Floating-point Robustness Estimation by Concrete Testing

Student: Wei-Fan Chiang
Advisors: Prof. Ganesh Gopalakrishnan and Prof. Zvonimir Rakamarić
School of Computing, University of Utah, USA

Our Recent Work

- “Sour spots” detection: [PPoPP’14]
  - Finding inputs causing high round-off errors

Our Focus: Robustness Estimation

- Estimate robustness via divergence detection
  - Divergence: given an input to both an imprecise and a precise implementation of the same computation, check if computed results agree.
  - See comparison det > 0.0 below as an example.

- Empirically estimate the probability of triggering divergence to indicate robustness

Proposed Framework

- Requirements on programs: generate a discrete output value/object which is decided by a floating-point value
  - E.g. A matrix determinant (float point) is computed to decide the 2D point orientation (integer).
  - We limit discrete values to integers and booleans.

- Step1: Employ an optimizer to search inputs causing high errors on the floating-point value which decides the discrete output
- Step2: Local Search by expanding the initial input to an input sub-domain
  - See Fig.1(b) as an example.

Our Key Idea

- Count the number of divergence inputs in some selected input sub-domains
  - Exhaustive enumeration is expensive.

- 2D orientation example:
  1) Purple points in Fig.1 denote the inputs causing divergences.
  2) Count the # of purple points we can find.

Proposed Solution: Concrete Testing

- Provides concrete inputs for program analysis and tuning
- Has good scalability
- Incorporates effect of compilation flags or hardware

Concrete testing can benefit to program analysis and tuning

Related Work

- Consistency Analysis [S. Chaudhuri POPL’14]
  - Over-approximation on an abstract model. We deal with real machine code.
  - Scalability is limited by float constraint solver.

- Floating-point round-off estimation
  - Over-approximation [G. Melquiond et al. SAC’06, E. Darulova OOPSLA’11 POPL’14]
  - Under-approximation [W. Chiang PPoPP’14]

- Performance-precision analysis tuning
  - Precision allocation for GPU programs [M. Tauer IPDPS’13]
  - Precision tuning tools: [E. Schultza PLDI’14, C. Rubio-Gonzalez SC’13]

- Continuity and robustness analysis [S. Chaudhuri Commun. ACM’12]

- Adaptive arithmetic [J. Shewchuk DCG’12, G. Melquiond IMACS’04]

Conclusions

- Concrete testing can benefit to program analysis and tuning
- Sour spot detection can estimate round-off error
- Divergence detection could estimate robustness
- Concrete inputs show the “brittleness” of a program

Preliminary Results

Robustness comparison between two implementations of 3D point-orientation primitives

- Decide orientation by calculating 3x3 or 4x4 matrix determinant
- Three strategies for finding the initial input
  - Randomly enumerate inputs for 3 minutes and choose “closest” one (URT)
  - Use BGRT [ppopp’14] to enumerate inputs for 3 minutes and choose the closest
  - Linear expansion as shown in Fig.1(b)
- An input is a vector of four 3D points: 12 coordinates in total
- Enumerate 12 sequence of inputs
- Inputs in the same sequence can be mapped to the initial input except one coordinate.
- The differences between each input and the initial are small.

Results

- Four runs under “Experiments” are four runs with different random seeds.

Future Work

- Investigate other methods of divergence detection
- Apply our robustness estimation framework to larger-scale numerical programs. E.g. convex hull construction
- Avoid enumerating inputs in large sub-domain
- Provide a stochastic argument for our concrete testing based robustness estimation

Supported in part by NSF CCF 1421726
(Design Validation Methods for Reliable and Efficient Floating-point)