Toward Standard Formats and Benchmark Suites for Floating Point Tools

Zach Tatlock
Collaborators

Nasrine Damouche
Matthieu Martel
Eva Darulova
Heiko Becker
Debasmita Lohar

Pavel Panchekha
Alex Sanchez-Stern
Chen Qiu
Sorin Lerner
Bill Zorn
Dan Grossman
Yardsticks and Assembly Lines
Diverse Yardsticks Across Numerics

Accuracy

*absolute, relative, ulp, bound, average*

Performance

*space, runtime, analysis time*

Expressiveness (domain)

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

Accuracy

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

Sigmund Freud

Civilization and Its Discontents
Diverse Yardsticks Across Numerics

Accuracy

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

Sigmoid F-round

Simulation and Its Discontinuities
# Diverse Tools Across Numerics

<table>
<thead>
<tr>
<th>Test</th>
<th>Debug</th>
<th>Verify</th>
<th>Optimize</th>
</tr>
</thead>
</table>

### Accuracy / Performance / Domain
Diverse Tools Across Numerics

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

Compilation (SPEC INT, EEMBC)

\textit{compile time, run time, code size}

SAT/SMT (DIMACS, SMT-LIB)

\textit{solver time, model size, theory support}

Synthesis (SyGuS)

\textit{invariant synth, programming by example, etc.}

\ldots
Measures & Fmts: Community

SyGuS competitions:
- challenge problems
- measure progress
- provide infra
- support research

Rishabh Singh
SyGuS Organizer
Measures & Fmts: Community

SyGuS competitions:
- challenge problems
- measure progress
- provide infra
- support research

Rishabh Singh
SyGuS Organizer

FUN!
Goal: Standard Numeric Yardsticks

Formats

*core, imperative, precisions, std error defs*

Tools

*reference / baseline eval, infrastructure*

Benchmark Suites

*diverse domains, objectives, challenge catalog*

...
**Goal:** Standard Numeric Yardsticks

**Formats**

**Vision:**

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

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

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

(FPCore (t0 w0 N)
  :name "Pendulum"
  :fpbench-domain science
  :pre (and (< -2 t0 2) (< -5 w0 5))
  :example ([N 1000])
  (let (/[h 0.01]
  [L 2.0]
  [m 1.5]
  [g 9.80665]
  (while (< n N)
  ([t t0 (let ([k1w (* (/ (- g) L) (sin t))]
            (let ([k2t (+ w (* (/ h 2) k1w))]
                (+ t (* h k2t))))]
  [w w0 (let ([k2w (* ((- g) L)
                (sin (+ t (* (/ h 2) w)))))
                (+ w (* h k2w))))]
  [n 0 (+ n 1)]
  t))))

loops

common C/Fortran ops
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 (damoucho-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_di 0.0 (let ((c 12.34)) (* c sl)))
     (delta_dr (let ((c 12.34)) (* c sr)))
     (delta_d 0.0)
     (let ((delta_dr (let ((c 12.34)) (* c sr)))
           (delta_di (let ((c 12.34)) (* c sl)))
           (+ (delta_di delta_dr) 0.5))
     (delta_theta 0.0)
     (let ((inv_l 0.1))
       (delta_di (let ((c 12.34)) (* c sl)))
       (delta_dr (let ((c 12.34)) (* c sr)))
       (+ (delta_dr delta_di) inv_l))
     (arg 0.0)
     (let ((delta_theta
            (let ((inv_l 0.1))
              (delta_di (let ((c 12.34)) (* c sl)))
              (delta_dr (let ((c 12.34)) (* c sr)))
              (+ (delta_dr delta_di) inv_l)))
              (+ theta (* delta_theta 0.5)))
       (cosi 0.0)
       (let ((arg
              (let ((inv_l 0.1))
                (delta_di (let ((c 12.34)) (* c sl)))
                (delta_dr (let ((c 12.34)) (* c sr)))
                (+ (delta_dr delta_di) inv_l)))
                (+ theta (* delta_theta 0.5)))
               (cos arg)))
     (delta_d
      (let ((delta_dr (let ((c 12.34)) (* c sr)))
            (delta_di (let ((c 12.34)) (* c sl)))
            (+ (* delta_di delta_dr) 0.5))
            (x (* delta_d cosi)))
     (sini 0.0)
     (let ((arg
            (let ((inv_l 0.1))
              (delta_di (let ((c 12.34)) (* c sl)))
              (delta_dr (let ((c 12.34)) (* c sr)))
              (+ (* delta_di delta_dr) 0.5)))
              (sin arg)))
     (p 0.0)
     (let ((sini
            (let ((inv_l 0.1))
              (delta_di (let ((c 12.34)) (* c sl)))
              (delta_dr (let ((c 12.34)) (* c sr)))
              (+ (* delta_di delta_dr) 0.5)))
              (+ y (* delta_d sini)))
     (theta
t (-0.985)
     (let ((delta_theta
            (let ((inv_l 0.1))
              (delta_di (let ((c 12.34)) (* c sl)))
              (delta_dr (let ((c 12.34)) (* c sr)))
              (+ (* delta_di delta_dr) inv_l)))
              (+ theta delta_theta)))
   (t 0 (+ t 1))
   (j 0 (if (== j 50) 0 (+ j 1)))
   (tmp 0.0 (if (== j 50) tmp))
   (sl* (if (== j 50) sl tmp))
   (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))
:final
(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 arg) 0.1666666666))
              (* (* arg arg arg arg arg) 0.0083333333))] =
    [= y (+ y (* delta_d sini))] =
    [= theta (+ theta delta_theta)] =
    [= t (+ t 1)]
  (if
   [(== j 50)
    [= j 0] [= tmp sl]
    [= sl sr] [= sr tmp]]
   [else
    [= j (+ j 1)])))))
  (output x y))
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
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 and Scala*
FPBench Tools: Input and Eval

Problem: bit-accurate specifications of inputs

long running source of confusion and bugs

WIP: *Titanic*, tools guaranteed to float correctly

correct infinite precision floating point eval

Helping nail down FPBench input details

provides “atlas” of floating point numbers

16758583.533951732330024242401123046875
123238695774976600420344224131100880928768
5.0793320042430423200130462646484375e-5
37703209896995967665977688064
FPBench Tools: Input

164944456120542988788857969579798898
848745580129785881719514679835851998
2827193678093476118971878733307512917
512214621913774369446099576353474057
244632351873214268185139305478293387
386686747852583229728529651430805712
753941691966333233175374150145035828
048915595264

http://titanic.uwplse.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
Approx 100 benches from pubs

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

From ~ 6 papers in FM, PLDI, POPL, FMICS, etc.

bound verifies, sound and heuristic improvers

http://fpbench.org/benchmarks.html
Demo: Composing Daisy / Herbie
Welcome to the Emacs shell

~/dagstuhl-talk $ |

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

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
FPBench

Formats: FPCore, FPImp

Tools: reference evaluators, infrastructure

Eval: growing suite, anecdotes, adoption

Future: more types, more benches, RFCs

http://fpbench.org
In the Works

Separate details for instantiating benchmarks provide type assignments, error measures, etc.

(FPCore (x)
  :name "NMSE example 3.9"
  :cite (hamming-1987)
  :fpbench-domain textbook
  :pre (≠ x 0@1)
  (- (/ 1@2 x)
    (/ 1@3 (tan x))))

(Eval “NMSE example 3.9”
  :name “NMSE 3.9 — double”
  :rounding away
  :type 'default binary64)

(Eval “NMSE example 3.9”
  :name “NMSE 3.9 — mixed”
  :rounding even
  :type x fixed32.8
  :type 1 binary32
  :type 2 binary64
  :type 3 double2)

(Eval “NMSE example 3.9”
  :name “NMSE 3.9 — single”
  :rounding nearest
  :type 'default binary32)
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!

FPBench: Formats, Suites, Tools

http://fpbench.org