

Herbie

or: what to do if you can't stop worrying and hate floating
point

Sharif Olorin <sio@tesseract.org>

Ambiata

July 10, 2016

Arithmetic quiz

```
sum :: [Double] -> Double
```

Arithmetic quiz

```
sum :: [Double] -> Double
sum = foldl (+) 0.0
```

Arithmetic quiz

```
sum :: [Double] -> Double
sum = foldl (+) 0.0 . sort
```

Arithmetic quiz

```
sum :: [Double] -> Double
sum [] = 0.0
sum xs =
  uncurry go $ bisect xs
  where
    go [y] [] = y
    go [] [z] = z
    go [y] [z] = y + z
    go ys zs =
      let (y1s, y2s) = bisect ys
          (z1s, z2s) = bisect zs in
          (go y1s y2s) + (go z1s z2s)

    bisect ws =
      let len = length ws `div` 2 in
          (take len ws, drop len ws)
```

Arithmetic quiz

```
sum :: [Double] -> Double
sum = fst . foldl add (0.0, 0.0)
  where
    add (acc, err) x =
      let
        -- Correct for the error from the last iteration.
        y = x - err
        acc' = acc + y
        -- Algebraically, err' should be zero.
        err' = (acc' - acc) - y
      in (acc', err')
```

A cautionary tale

```
stddev :: Double -> Double
stddev variance = sqrt $ abs variance
```

A cautionary tale

```
stddev :: Double -> Double  
stddev variance = sqrt $ abs variance
```

$$\sigma_{1:n}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu_{1:n})^2$$

Refresher on IEEE 754

$$F = \{s \times b^e \mid s, e \in \mathbb{Z}; b \in \mathbb{N}\}$$

- ▶ In general, arithmetic operations are commutative but not associative.
- ▶ Common causes of error are subtracting very similar values and adding very different values.
- ▶ Multiplication, squaring et cetera can compound existing error.
- ▶ Rounding contributes at most 0.5 ULPs (units in the last place) of error per operation.

What is error?

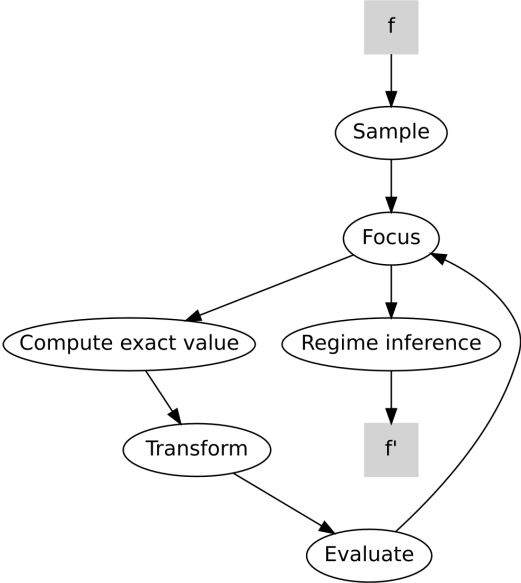
$$\epsilon(x, y) = \log_2 |z \in FP | \min(x, y) \leq z \leq \max(x, y)|$$

- ▶ Error in ULPs: number of floating-point values between the exact result and the approximate result[3].
- ▶ Consistent representation of error independent of magnitude.
- ▶ The binary log approximates “number of incorrect bits”.

Herbie

- ▶ Provides automated synthesis of more accurate versions of floating-point computations[2].
- ▶ Written by Pavel Panchekha et. al. at the University of Washington.
- ▶ Around 10KLOC of Racket.
- ▶ Intended for scientists, statisticians, people who don't necessarily have a background in numerical analysis.

Herbie



Sample

- ▶ Sample inputs are drawn uniformly from the computation's domain.
- ▶ Herbie defaults to 256 samples per iteration.
- ▶ More samples leads to greater probability of identifying regions of the domain with differing error behaviour.

Error and input domain

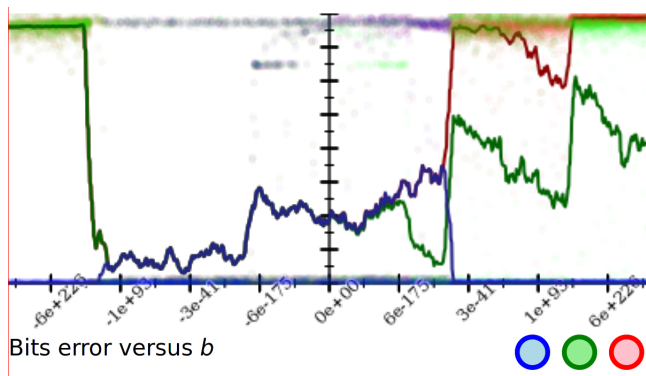


Figure: Herbie's error estimates for the quadratic formula $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$

Focus

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

- ▶ For each operator \star , evaluate operands a and b in exact arithmetic for all of the sampled inputs.
- ▶ Evaluate $a \star b$ in both exact arithmetic and floating-point arithmetic.
- ▶ Focus search on operators which contribute the most error.

“Exact” results

Even with arbitrary precision, how do you know how many bits are “enough”? Herbie guesses:

- ▶ Compute the result with n bits of precision over your entire sample.
- ▶ Do it again with $2n$ bits.
- ▶ If the most significant 64 bits of the results match, this is your exact answer; otherwise continue.

Transform

- ▶ Hill-climbing greedy search of a database of rewrite rules.
 - ▶ $x^2 - y^2 \rightsquigarrow (x - y)(x + y)$
- ▶ Transformations are either mathematical identities or near-identities - sometimes using an approximation can result in a numerical result closer to the true value.
- ▶ Followed by a series-expansion pass.
 - ▶ $e^x - 1 \rightsquigarrow x + \frac{1}{2}x^2 + \frac{1}{6}x^3$ for $x \approx 0$
- ▶ Simplification phase to cancel like terms, et cetera - pattern-match expressions which can be reduced.
 - ▶ $\frac{y}{e^x - x} \rightsquigarrow y$

Regime inference

- ▶ Many computations have error characteristics which vary based on the magnitude of the input within the domain (“regime”).
- ▶ This necessitates the selection of different implementations at runtime based on the value of the inputs, as no single formula will be accurate in all cases.
- ▶ Herbie localises regime boundaries using the segmented least squares dynamic programming algorithm[1].
- ▶ To avoid overfitting, a penalty is added when evaluating each segmentation - one bit of error per branch.

A simple example

$$\sqrt{x} - \sqrt{x-1}$$

```
(herbie-test (x)
  "Subtracting square roots"
  (- (sqrt x)
     (sqrt (- x 1))))
```

Combining variance of subsamples

$$\sigma_{1:n+m}^2 = \frac{m(\sigma_{1:m}^2 + \mu_{1:m}^2) + n(\sigma_{m+1:n}^2 + \mu_{m+1:n}^2)}{m+n} - \mu_{1:n+m}^2$$

- ▶ $\sigma_{a:b}^2$ is the variance of the subsample from values a to b .
- ▶ $\mu_{a:b}^2$ is the mean of the subsample from values a to b .

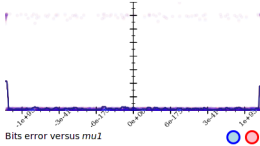
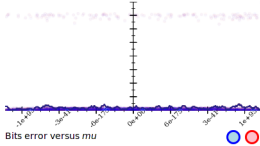
Combining variance of subsamples

```
(herbie-test (mu
              mu1
              [var1 (uniform 0 1000000000)]
              [n (< 0 int)]
              mu2
              [var2 (uniform 0 1000000000)]
              [m (< 0 int)])
  "Combine variance of subsamples"
  (- (/ (+ (* n (+ var1 (sqr mu1)))
           (* m (+ var2 (sqr mu2)))))
      (+ n m))
     (sqr mu)))
```

Combining variance of subsamples

$$\frac{n \cdot (\text{var1} + \text{mu1}^2) + m \cdot (\text{var2} + \text{mu2}^2)}{n + m} - \text{mu}^2$$

Test: Combine variance of subsamples
Bits: 128 bits



Time: 8.8 m Input Error: 1.3 Output Error: 1.4 🔍

$$(n \cdot (\text{var1} + \text{mu1}^2) + m \cdot (\text{var2} + \text{mu2}^2)) \cdot \frac{1}{n+m} - \text{mu}^2$$



Started with

$$\frac{n \cdot (\text{var1} + \text{mu1}^2) + m \cdot (\text{var2} + \text{mu2}^2)}{n + m} - \text{mu}^2$$

Using strategy n

1.3 bits

Applied div-inv to get

1.4 bits

$$\frac{n \cdot (\text{var1} + \text{mu1}^2) + m \cdot (\text{var2} + \text{mu2}^2)}{n + m} - \text{mu}^2 \rightsquigarrow \frac{(n \cdot (\text{var1} + \text{mu1}^2) + m \cdot (\text{var2} + \text{mu2}^2)) \cdot \frac{1}{n+m} - \text{mu}^2}{1}$$

Removed slow pow expressions

Figure: Herbie report for combining subsample variance.

Miscellanea

- ▶ Herbie: <https://github.com/uwplse/herbie>
- ▶ GHC plugin:
<https://github.com/mikeizbicki/HerbiePlugin>
- ▶ Rust plugin:
<https://github.com/mcarton/rust-herbie-lint>
- ▶ Valgrind plugin: <https://github.com/uwplse/herbgrind>
- ▶ These slides: <https://tesseract.org/doc/slides/2016-05-25-fp-syd-herbie.pdf>

Bibliography



Jon Kleinberg and Éva Tardos.

Algorithm design.

Pearson Education India, 2006.



Pavel Panchekha, Alex Sanchez-Stern, James R Wilcox, and Zachary Tatlock.

Automatically improving accuracy for floating point expressions.

In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1–11. ACM, 2015.



Eric Schkufza, Rahul Sharma, and Alex Aiken.

Stochastic optimization of floating-point programs with tunable precision.

ACM SIGPLAN Notices, 49(6):53–64, 2014.