Checking Roundoff Errors using Counterexample-Guided Narrowing

Do Thi Bich Ngoc
Japan Advanced Institute of Science and Technology
dongoc@jaist.ac.jp

Mizuhito Ogawa
Japan Advanced Institute of Science and Technology
mizuhito@jaist.ac.jp

ABSTRACT
This paper proposes a counterexample-guided narrowing approach, which mutually refines analyses and testing if (possibly spurious) counterexamples are found. A prototype tool CANAT for checking roundoff errors between floating point and fixed point numbers is reported with preliminary experiments. We explain this methodology by a running example (Example 1), and details have been reported in [7].

1. INTRODUCTION
Machine representations of real numbers are typically floating point numbers. Our targets are digital signal processing (DSP), such asmpeg4 decoders. Their implementations on floating point numbers is reported with preliminary experiments, but also for finding dominant RE factors in inputs. Pre-narrowing example (Example 1), and details have been reported in [7].

2. EXTENDED AFFINE INTERVAL
A Classical interval (CI) is an interval \([l, h]\) with \(l \leq h\). Let \([l_1, h_1] \subseteq [l_2, h_2]\) if \(h_1 \leq l_2\) and \([l_1, h_1] = \max([l_1, h_1])\). For two CIs \(\mathcal{X}, \mathcal{Y}\), and \(o \in \{+, -, \times, \div\}\), \(\mathcal{X} \circ \mathcal{Y}\) is the smallest CI that contain all possible values of \(x \circ y\) for each \(x \in \mathcal{X}, y \in \mathcal{Y}\)

\[\mathcal{X} \circ \mathcal{Y} = \min(\mathcal{X} \circ \mathcal{Y}), \max(\mathcal{X} \circ \mathcal{Y})\]

where \(\varepsilon_i\) is a noise symbol and \(\mathcal{F}_i\) is a CI for each \((i \leq n)\).

Each \(\varepsilon_i\) is interpreted as a value in \([-1, 1]\). In [3], the operations on EAI are designed for under approximation. In [6], we proposed them for over approximation.

DEFINITION 2. EAI arithmetic consists of operations \{+, -, \times, \div\} on pairs of CIs. Let \(\tilde{x} = \mathcal{X}_0 + \sum_{i=1}^{n} \mathcal{X}_ie_i, \tilde{y} = \mathcal{Y}_0 + \sum_{i=1}^{n} \mathcal{Y}_ie_i, \tilde{X} = \sum_{i=1}^{n}(\mathcal{X}_i \times [-1, 1]), \tilde{Y} = \sum_{i=1}^{n}(\mathcal{Y}_i \times [-1, 1])\). Then,

\[\tilde{x} + \tilde{y} = (\mathcal{X}_0 + \mathcal{Y}_0) + \sum_{i=1}^{n}((\mathcal{X}_i + \mathcal{Y}_i))e_i\]
\[\tilde{x} - \tilde{y} = (\mathcal{X}_0 - \mathcal{Y}_0) + \sum_{i=1}^{n}((\mathcal{X}_i - \mathcal{Y}_i))e_i\]
\[\tilde{x} \times \tilde{y} = \tilde{x} \tilde{y} (\frac{\tilde{X}}{\tilde{Y}})\text{ if }\tilde{Y} \neq 0\]
\[\tilde{x} \div \tilde{y} = \tilde{x} \tilde{y} (\frac{\tilde{X}}{\tilde{Y}})\text{ if }\tilde{Y} \neq 0\]

where:

\[B - \left\{\sum_{i=1}^{n}(\mathcal{X}_i \times \mathcal{Y}_i)\varepsilon_i\right\}\text{ if }\mathcal{Y} \leq \mathcal{X}\]
\[\sum_{i=1}^{n}(\mathcal{X}_i \times \mathcal{Y}_i)\varepsilon_i\text{ otherwise}\]

and \(\frac{\tilde{X}}{\tilde{Y}}\) is computed by Chebyshev approximation [7, 8].

We define the conversion between CI and EAI as follows:

- CI to EAI: Given a CI \(\mathcal{X} = [l, h]\), the EAI coercion is \(\tilde{x} = \frac{l+\varepsilon_h}{2} + \frac{h-l}{2}\varepsilon_i\).
- EAI to CI: Given an EAI \(\tilde{x} = \mathcal{X}_0 + \sum_{i=1}^{n} \mathcal{X}_ie_i\), the EAI projection is \(\mathcal{X} = \mathcal{X}_0 \mathcal{Y} \sum_{i=1}^{n}(\mathcal{X}_i \times \mathcal{X}_i)\varepsilon_i\)
Target programs

Our observation on DSP reference algorithms is that their cores mostly consist of loops with bounded iterations, arrays with fixed sizes, and pointer manipulations without side effects. For instance, in mpeg4 decoder, both the size of arrays and the iterations of loops are mostly 8 x 8, and only the outermost loop repeats depending on the resolution (Figure 1). We limit our discussion to a small subclass of C programs, which have only bounded loops, fixed size arrays, and no pointer manipulations. Thus, the target model is acyclic after replacing each array element with a variable and unfolding bounded loops.

![Figure 1: Structure of mpeg4 decoder reference algorithm](image)

**Example 1.** Throughout Sections 3 and 4, we will explain using this example C program with annotations below.

```c
/* CANAT ALL sign 11 4
P2 r range = 3
P2 y range = 10 10
_test global rst 0.26 */
typedef float Real;
Real rst;
Real P2(Real x, Real y){
(1) if (x>0)
(2) {rst=x*x;}
(3) else rst = 3*x;
(4) rst = rst - y;
(5) return rst; }
```

The annotations describe the inputs of the ORE problem:
- the fixed point format (sp, ip, fp) = (1, 11, 4) with the base b = 2,
- the initial ranges x ∈ [-1,3], y ∈ [-10,10], and
- the RE threshold θ = 0.26.

In the first item, sp is the sign bit, ip is the number of bits of the integer part, and fp is the number of bits of the fraction part. Then, the ORE problem consists of questions:
- does the RE of rst lie within [-0.26, 0.26]?
- does an overflow error occur?

**ORE abstraction**

OREs are estimated by a pair of EAI s, which describe the fixed point number range and the roundoff error range. As a convention, we will refer to those pairs by \( \hat{x}_f, \hat{x}_r \) and \( \hat{y}_f, \hat{y}_r \), respectively.

**Definition 3.** For two pairs of EAI s \( (\hat{x}_f, \hat{x}_r), (\hat{y}_f, \hat{y}_r) \), \( \overline{\mathbb{R}}, \overline{\mathbb{R}}, \overline{\mathbb{R}}, \) \( \overline{\mathbb{R}} \) are defined as:

\[
\begin{align*}
(\hat{x}_f, \hat{x}_r) &\oplus (\hat{y}_f, \hat{y}_r) = (\hat{x}_f + \hat{y}_f, \hat{x}_r + \hat{y}_r \pm \delta) \\
(\hat{x}_f, \hat{x}_r) &\ominus (\hat{y}_f, \hat{y}_r) = (\hat{x}_f - \hat{y}_f, \hat{x}_r - \hat{y}_r \pm \delta) \\
(\hat{x}_f, \hat{x}_r) &\otimes (\hat{y}_f, \hat{y}_r) = (\hat{x}_f \times \hat{y}_f, \hat{x}_r \times \hat{y}_r + \hat{x}_f \times \hat{y}_r + \hat{x}_r \times \hat{y}_f \pm \delta) \\
(\hat{x}_f, \hat{x}_r) &\div (\hat{y}_f, \hat{y}_r) = (\hat{x}_f \div \hat{y}_f, (\hat{x}_f + \hat{x}_r) \div (\hat{y}_f + \hat{y}_r)
\end{align*}
\]

where \( \delta = [-b^{-f}/2, b^{-f}/2] \).

**Roundoff error analysis**

**Example 2.** The input ranges of \( x \) and \( y \) (in Example 1) are represented by

\[
\begin{align*}
\hat{x}_r &= [1, 1] + [2, 2]\varepsilon_1 \\
\hat{y}_r &= [0, 0] + [10, 10]\varepsilon_2
\end{align*}
\]

Since \( fp = 4 \), the initial REs of \( x \) and \( y \) lie in \([-2^{-5}, 2^{-5}]\).

At line (1), since the initial range of \( x \) is \([-1,3]\), CANA cannot decide \((x > 0)\). Therefore, it traces both line (2) and line (3), and later merges their results before line (4).

At lines (2) and (3), the REs of rst are computed by \( \overline{\mathbb{R}} \).

\[
\begin{align*}
\tilde{r}_s_{t}^{(2,3)} &= [-0.031250, 0.031250] + [-0.123091, 0.123091]\varepsilon_1 + [0.059615, 0.065385]\varepsilon_3 \\
\tilde{r}_s_{t}^{(3)} &= 3 \times \hat{x}_r = [0.093750, 0.093750]\varepsilon_3
\end{align*}
\]

They are merged as:

\[
\tilde{r}_s_{t}^{(3)} = [-0.031250, 0.031250] + [-0.123091, 0.123091]\varepsilon_1 + [0.059615, 0.093750]\varepsilon_3 + [-0.031250, -0.031250]\varepsilon_4
\]

At line (4), we obtain the RE of rst by \( \overline{\mathbb{R}} \):

\[
\tilde{r}_s_{t}^{(4)} = [-0.031250, 0.031250] + [-0.123091, 0.123091]\varepsilon_1 + (0.059615, 0.093750]\varepsilon_3 + [-0.031250, -0.031250]\varepsilon_4
\]

The RE \( \hat{r} \) of rst (i.e., \( \tilde{r}_s_{t}^{(5)} \)) coincides with \( \tilde{r}_s_{t}^{(4)} \), and is bounded by \([-0.279341, 0.279341] \)

We denote

\[
\hat{r} - \hat{r}_0 + \sum_{i=1}^{4} \hat{r}_i \varepsilon_i
\]

and refer to the coefficient \( \varepsilon_i \) (in \( \tilde{r}_s_{t}^{(4)} \)) by \( \varepsilon_i \).

Note that, during this analysis, over approximations occur at the conditional branch (line (1)) with \( \omega \) and the multiplication (line (2)) with \( \delta = -0.031250, 0.031250 \).

**Observation on RE analysis**

The ORE analysis above is over approximate. Thus, there may be spurious counterexamples.

**Example 3.** In Example 1, let inputs be \( x = 3 \) and \( y = 10 \). Then \( x_f = 3, x_r \approx 2^{-5}, y_f = 10, y_r \approx -2^{-5} \). CANA detects \( \hat{r} = -0.06348, 0.279341 \) \( \subseteq [-0.26, 0.26] \), whereas testing detects REs at most 0.219727 \( \subseteq [-0.26, 0.26] \).

Fortunately, the analysis result shows extra information about the effects of inputs on the RE, since EAI coercions of input ranges and the RE share common noise symbols.

**Example 4.** In the analysis result for Example 2:

- **Irrelevant noise symbol:** Since \( \hat{r}_0 = [0, 0] \), \( \varepsilon_2 \) in \( \hat{r} \) is an irrelevant noise symbol. Hence, \( y_f \) (the fixed point part of \( y \)) does not affect the RE of rst.
- **Sensitivity of noise symbols:** Since \( [\hat{r}_1] = -0.123091 - \max \{|\hat{r}_1|, \ldots, |\hat{r}_4|\} \), \( \varepsilon_1 \) is the most sensitive, and \( x_f \) (the fixed point part of \( x \)) affects \( \hat{r} \) the most.
- **Pre-evaluation:** For \( t = (x_f, y_f, x_r, y_r) = (1, 5, 0, 0) \), the corresponding values of noise symbols \( \varepsilon_2, \varepsilon_3, \varepsilon_4 \) are \( (0, 0.5, 0.0) \). By instantiating them to \( \hat{r} \), the RE for the input \( t \) is bounded as \([-0.031250, 0.031250] \subseteq [-0.26, 0.26] \). Hence, we can exclude \( t \) without testing.
4. COUNTEREXAMPLE-GUIDED NARROWING

The counterexample-guided narrowing mutually refines analyses and testing when counterexamples are found.

Assume that there are $2m$ noise symbols. All combinations of $k_i$ ticks of $[h_i, h_i]$ for $i \leq 2m$, which are the lattice points of the grid over the input domain $D = [l_1, h_1] \times \cdots \times [l_{2m}, h_{2m}]$, compose test cases.

**Definition 4.** For an interval $[l, h]$ and $k \geq 1$,

- the $k$-random ticks are $\{c_1, \cdots, c_k\}$, and
- the $k$-periodic ticks are $\{c, c + \Delta, \cdots, c + (k-1)\Delta\}$,

where $\Delta = \frac{h - l}{k}$, and $c \in [l, l + \Delta]$, $c_1, \cdots, c_k \in [l, h]$ are randomly generated.

The $k_i$-random ticks and the $k_i$-periodic ticks are used for random testing and counterexample-guided narrowing, respectively. For periodic ticks, the offset $c$ is randomly chosen to avoid overlaps in refinement loops.

Since the number of test data grows to the power of the $2m$-th degree, they can easily explode. Based on observations of Example 4, we optimize test data generation.

- **Narrowing test domain:** We ignore irrelevant noise symbols. Further, we can reduce the subdomain if the coefficient CI of a noise symbol does not contain $0$.
- **More ticks for more sensitive noise symbols:** We set ticks for each input range proportional to sensitivity of noise symbols.
- **Pre-evaluation:** We can avoid test cases if their pre-evaluations are within the RE threshold bound.

The narrower the input ranges, the more precise the ORE analysis result. When refining the ORE analysis, we divide the input domain into two by splitting the input range of the noise symbol $\varepsilon_i$ that has the largest $|\tau_i|$. Then, the input subdomains are checked in the breadth-first manner. Our heuristics firstly tries one that contains the test case resulting the largest (absolute) RE.

Counterexample-guided narrowing is implemented as a prototype tool CANAT (C ANalyzer and T ester) on CANA. Figure 2 shows the construction of CANAT, in which CIL\(^1\) and Weighted PDS\(^2\) are used as a preprocessor and a back-end engine, respectively.

**Narrowing test domain**

**Example 5.** In Example 4, $\varepsilon_2$ is an irrelevant noise symbol. Thus, we can fix the value of $y_1$ to $0 \in [-10, 10]$.

**Lemma 1.** If $0 < u \leq v$, $-u \leq [u, v] \cap [-\frac{u}{2}, \frac{v}{2}] \leq u$.

Let $\tau_i = [u_i, v_i]$ be the coefficient of a noise symbol $\varepsilon_i$ with $u_i > 0$. Let $\tau_i = [-\frac{u_i}{2}, \frac{u_i}{2}]$. Then, $\tau_i \subseteq [-u, u]$ by Lemma 1. Hence, we can ignore the noise range $\tau_i$ since it is bounded by the boundary cases $\varepsilon_i = -1$ and $\varepsilon_i = 1$.

**Example 6.** In Example 4, the coefficient CIs $\tau_3$ and $\tau_4$ do not include $0$. Thus, Lemma 1 enables us to ignore

- $\tau_3 = \left[\left[-\frac{0.031250}{0.031250}, \frac{0.031250}{0.031250}\right], \left[-0.63589, 0.63589\right]\right] = \left[-0.63589, 0.63589\right]$.

**5. PRELIMINARY EXPERIMENTS**

The first column in Table 1 shows the names of 4 programs, with the following 40 settings for each.
1. P2 (Example 1): The initial range \([-1, 3], [-10, 10]\), \(fp = [7, 8, 9, 10]\), and \(\theta \in [0.001 + 0.002i | 0 \leq i \leq 9]\).
2. P5 (1 - x^3 - \theta^2 x - \theta^3 x^4 - 5x^5): The initial range \([0, 1]\), \(fp = [7, 8, 9, 10]\), and \(\theta \in [0.001 + 0.001i | 0 \leq i \leq 9]\).
3. Sine (by Taylor expansion up to degree 21): The initial range \([0, 1]\), \(fp \in [7, 8, 9, 10]\), and \(\theta \in [0.001 + 0.005i | 0 \leq i \leq 9]\).
4. subMpeg (a fragment taken from the mpeg4 decoder reference algorithm, consisting of an 8 x 8 loop): The initial range \([0.30], fp \in [7, 8, 9, 10]\), and \(\theta \in [0.001 + 0.05i | 0 \leq i \leq 9]\).

Table 1 compares experimental results between
- ORE analysis (CANA) followed by random testing (Random test) with 200 instances, and
- counterexample-guided narrowing by repeating ORE analysis (Analysis) and testing (Test) 10 times. Each test executes 20 instances.

\%Checked columns show the percentages (among 40 settings) of correct detections (i.e., either safe or violation). Although the experiment is just preliminary, it shows considerable improvement. The execution times with Intel(R) Xeon(TM) 3.60GHz and 3.37GB RAM of whole 40 settings) of correct detections (i.e., either safe or violation).

### Acknowledgments

This research is partially supported by STARC (Semiconductor Technology Academic Research Center). The authors thank anonymous reviewers, Prof. Shin Nakajima (NII), Dr. Hirokazu Anai (Fujitsu), and our colleagues for their valuable comments.

### 8. REFERENCES


