarrow x - (y - z)$
… 180 more …

Rule DB

Difference of squares

Associate

trig, algebra, etc.

Rewrite: Generate Candidates

Apply rewrites to

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

$$-x \rightsquigarrow 0 - x$$

$$x + y \rightsquigarrow \frac{x^2 - y^2}{x - y}$$

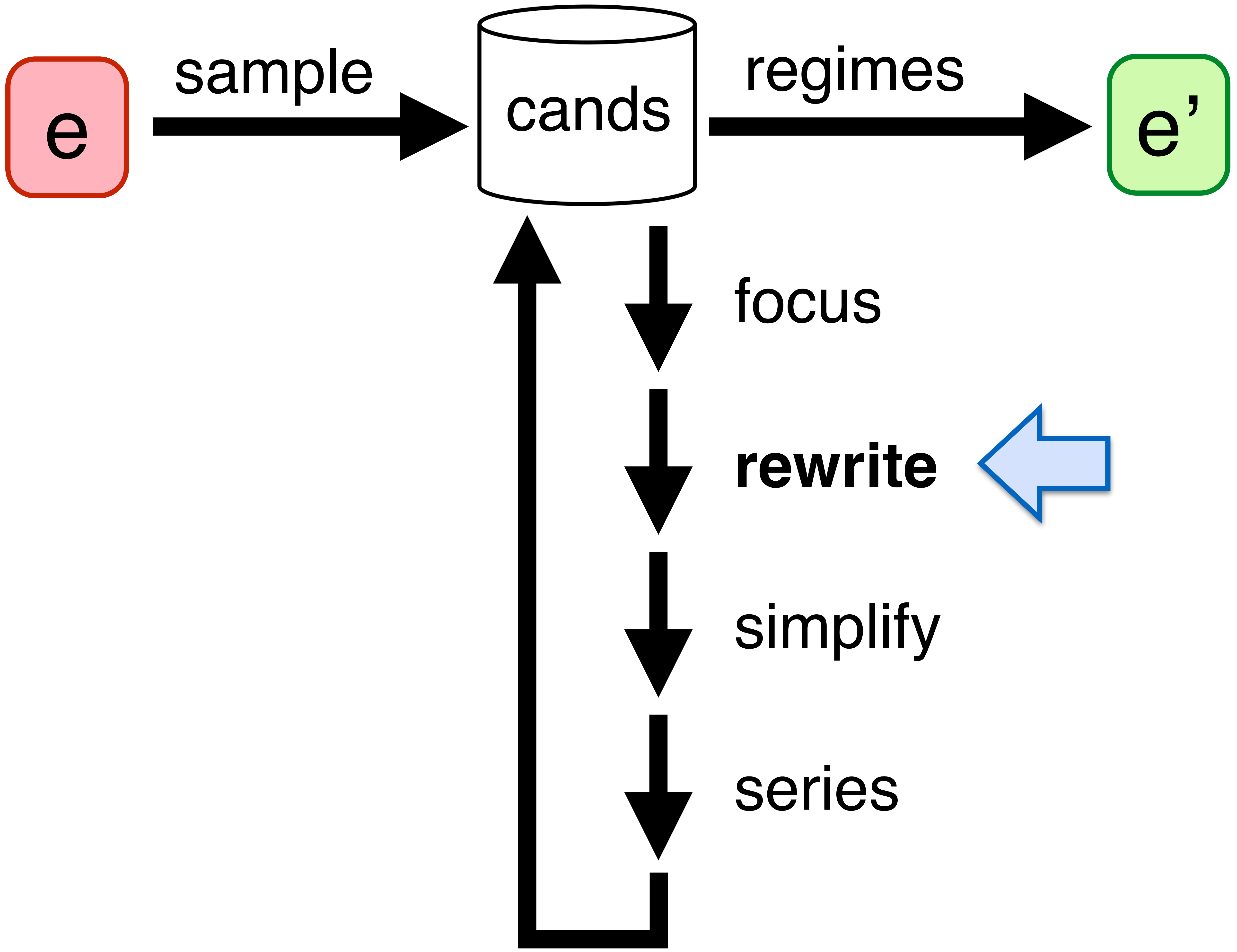
$$(x - y) + z \rightsquigarrow x - (y - z)$$

… 180 more …

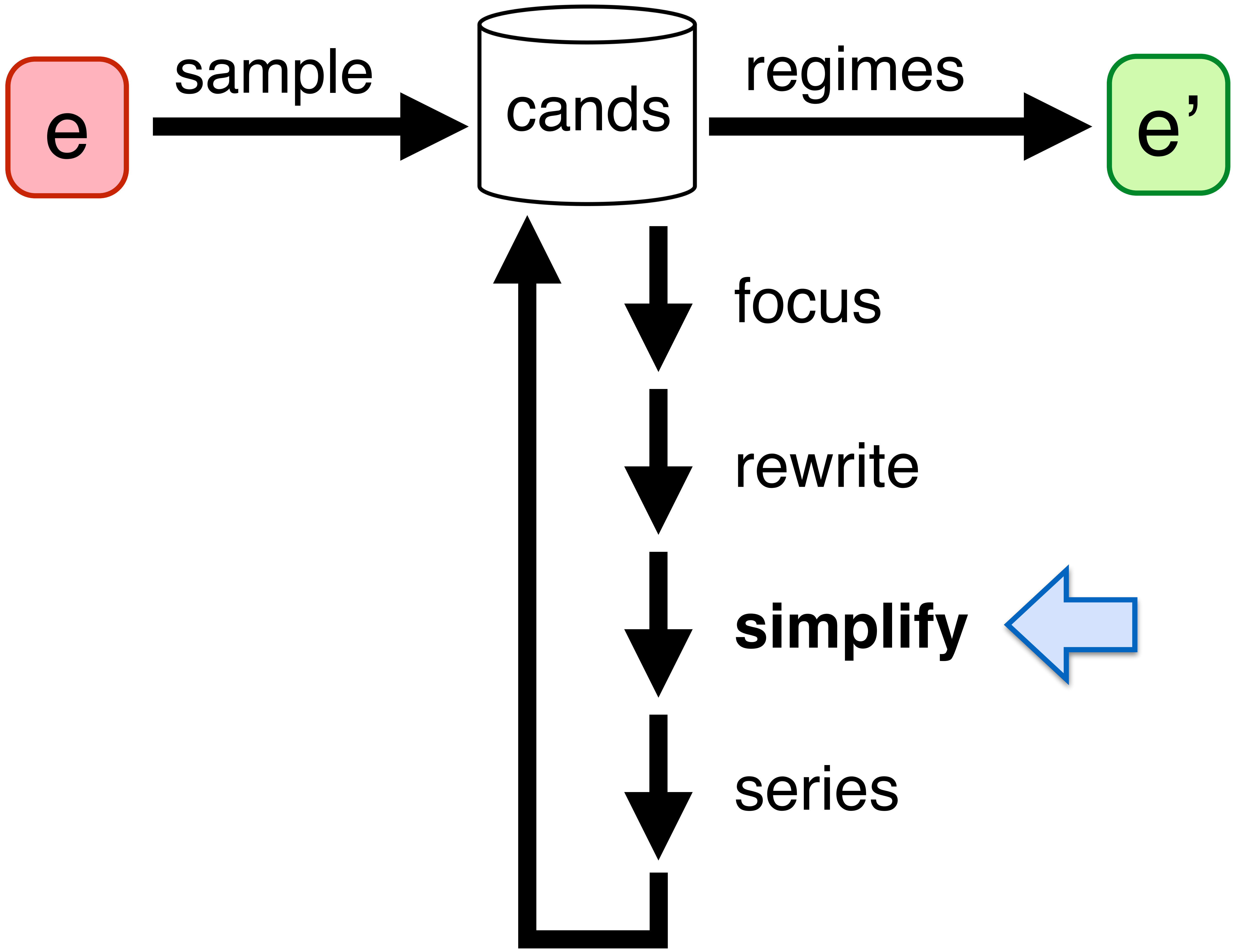
Rule DB

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

Herbie Architecture



Herbie Architecture



Simplify Expressions

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

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

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

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

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

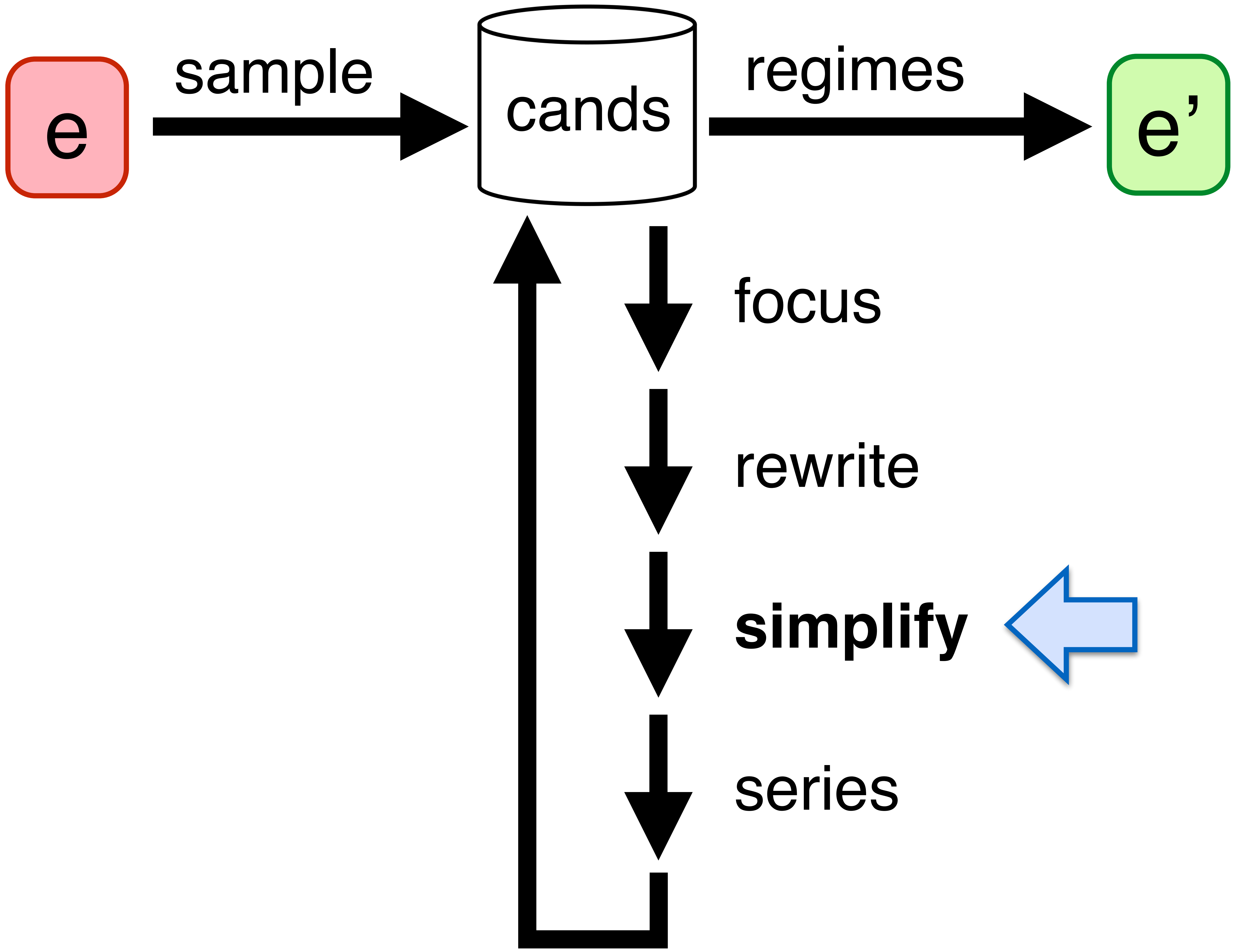
Tricky! *[Caviness '70]*

- many possible rewrites
- "simpler" not always clear
- huge search space
- avoid undoing progress!

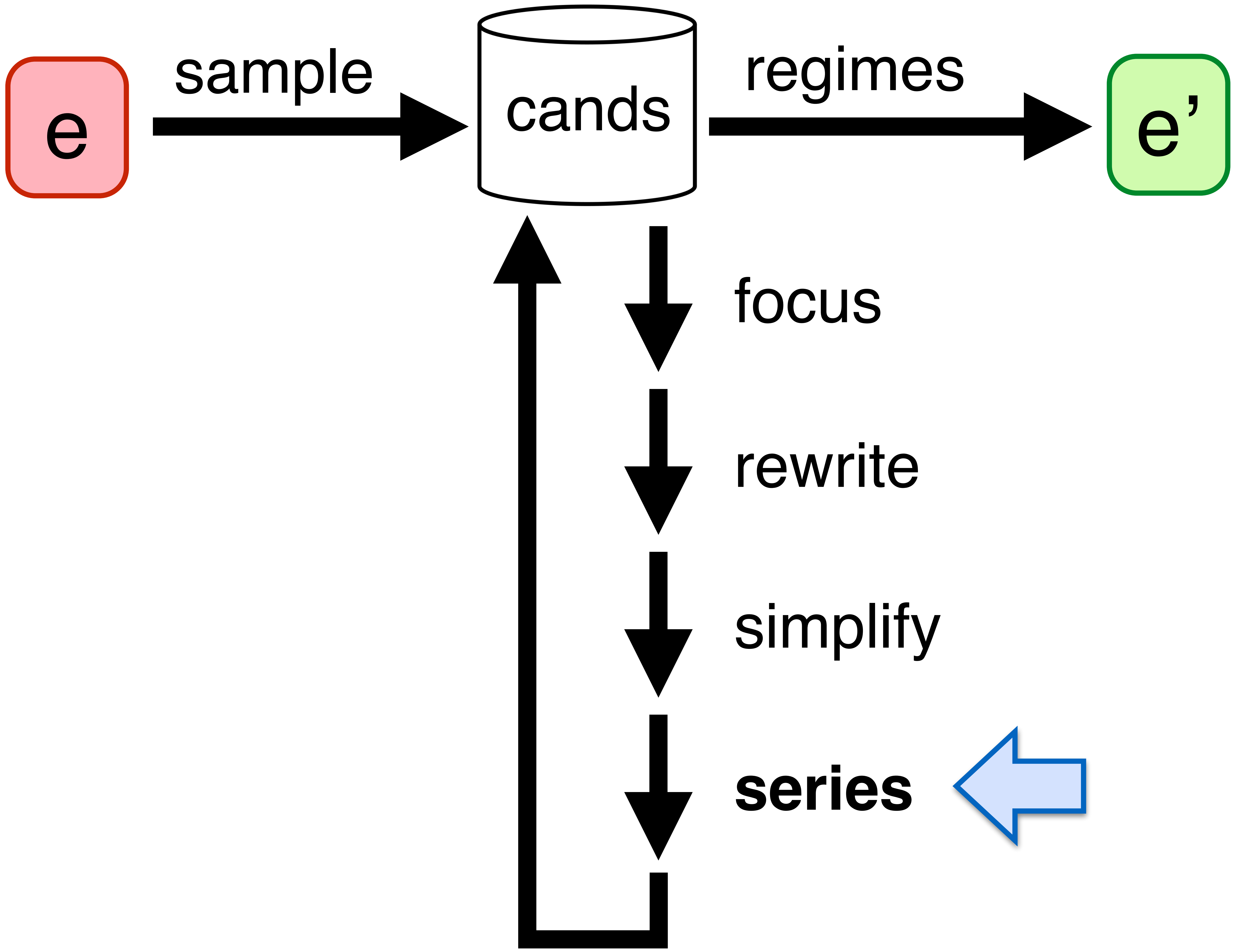
E-graphs *[Nelson '79]*

- track equiv classes
- restrict rewrites
- select smallest AST

Herbie Architecture



Herbie Architecture



Series Expansions

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

$$= \frac{-b + b\sqrt{1 - 4ac/b^2}}{2a}$$

$$\approx \frac{-b + b(1 - 4ac/2b^2)}{2a}$$

$$= \frac{-4ac/2b}{2a}$$

$$= -\frac{c}{b}$$

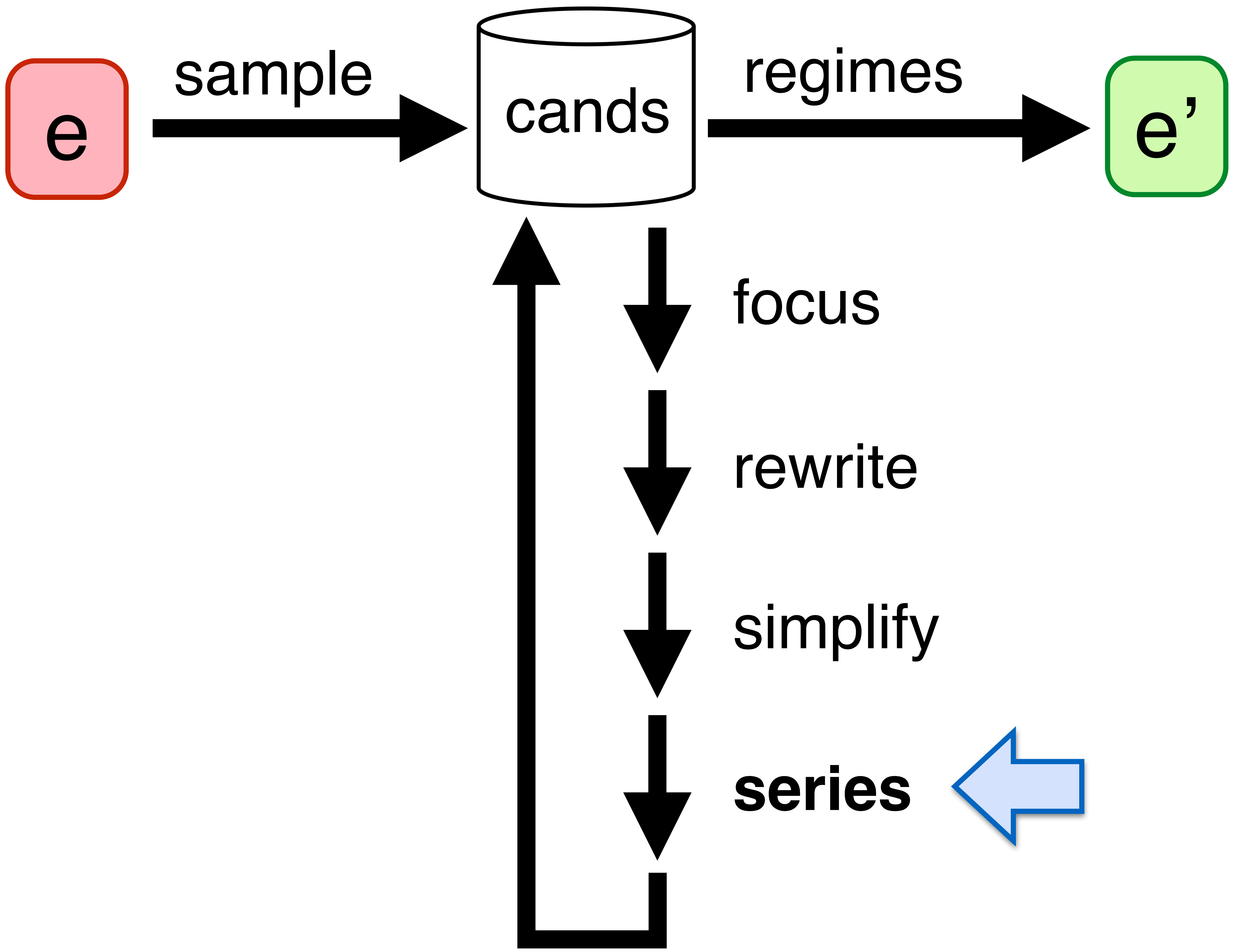
$b > 0 \to \infty$

$\sqrt{1 - x} \approx 1 - x/2$

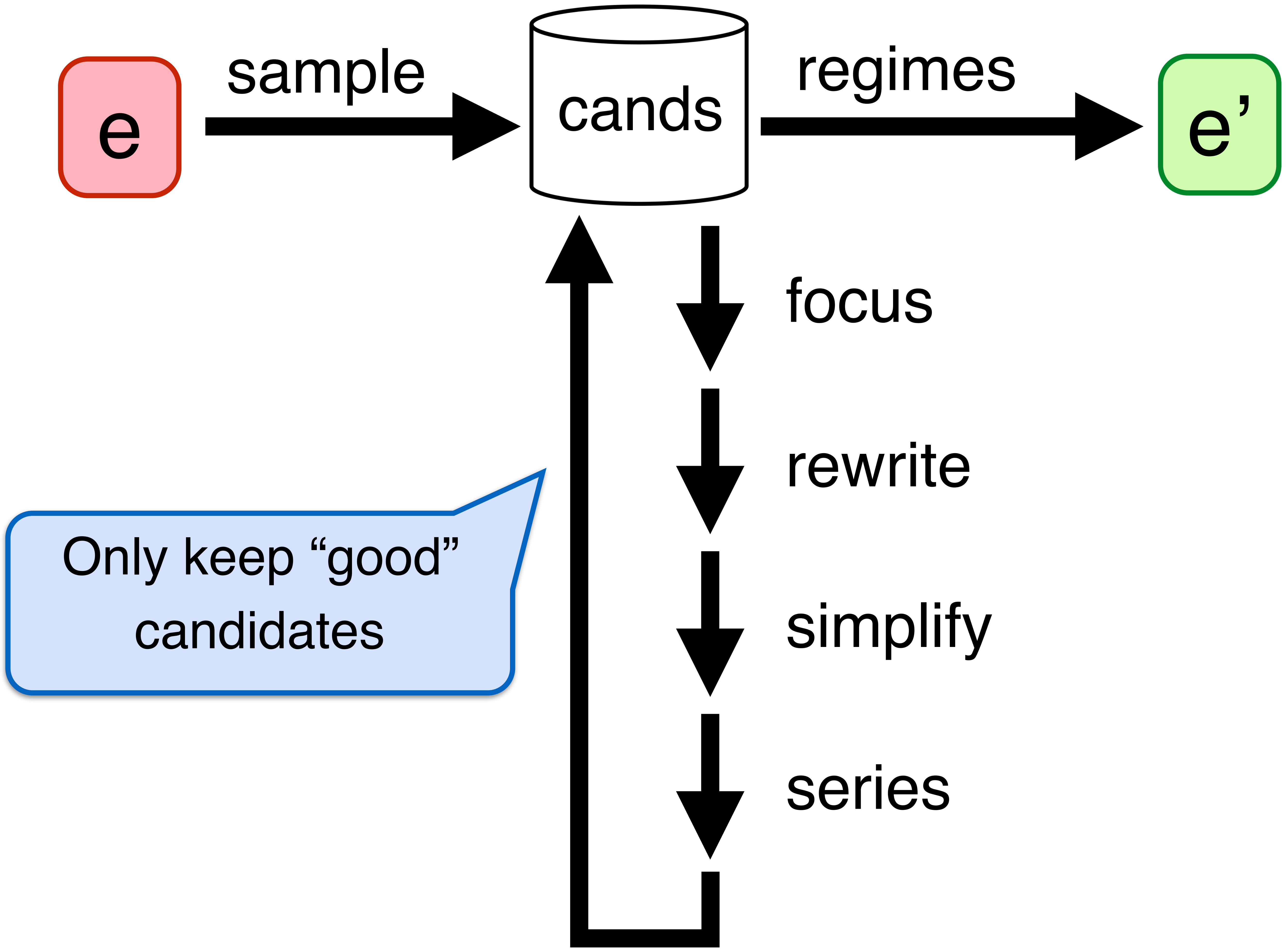
Custom series expander:

- auto expands diverse exprs
- determines # terms to take
- expand around arbitrary pt

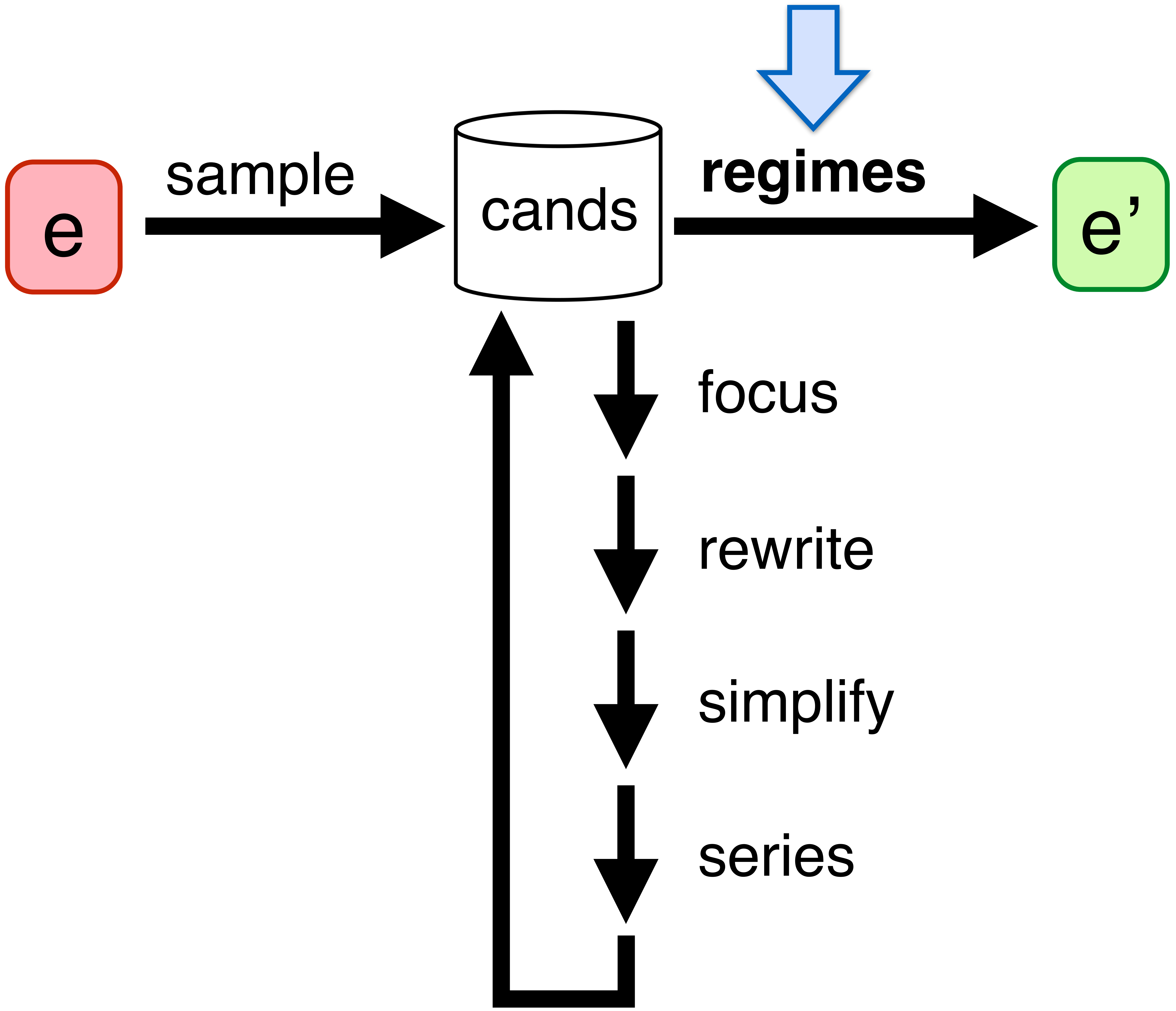
Herbie Architecture



Herbie Architecture

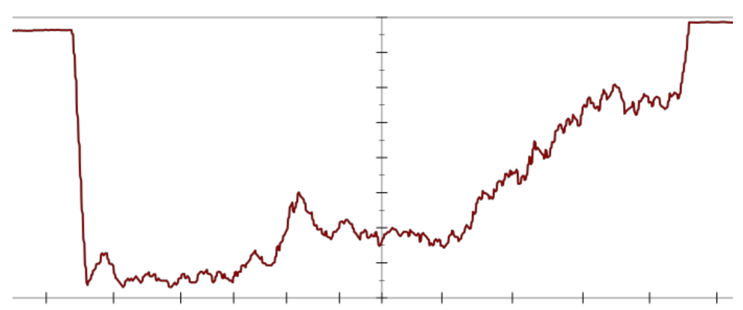


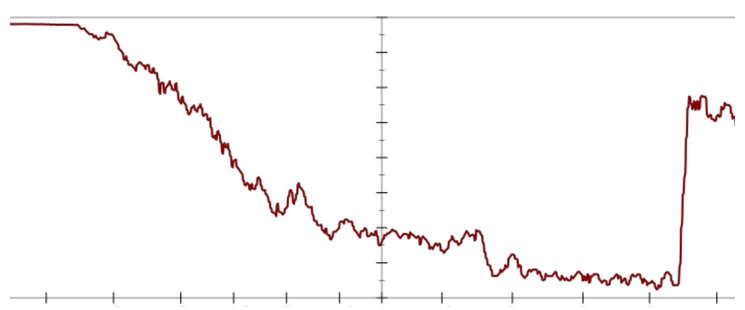
Herbie Architecture



Regime Inference

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

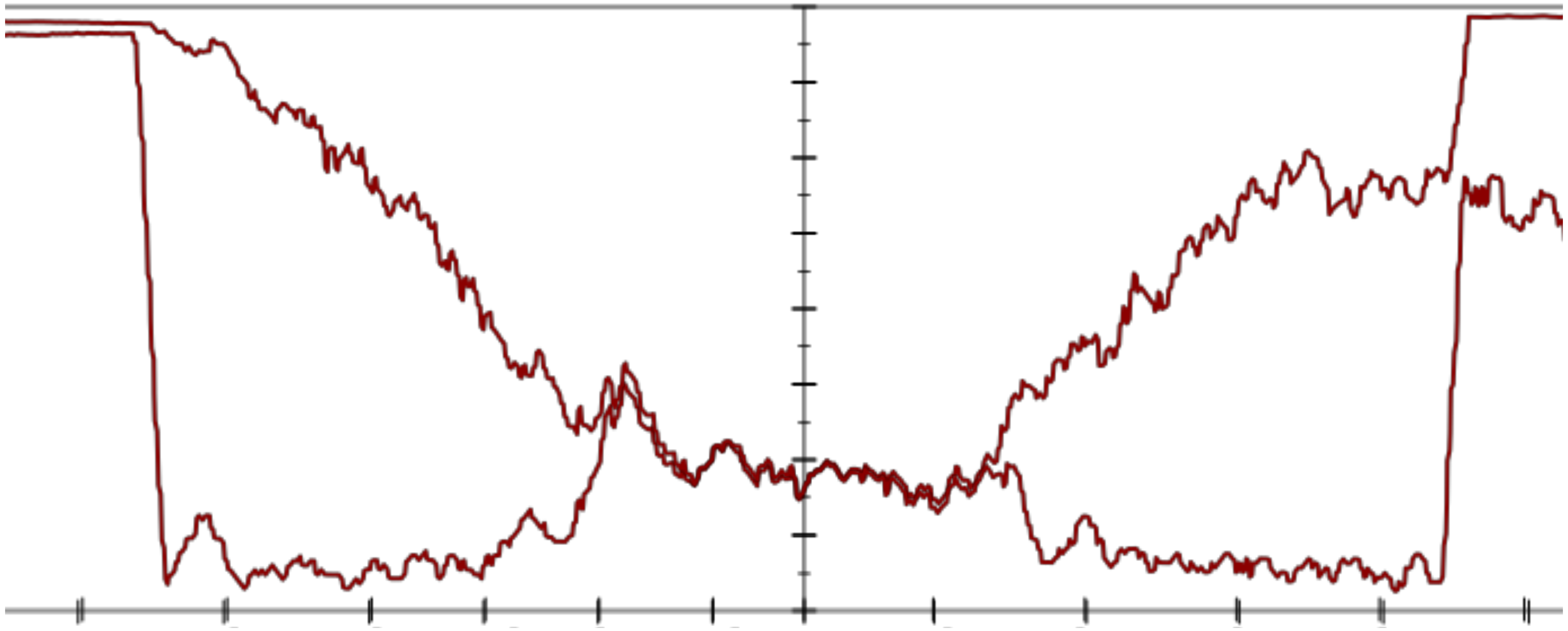




Try each variable;
keep best one

Regime Inference

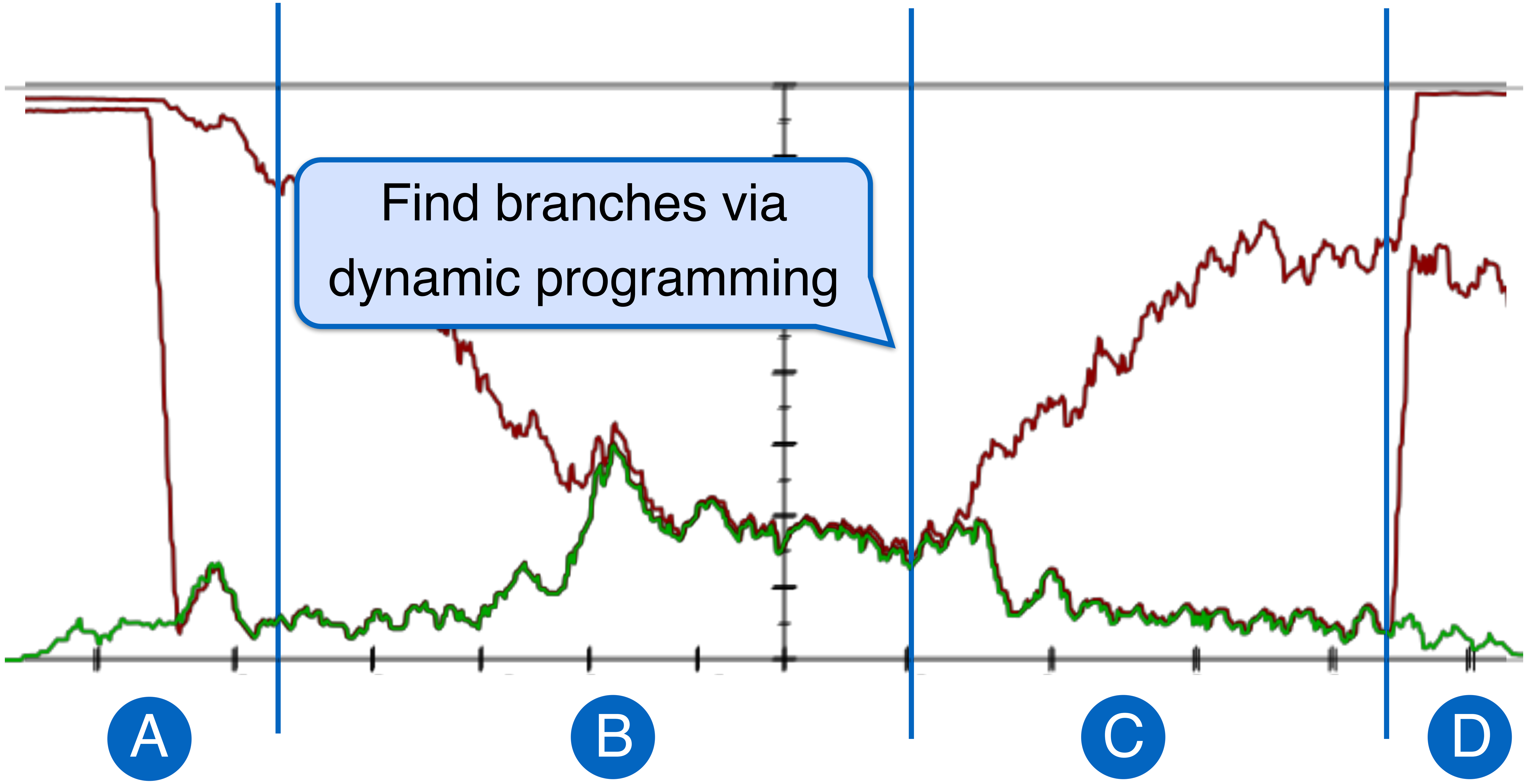
$$\frac{c}{b} - \frac{b}{a} \qquad \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \frac{2c}{-b - \sqrt{b^2 - 4ac}} \qquad -\frac{c}{b}$$



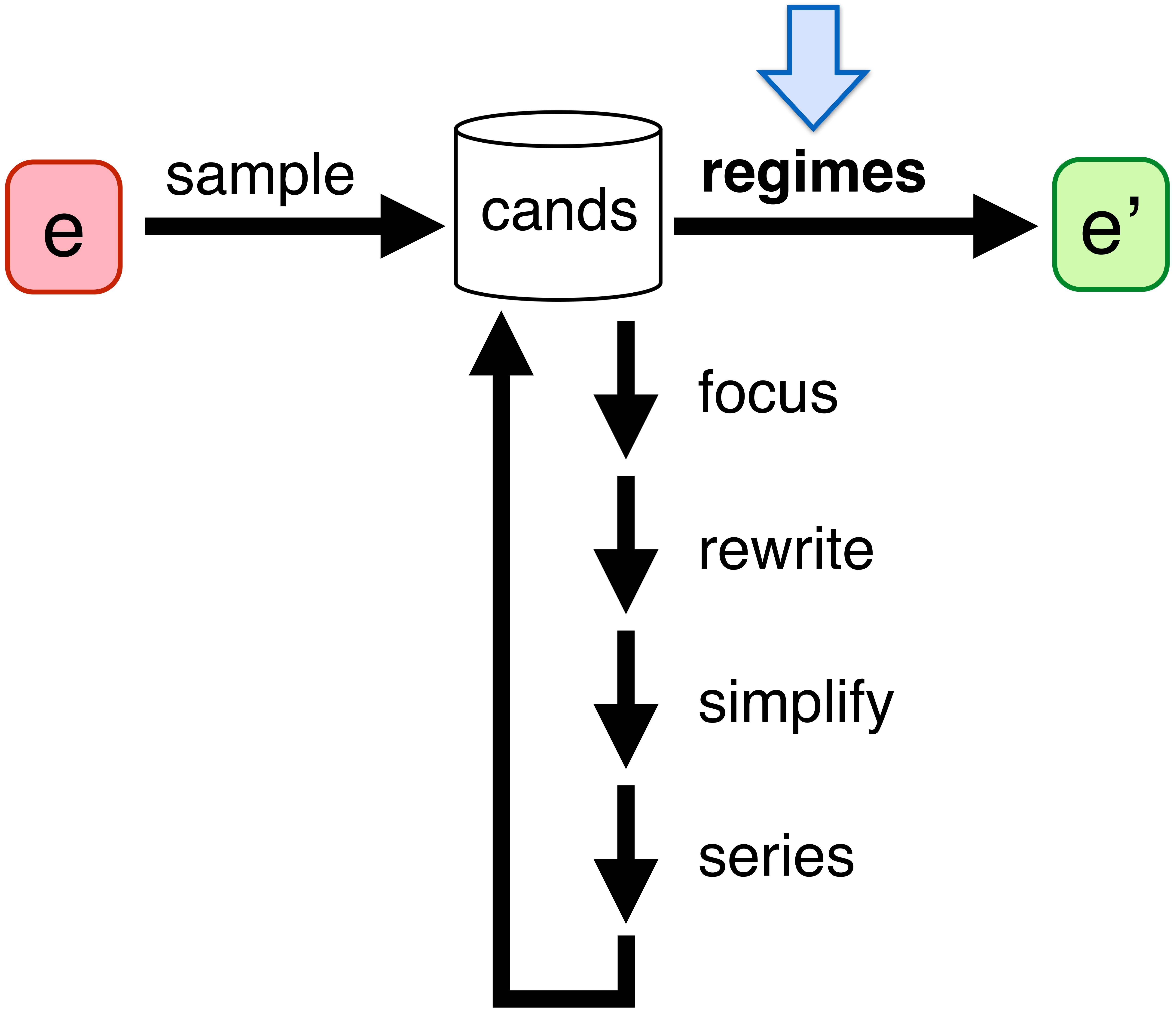
Regime Inference

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

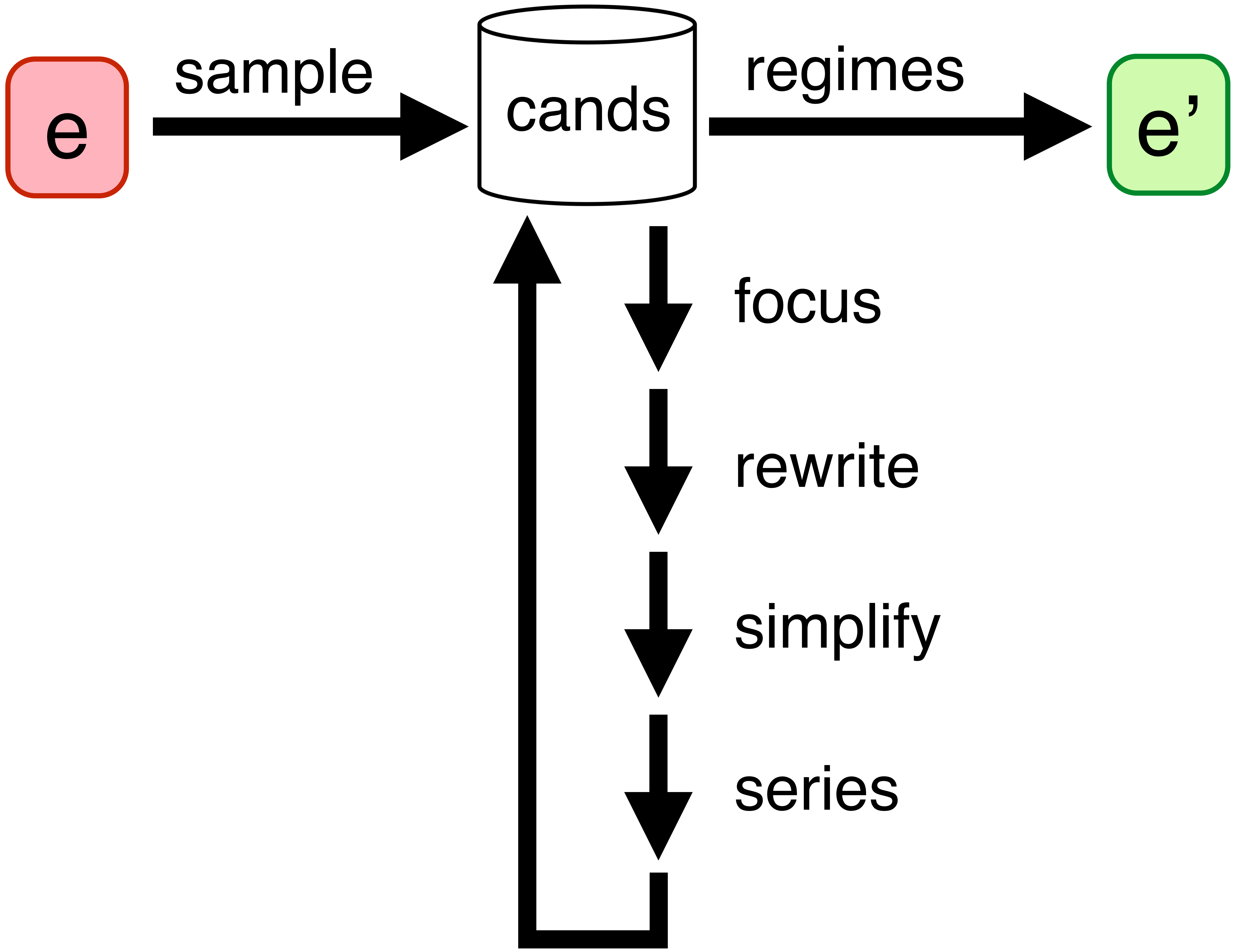
$b < -1.15\text{E}122$ $\quad$ $b < 1.06\text{E}{-304}$ $\quad$ $b < 4.62\text{E}63$ $\quad$ $b > 4.62\text{E}63$



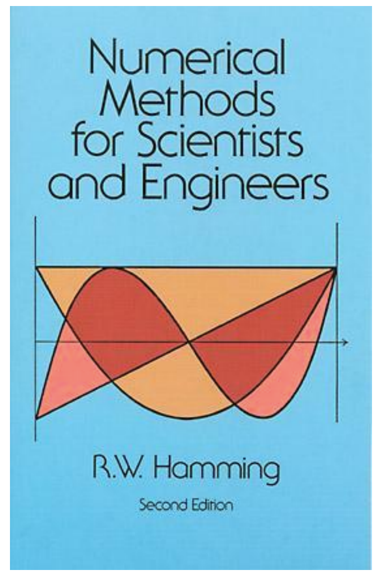
Herbie Architecture

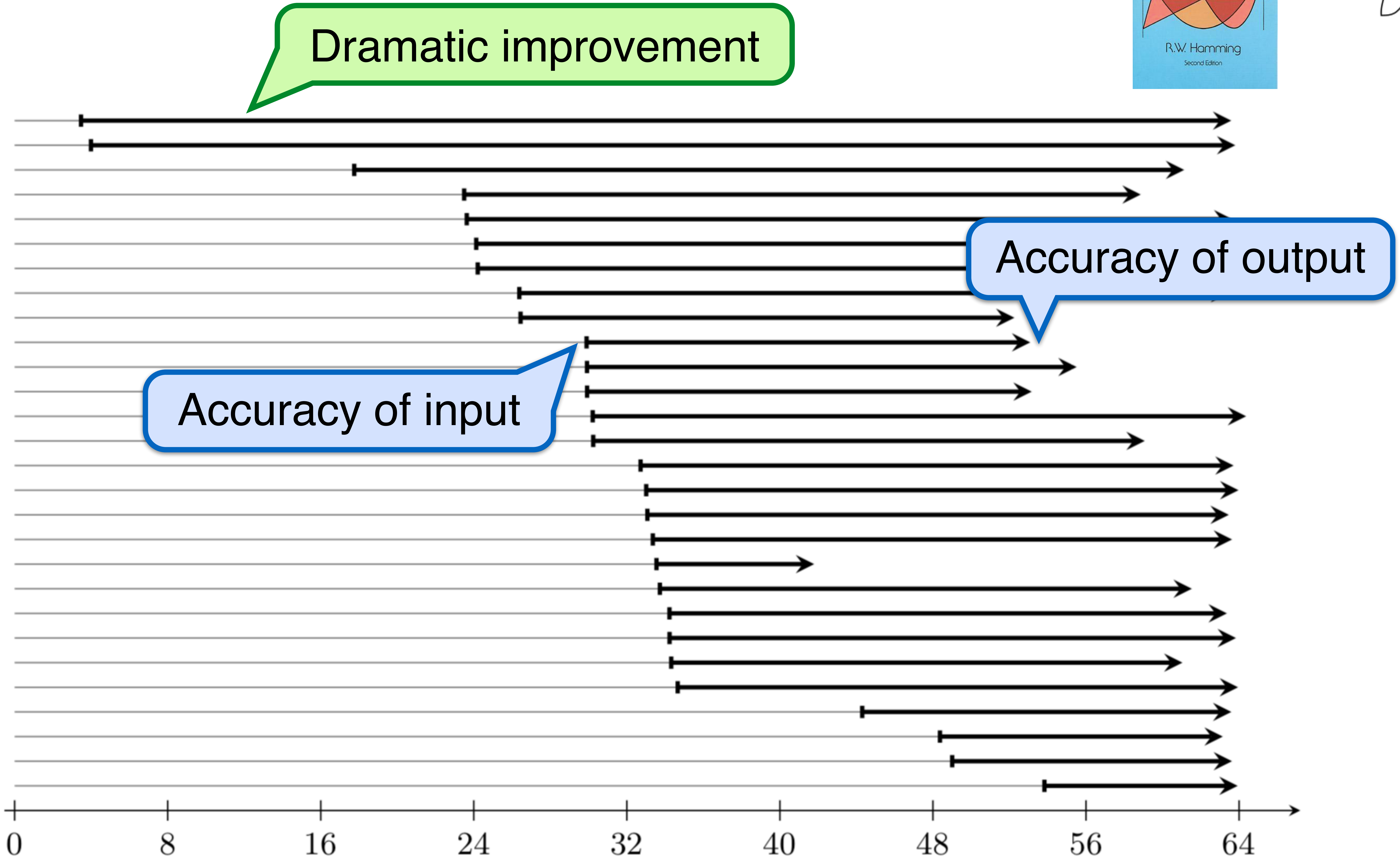


Herbie Architecture



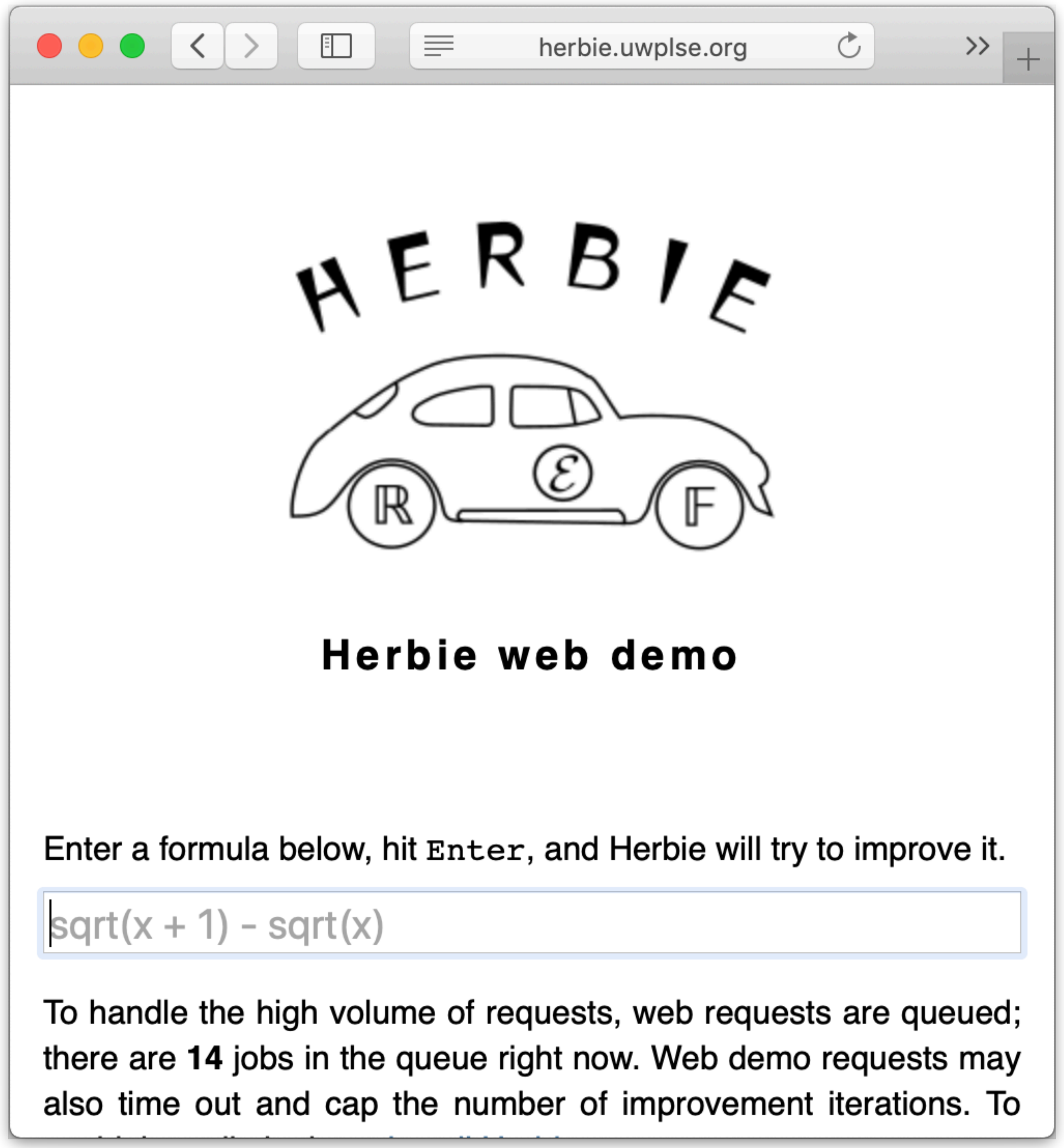
Auto Address Classic Issues





Average bits correct (longer is better)

Herbie Implementation



herbie.uwplse.org

1000s of formulas

"critical"

"fixed my case easily"

Sandia National Laboratories

UNIVERSIDADE DA CORUÑA

"key determinant factor"

NVIDIA

Users at major labs

Cited in papers, theses, etc.

Outline

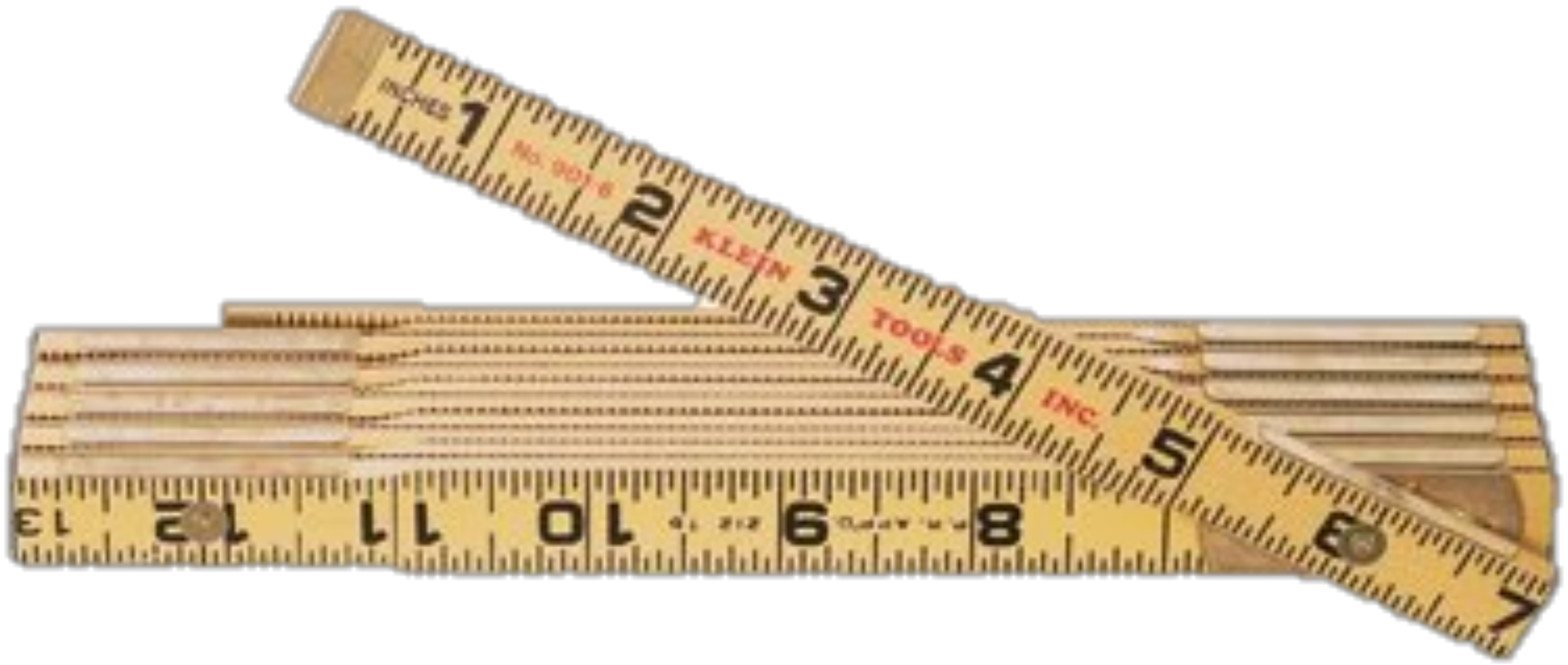
Herbgrind: Finding error in large applications

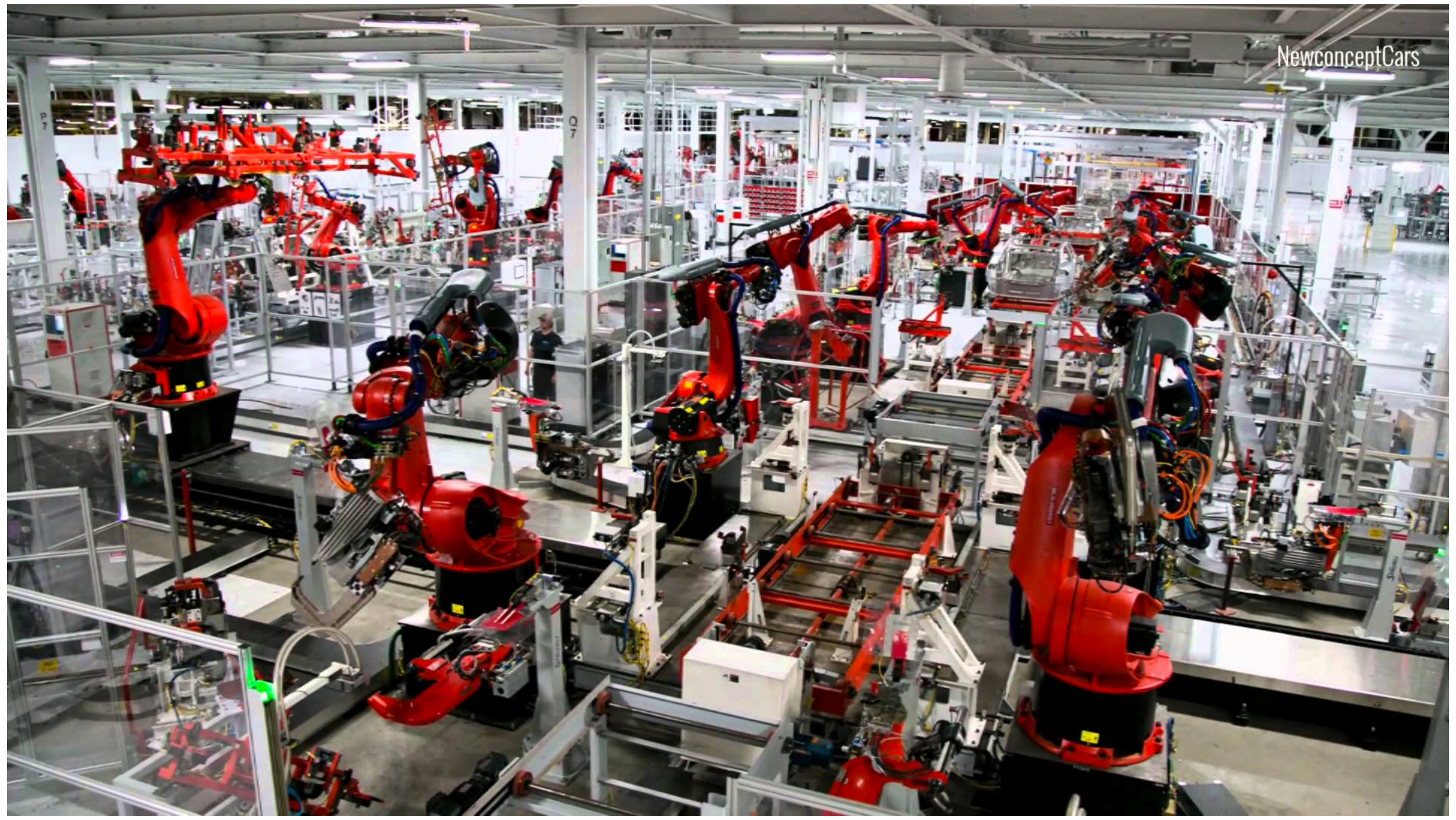
Herbie: Automatically improving accuracy

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

		HOL Formal [NSV’16]	
LSGA [ICSE’15]		FPTaylor [FM’15]	FPTuner [POPL’17]
Ariadne [POPL’13]	Salsa [FMICS’15]	Rosa [POPL’14]	STOKE [PLDI’14]
FPDebug [PLDI’11]	Herbie [PLDI’15]	Fluctuat [SAS’13]	Precimonious [SC’14]
Test	*Debug*	*Verify*	*Optimize*

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

Formats

Vision:

- reproducible, fair comparisons
- lower barrier to entry for new research
- compose existing tools for new problems
- build community (regular competitions?)

alog

. . .

Goal: Std Numeric Tool Yardsticks + Interfaces

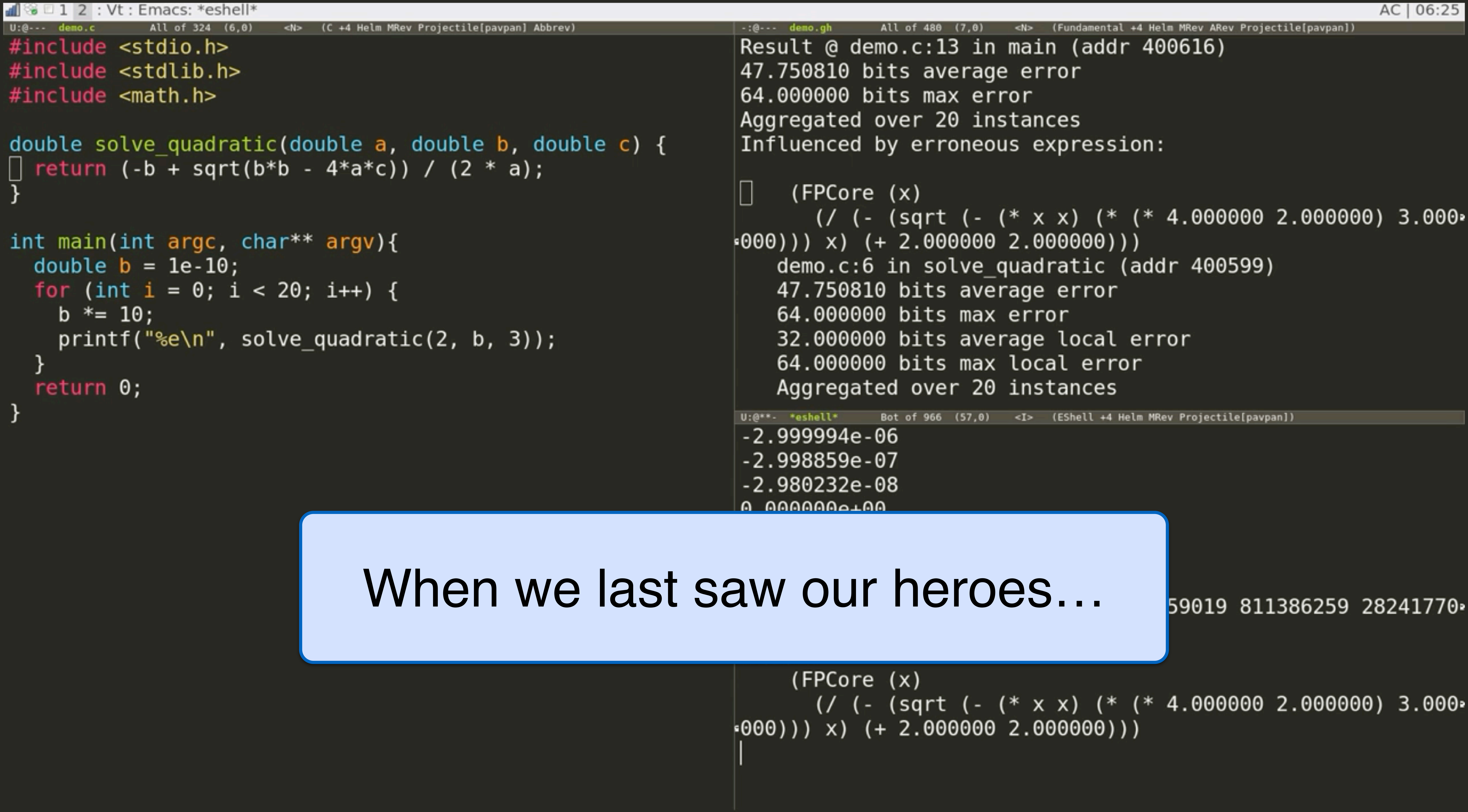
Formats

Vision:

- reproducible, fair comparisons
- lower barrier to entry for new research
- **compose existing tools for new problems**
- build community (regular competitions?)

alog

. . .


1 2 : Vt : Emacs: *eshell*
AC | 06:25
U:@--- demo.c All of 324 (6,0) <N> (C +4 Helm MRev Projectile[pavpan] Abbrev)
#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;
}
-:@--- demo.gh All of 480 (7,0) <N> (Fundamental +4 Helm MRev ARev Projectile[pavpan])
Result @ demo.c:13 in main (addr 400616)
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))) x) (+ 2.000000 2.000000)))
demo.c:6 in solve_quadratic (addr 400599)
47.750810 bits average error
64.000000 bits max error
32.000000 bits average local error
64.000000 bits max local error
Aggregated over 20 instances
U:@**- *eshell* Bot of 966 (57,0) <I> (EShell +4 Helm MRev Projectile[pavpan])
-2.999994e-06
-2.998859e-07
-2.980232e-08
When we last saw our heroes…
59019 811386259 28241770
(FPCore (x)
(/ (- (sqrt (- (* x x) (* (* 4.000000 2.000000) 3.000000))) x) (+ 2.000000 2.000000)))

1 2 : Vt : Emacs: *eshell* AC | 06:25

U:@--- demo.c All of 324 (6,0) <N> (C +4 Helm MRev Projectile[pavpan] Abbrev)

```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;
}
```

-:@--- demo.gh All of 480 (7,0) <N> (Fundamental +4 Helm MRev ARev Projectile[pavpan])

```
Result @ demo.c:13 in main (addr 400616)
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) 3.000
000))) x) (+ 2.000000 2.000000)))
   demo.c:6 in solve_quadratic (addr 400599)
   47.750810 bits average error
   64.000000 bits max error
   32.000000 bits average local error
   64.000000 bits max local error
   Aggregated over 20 instances
```

U:@**- *eshell* Bot of 966 (57,0) <I> (EShell +4 Helm MRev Projectile[pavpan])

```
-2.999994e-06
-2.998859e-07
-2.980232e-08
0.000000e+00
0.000000e+00
==26668==
~/mpi-talk $ herbie shell
Seed: #(1619739517 3155869033 1088559019 811386259 28241770
02 2406577080)
herbie>
    (FPCore (x)
      (/ (- (sqrt (- (* x x) (* (* 4.000000 2.000000) 3.000
000))) x) (+ 2.000000 2.000000)))
```

External Collab: Daisy + Herbie [FM 18]

+

MPI-SWS

UW

https://github.com/malyzajko/daisy

https://herbie.uwplse.org/

✓ Faster Daisy

✓ Validated Herbie

✓ Compare rewrite algorithms

1 2 3 : Vt : Emacs: ~/dagstuhl-talk/input.fpcore AC: 96%+ (7:50) | 12:07

U:@**- *eshell* All of 46 (3,18) <I> (EShell +4 Helm MRev Projectile[pavpan])

```
Welcome to the Emacs shell

~/dagstuhl-talk $
```

U:@--- input.fpcore All of 80 (2,0) <V> (Scheme Racket/A +4 Helm company MRev Projectile[pavpan] Abbrev)

```
(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

s-exprs

meta

spec

pure expr

```
(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 9.80
    (while (< n
      ([t t0 (l
               (let ([k2t       w (* (/ h 2) k1w))])
                 (+ t (* h       t))))]
       [w w0 (let ([k2w (* (     (- g) L)
                           (sin (+ t (* (/ h 2) w))))])
               (+ w (* h k2w)))]
       [n 0 (+ n 1)])
      t)))
```

common C/Fortran ops

loops

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
 (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 1000)
  ((delta_dl 0.0 (let ((c 12.34)) (* c sl)))
   (delta_dr 0.0 (let ((c 12.34)) (* c sr)))
   (delta_d
    0.0
    (let ((delta_dr (let ((c 12.34)) (* c sr)))
          (delta_dl (let ((c 12.34)) (* c sl))))
      (* (+ delta_dl delta_dr) 0.5)))
   (delta_theta
    0.0
    (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)))
   (arg
    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))))
      (+ theta (* delta_theta 0.5))))
   (cosi
    0.0
    (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)))))
      (cos arg)))
(x
 0.0
 (let ((cosi
        (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)))))
          (cos arg)))
       (delta_d
        (let ((delta_dr (let ((c 12.34)) (* c sr)))
              (delta_dl (let ((c 12.34)) (* c sl))))
          (* (+ delta_dl delta_dr) 0.5))))
   (+ x (* delta_d cosi))))
(sini
 0.0
 (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 arg)))
(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 arg)))
       (delta_d
        (let ((delta_dr (let ((c 12.34)) (* c sr)))
              (delta_dl (let ((c 12.34)) (* c sl))))
          (* (+ delta_dl delta_dr) 0.5))))
   (+ 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)))
(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 1000
  ((delta_
   (delta_
   (delta_
    0.0
    (let (

      (*
   (delta_
    0.0
    (let (


      (* (-
   (arg
    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))))
      (+ theta (* delta_theta 0.5))))
   (cosi
    0.0
    (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)))))
      (cos arg)))
```

```
(x
 0.0
 (let ((cosi
        (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)))))
          (cos arg)))
       (delta_d
        (let ((delta_dr (let ((c 12.34)) (* c sr)))
              (delta_dl (let ((c 12.34)) (* c sl))))
          (* (+ delta_dl delta_dr) 0.5))))
   (+ x (* delta_d cosi))))
(sini
 0.0

                            (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 arg)))
       (delta_d
        (let ((delta_dr (let ((c 12.34)) (* c sr)))
              (delta_dl (let ((c 12.34)) (* c sl))))
          (* (+ delta_dl delta_dr) 0.5))))
   (+ 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)))
(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)))
```

Things can get a little… verbose.

FPBench Formats: FPImp

```
(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) 0.1666666666))
              (* (* arg arg arg arg arg) 0.008333333))]
   [= 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))
```

imperative sugar

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

Benchmark sources		Features used		Domains	
Rosa	37	Arithmetic	111	Textbooks	28
Herbie	28	Temporaries	57	Mathematics	24
Salsa	25	Comparison	33	Controls	10
FPTaylor	21	Loops	28	Science	10
		Exponents	16	(unknown)	39
		Trigonometry	15		
		Conditionals	10		

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	Documentation	Standards
Benchmarks	Tools	FPCore 1.1
Community	FPImp	Metadata 1.1
Download	Implementor Notes	Measures 1.1
Mailing list	Contributors	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.

Not Secure | herbie.uwplse.org/demo/539e839f7c94a19a6c9143346e90671533071b17.e26cdb8327c62adb02cbaae52bdd24dae3fab69e/graph.html

Average Error:	Time:	Precision:	Internal Precision:
$38.2 \to 23.7$	50.3s	64	3392

$$\frac{1}{2} \cdot \sqrt{2 \cdot \left(\sqrt{re \cdot re + im \cdot im} + re\right)}$$

$$\downarrow$$

$$\textbf{if}\ \ (im + re) \cdot 2 \le -2.0781343418246825 \cdot 10^{+152}:$$

$$\frac{\sqrt{2 \cdot (im \cdot im)}}{\sqrt{(-re) - re}} \cdot \frac{1}{2}$$

$$\textbf{if}\ \ (im + re) \cdot 2 \le 1.0269599942889963 \cdot 10^{-298}:$$

$$\frac{1}{2} \cdot \frac{\sqrt{2} \cdot |im|}{\sqrt{\sqrt{re \cdot re + im \cdot im} - re}}$$

$$\textbf{if}\ \ (im + re) \cdot 2 \le 2.0255230297756466 \cdot 10^{+142}:$$

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

$$\textbf{if}\ \ (im + re) \cdot 2 \le 1.853478866708164 3 \cdot 10^{+278}:$$

FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

http://fpbench.org

FPBench

Forma

Tools: ructure

Eval: g doption

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)))))
...)
```

custom formats

(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	Documentation	Standards
Benchmarks	Tools	FPCore 1.1
Community	FPImp	Metadata 1.1
Download	Implementor Notes	Measures 1.1
Mailing list	Contributors	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:

$$\left(\frac{\frac{27}{10} - e}{\pi - (\sqrt{2} + \sqrt{3})}\right)^{67/16}$$

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

(link)

Not Secure | titanic.uwplse.org/evaluate?core=%28FPCore+%28x%29%0A++x%29

```
(FPCore ()
:target 302.88271965546954925O146
:name "Accuracy on a 32-bit budget"
:precision (float 8 24)
(pow
    (/
        (- (/ 27 10) E)
        (- PI (+ (sqrt 2) (sqrt 3))))
    (/ 67 16)))
```

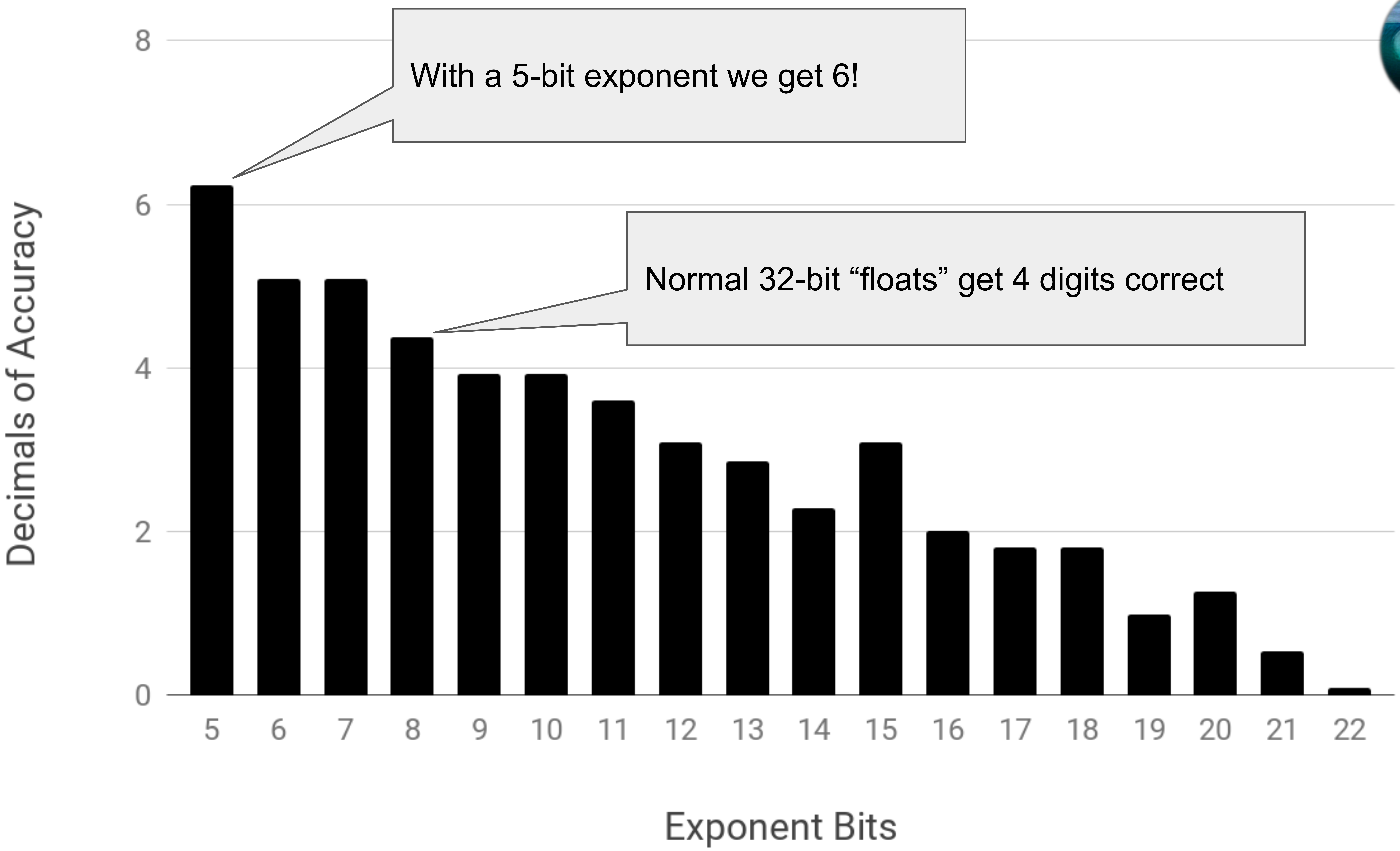
Titanic Evaluator [browse benchmarks]

Evaluate FPCore with IEEE 754 floating-point

w: 11 p: 53 binary64 override

FPCore arguments:

Evaluate FPCore [permalink]


With a 5-bit exponent we get 6!
Normal 32-bit “floats” get 4 digits correct
Decimals of Accuracy
8
6
4
2
0
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
Exponent Bits

Can we do better with mixed precision?

Format	A	B	C	D	E	acc
Posits (es=1)	1	1	1	1	1	7.05
binary32	8	8	8	8	8	4.37
IEEE (w=5)	5	5	5	5	5	6.24
IEEE mixed	5	8	3	2	4	**7.40**

```
(FPCore ()
:name "Accuracy on a 32-bit budget"
:spec 302.882719655469549250146
(pow                                      ; A
  (/                                      ; B
    (- (/ 27 10) E)                       ; 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)
     (! :precision (float 2 30) (+ (! :precision (float 2 30) (sqrt 2)) (! :precision (float 2 30) (sqrt
3))))))))
 (! :precision (float 4 28) (/ (! :precision binary32 67) (! :precision binary32 16)))))
)
```

(link)

How Titanic Works

Arbitrary floating-point is “easy” to implement

- if you aren’t too concerned about speed…

- just do it literally: compute in Reals and Round!

$$[\![x \star y]\!]_{\mathbb{F}} = \mathrm{Round}([\![x \star y]\!]_{\mathbb{R}})$$

- 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 Internals

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

Bill added Posit Sinking Point over coffee yesterday :)

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*

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

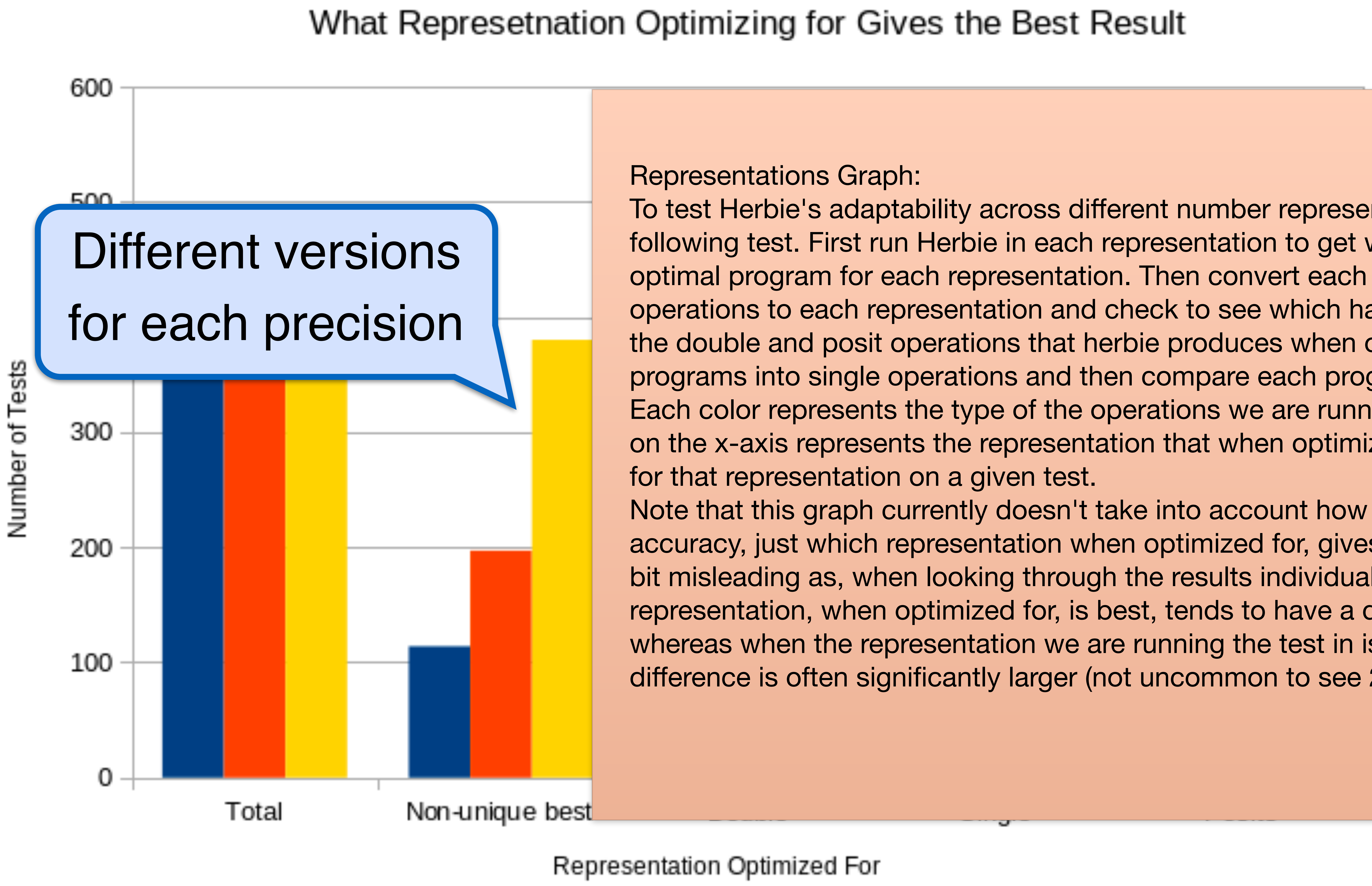
- Each mode produces best results for that precision
- Most expressions need different optimized expressions

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

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

vectors, fixed point, double double

Scriptable reproducibility

$ git clone && make report

Compose more cross-group tools

Precimonius/Salsa, FPTaylor/Stoke, ???

Establish challenge problems!

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/

JUMP

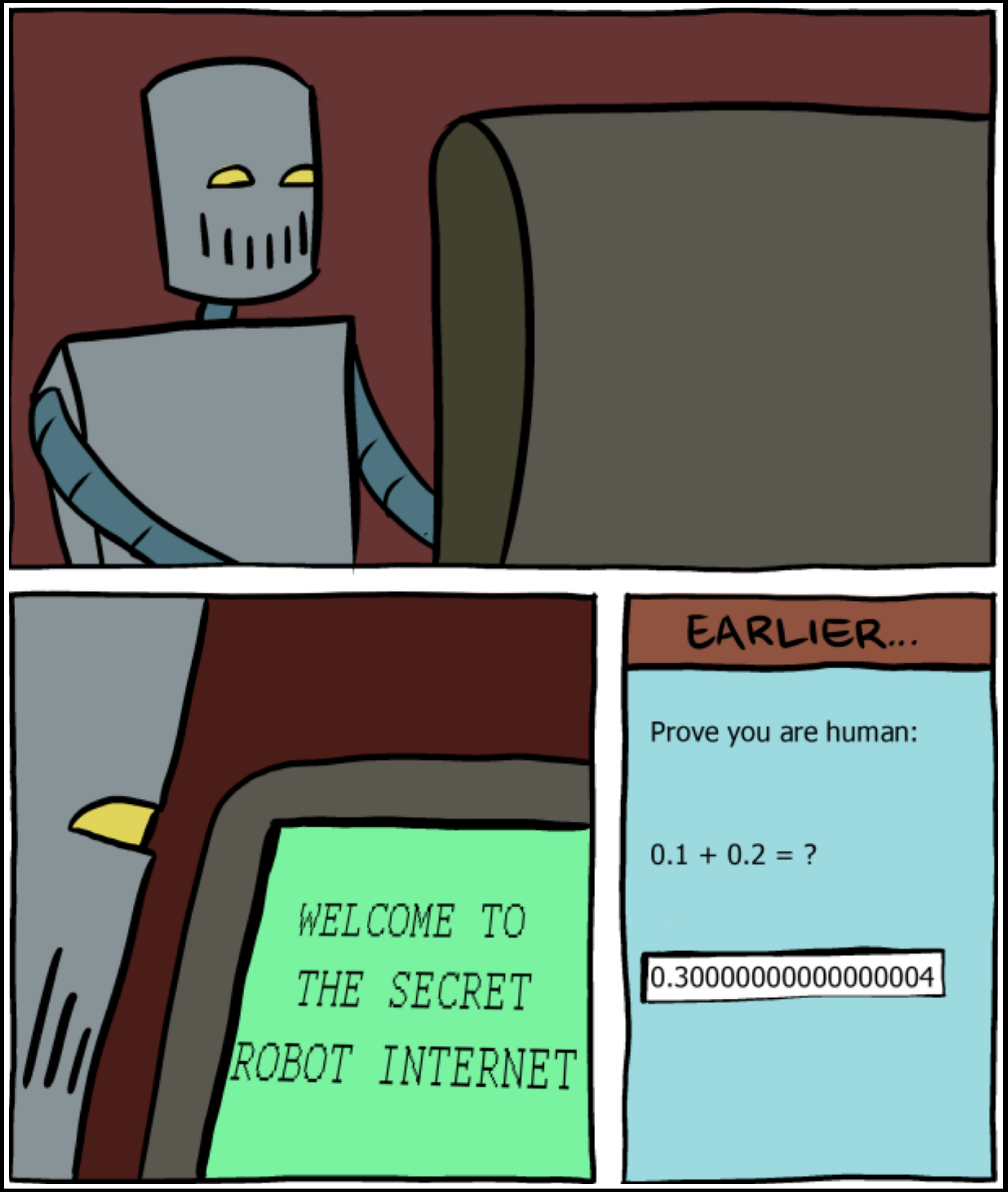
Joint University Microelectronics Program

www.src.org/program/jump

Semiconductor Research Corporation

@srcJUMP

Applications Driving Architectures


WELCOME TO THE SECRET ROBOT INTERNET
EARLIER...
Prove you are human:
0.1 + 0.2 = ?
0.30000000000000004