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.

1. INTRODUCTION

Machine representations of real numbers are typically floating point numbers. Our targets are digital signal processing (DSP), such as mpeg4 decoders. Their implementations on chips or in embedded systems often use *fixed point numbers* for lower cost and higher speed. However, the direct floating-point-to-fixed-point conversion from their reference algorithms frequently causes visible overflow and roundoff errors (OREs).

Several ORE analyses have been proposed; for OREs of floating point numbers [2, 3, 5], and for OREs of fixed point numbers [1, 6]. We adopt an ORE analyzer CANA [6], which combines an abstraction to *extended affine interval* (EAI) [3, 6] and weighted model checking on an acyclic model. Note that CANA focuses on programs with bounded loops and arrays with fixed length, which typically appear in DSP encoder/decoder reference algorithms.

Due to the over approximation, CANA may report spurious counterexamples. Fortunately, we can compute exact roundoff errors (REs) between the floating point and fixed point numbers. Further, the result of CANA clarifies: (1) the variables irrelevant to the RE, (2) the variables most influential to the RE, and (3) the ranges of inputs that are most likely to maximize the RE. These observations effectively narrow the focus of test data generation.

This paper proposes a new testing refinement loop, called *counterexample-guided narrowing*, by combining CANA and testing. They are composed as a prototype tool CANAT, in which the analysis results are used not only for RE estimation, but also for finding dominant RE factors in inputs. Preliminary experiments show that the counterexample-guided narrowing approach considerably improves both testing and static analysis. We will explain this methodology by a running example (Example 1), and details have been reported in [7].

2. EXTENDED AFFINE INTERVAL

A *Classical interval* (CI) is an interval $[l, h]$ with $l \leqslant h$. Let $[l_1, h_1] \leqslant [l_2, h_2]$ if $h_1 \leqslant l_2$ and $|[l, h]| = max(|l|, |h|)$. For two CIs $\overline{x}, \overline{y}$, and $\circ \in \{+, -, \times\}$, $\overline{x} \mathbin{\overline{\circ}} \overline{y}$ is the smallest CI that contain all possible values of $x \circ y$ for each $x \in \overline{x}, y \in \overline{y}$ (e.g., $[l_1, h_1] \mathbin{\overline{+}} [l_2, h_2] = [l_1 + h_1, l_2 + h_2]$).

Definition 1. ([3]) *An* Extended affine interval *(EAI)* $\hat{x}$ *is an expression*

$$\hat{x} = \overline{x}_0 \ + \ \sum_{k=1}^{n} \overline{x}_k \varepsilon_k$$

where ε_i *is a noise symbol and* $\overline{x}_i$ *is a CI for each* $i (\leqslant n)$.

Each ε_i is interpreted as a value in $[-1, 1]$. In [3], the operations on EAIs are designed for under approximation. In [6], we proposed them for over approximation.

Definition 2. **EAI arithmetic** *consists of operations* $\{\hat{+}, \hat{-}, \hat{\times}, \hat{\div}\}$ *on pairs of EAIs. Let* $\hat{x} = \overline{x}_0 \ + \ \sum_{i=1}^{n} \overline{x}_i \varepsilon_i$, $\hat{y} = \overline{y}_0 \ + \ \sum_{i=1}^{n} \overline{y}_i \varepsilon_i$, $\overline{X} = \sum_{k=1}^{n} (\overline{x}_k \mathbin{\overline{\times}} [-1, 1])$, *and* $\overline{Y} = \sum_{k=1}^{n} (\overline{y}_k \mathbin{\overline{\times}} [-1, 1])$. *Then,*

$$\begin{array}{rcl}
\hat{x} \mathbin{\hat{+}} \hat{y} & = & (\overline{x}_0 \mathbin{\overline{+}} \overline{y}_0) \ + \ \sum_{i=1}^{n} (\overline{x}_i \mathbin{\overline{+}} \overline{y}_i) \varepsilon_i \\
\hat{x} \mathbin{\hat{-}} \hat{y} & = & (\overline{x}_0 \mathbin{\overline{-}} \overline{y}_0) \ + \ \sum_{i=1}^{n} (\overline{x}_i \mathbin{\overline{-}} \overline{y}_i) \varepsilon_i \\
\hat{x} \mathbin{\hat{\times}} \hat{y} & = & \overline{x}_0 \mathbin{\overline{\times}} \overline{y}_0 \ + \ \sum_{i=1}^{n} (\overline{x}_0 \mathbin{\overline{\times}} \overline{y}_i \mathbin{\overline{+}} \overline{x}_i \mathbin{\overline{\times}} \overline{y}_0) \varepsilon_i \ + \ B \\
\hat{x} \mathbin{\hat{\div}} \hat{y} & = & \hat{x} \mathbin{\hat{\times}} (\frac{1}{\hat{y}}) \quad \text{if } 0 \notin \overline{x}_0 \ + \ \sum_{k=1}^{n} \overline{x}_k \mathbin{\overline{\times}} [-1, 1]
\end{array}$$

where:

$$B = \begin{cases} \sum_{i=1}^{n} (\overline{x}_i \mathbin{\overline{\times}} \overline{Y}) \varepsilon_i & \text{if } \overline{Y} \ \subseteq \ \overline{X} \\ \sum_{i=1}^{n} (\overline{X} \mathbin{\overline{\times}} \overline{y}_i) \varepsilon_i & \text{otherwise} \end{cases}$$

and $\frac{1}{\hat{y}}$ *is computed by Chebyshev approximation* [7, 8].

We define the conversion between CI and EAI as follows:

- *CI to EAI*: Given a CI $\overline{x} = [l, \ h]$, the *EAI coercion* is $\hat{x} = \frac{l+h}{2} \ + \ \frac{h-l}{2} \varepsilon$.
- *EAI to CI*: Given an EAI $\hat{x} = \overline{x}_0 \ + \ \sum_{i=1}^{n} \overline{x}_i \varepsilon_i$, the *EAI projection* is $\overline{x} = \overline{x}_0 \mathbin{\overline{+}} \sum_{i=1}^{n} \overline{x}_i \mathbin{\overline{\times}} [-1, \ 1]$.

3. ORE ANALYSIS

This section briefly reviews the ORE analyzer CANA [6, 7]. Throughout the paper, we focus on the RE only.

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 8×8, 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.

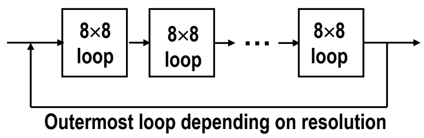

Figure 1: Structure of mpeg4 decoder reference algorithm

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

```
    /* CANAT
    CANAT ALL sign 11 4
    P2 x range -1 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* $x \in [-1, 3]$, $y \in [-10, 10]$, *and*
- *the RE threshold* $\theta = 0.26$.

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 EAIs, which describe the fixed point number range and the roundoff error range. As a convention, we will refer to those pairs by $\widehat{x}_f, \widehat{x}_r$ and $\widehat{y}_f, \widehat{y}_r$, respectively.

DEFINITION 3. *For two pairs of EAIs* $(\hat{x}_f,\ \hat{x}_r)$, $(\hat{y}_f,\ \hat{y}_r)$, $\{\hat{\boxplus}, \hat{\boxminus}, \hat{\boxtimes}, \hat{\boxdiv}\}$ *are defined as:*

$$
\begin{aligned}
(\hat{x}_f,\ \hat{x}_r) \mathbin{\hat{\boxplus}} (\hat{y}_f,\ \hat{y}_r) &= (\hat{x}_f \mathbin{\hat{+}} \hat{y}_f,\ \hat{x}_r \mathbin{\hat{+}} \hat{y}_r \mathbin{\hat{+}} \widehat{\delta})\\
(\hat{x}_f,\ \hat{x}_r) \mathbin{\hat{\boxminus}} (\hat{y}_f,\ \hat{y}_r) &= (\hat{x}_f \mathbin{\hat{-}} \hat{y}_f,\ \hat{x}_r \mathbin{\hat{-}} \hat{y}_r \mathbin{\hat{+}} \widehat{\delta})\\
(\hat{x}_f,\ \hat{x}_r) \mathbin{\hat{\boxtimes}} (\hat{y}_f,\ \hat{y}_r) &= (\hat{x}_f \mathbin{\hat{\times}} \hat{y}_f,\ \hat{x}_r \mathbin{\hat{\times}} \hat{y}_f \mathbin{\hat{+}} \hat{x}_f \mathbin{\hat{\times}} \hat{y}_r \mathbin{\hat{+}}\\
&\qquad \hat{x}_r \mathbin{\hat{\times}} \hat{y}_r \mathbin{\hat{+}} \widehat{\delta})\\
(\hat{x}_f,\ \hat{x}_r) \mathbin{\hat{\boxdiv}} (\hat{y}_f,\ \hat{y}_r) &= (\hat{x}_f \mathbin{\hat{\div}} \hat{y}_f, (\hat{x}_f \mathbin{\hat{+}} \hat{x}_r) \mathbin{\hat{\div}} (\hat{y}_f \mathbin{\hat{+}} \hat{y}_r)\\
&\qquad \mathbin{\hat{-}} \hat{x}_f \mathbin{\hat{\div}} \hat{y}_f \mathbin{\hat{+}} \widehat{\delta})
\end{aligned}
$$

where $\widehat{\delta} = [-b^{-fp}/2,\ b^{-fp}/2]$.

Roundoff error analysis

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

$$
\begin{aligned}
\widehat{x}_f &= [1,\ 1]\ +\ [2,\ 2]\varepsilon_1 \quad \textit{and} \quad \widehat{x}_r = [2^{-5},\ 2^{-5}]\varepsilon_3,\\
\widehat{y}_f &= [0,\ 0]\ +\ [10,\ 10]\varepsilon_2 \quad \textit{and} \quad \widehat{y}_r = [2^{-5},\ 2^{-5}]\varepsilon_4.
\end{aligned}
$$

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 $\hat{\boxtimes}$.

$$
\begin{aligned}
\widehat{rst}_r^{(2)} &= [-0.031250, 0.031250]\ +\ [-0.123091, 0.123091]\varepsilon_1\\
&\quad +\ [0.059615, 0.065385]\varepsilon_3\\
\widehat{rst}_r^{(3)} &= 3 \mathbin{\hat{\times}} \hat{x}_r = [0.093750, 0.093750]\varepsilon_3
\end{aligned}
$$

They are merged as:

$$
\begin{aligned}
\widehat{rst}_r^{(2,3)} &= [-0.031250, 0.031250]\ +\ [-0.123091, 0.123091]\varepsilon_1\\
&\quad +\ ([0.059615, 0.065385] \cup [0.093750, 0.093750])\varepsilon_3\\
&= [-0.031250, 0.031250]\ +\ [-0.123091, 0.123091]\varepsilon_1\\
&\quad +\ [0.059615, 0.093750]\varepsilon_3
\end{aligned}
$$

At line (4), we obtain the RE of rst by $\hat{\boxminus}$*:*

$$
\begin{aligned}
\widehat{rst}_r^{(4)} &= [-0.031250, 0.031250]\ +\ [-0.123091, 0.123091]\varepsilon_1\ +\\
&\quad [0.059615, 0.09375]\varepsilon_3\ +\ [-0.031250, -0.031250]\varepsilon_4
\end{aligned}
$$

The RE $\hat{r}$ *of rst (i.e.,* $\widehat{rst}_r^{(5)}$*) coincides with* $\widehat{rst}_r^{(4)}$*, and is bounded by* $[-0.279341, 0.279341]$. *We denote*

$$
\hat{r} = \overline{r}_0\ +\ \sum_{i=1}^{4} \overline{r}_i \varepsilon_i
$$

and refer to the coefficient CI of ε_i *(in* $\widehat{rst}_r^{(4)}$*) by* $\overline{r}_i$.

Note that, during this analysis, over approximations occur at the conditional branch (line (1)) with $\cup$ *and the multiplication (line (2)) with* $\widehat{\delta} = [-0.031250, 0.031250]$.

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* $\hat{r} = [-0.06348, 0.279341] \nsubseteq [-0.26, 0.26]$, *whereas testing detects REs at most* $0.219727 \in [-0.26, 0.26]$.

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* $\overline{r}_2 = [0, 0]$, ε_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* $|\overline{r}_1| = 0.123091 = max\{|\overline{r}_1|, \ldots, |\overline{r}_4|\}$, ε_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* $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$ *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 $[l_i, h_i]$ (for $i \leqslant 2m$), 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* $k \geqslant 1$,

- *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 device the input domain into two by splitting the input range of the noise symbol ε_i that has the largest $|\overline{r}_i|$. 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 Tester*) 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,* ε_2 *is an irrelevant noise symbol. Thus, we can fix the value of* y_f *to* 0 *(*$\in [-10, 10]$*).*

Lemma 1. *If* $0 < u \leqslant v$, $-u \leqslant [u, v] \,\overline{\times}\, [-\frac{u}{v}, \frac{u}{v}] \leqslant u$.

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

Example 6. *In Example 4, the coefficient CIs* $\overline{r}_3$ *and* $\overline{r}_4$ *do not include* 0. *Thus, Lemma 1 enables us to ignore*

- $\overline{\varepsilon}_3 = [-\frac{0.059615}{0.09375}, \frac{0.059615}{0.09375}] = [-0.63589, 0.63589]$,

[1] http://hal.cs.berkeley.edu/cil/

[2] http://www.fmi.uni-stuttgart.de/szs/tools/wpds/

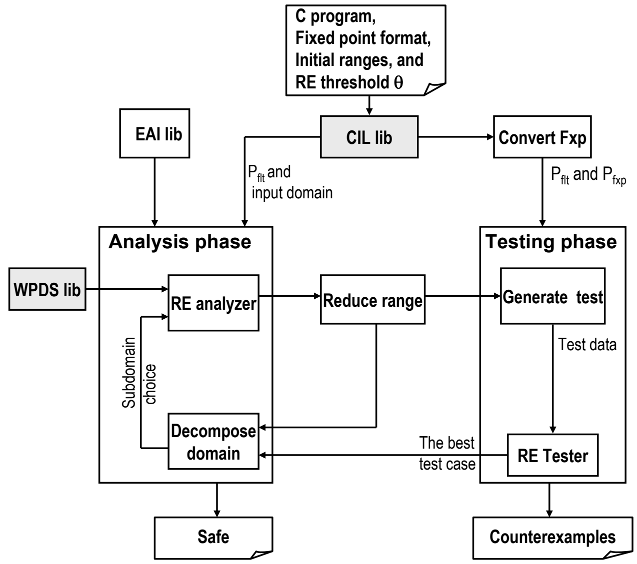

Figure 2: CANAT system

- $\overline{\varepsilon}_4 = [-\frac{-0.031250}{-0.031250}, \frac{-0.031250}{-0.031250}] = [-1, 1]$.

Let

$$D_{max} = ([1,1] \ + \ [2,2]\varepsilon_1, 0, [2^{-5}, 2^{-5}]\varepsilon_3^+, [2^{-5}, 2^{-5}]\varepsilon_4^+)$$
$$D_{min} = ([1,1] \ + \ [2,2]\varepsilon_1, 0, [2^{-5}, 2^{-5}]\varepsilon_3^-, [2^{-5}, 2^{-5}]\varepsilon_4^-)$$

where $\varepsilon_3^+ = [0.63589, 1]$, $\varepsilon_3^- = [-1, -0.63589]$, $\varepsilon_4^+ = [1, 1]$, *and* $\varepsilon_4^- = [-1, -1]$. *They contain test cases that maximize and minimize REs, respectively. Their EAI projections are*

$$D_{max} = ([-1,3], 0, [0.019872, 0.03125], -0.03125)$$
$$D_{min} = ([-1,3], 0, [-0.03125, -0.019872], 0.03125)$$

More ticks for more sensitive noise symbols

We will choose k_i (the number of ticks in Definition 4) proportional to $|\overline{r}_i|$ (the coefficient of ε_i) such that the product of all k_i's is approximately the required number of test data.

Example 7. *We focus on* D_{max} *in Example 6. Our heuristics sets the tick frequencies* t_1 *and* t_3*, proportional to* $|\overline{r}_1| = 0.123091$ *and* $|\overline{r}_3| = 0.09375$ *(in Example 2), respectively. Thus, if we intend to generate* 200 *test cases,* $t_1 \approx |\overline{r}_1| \cdot \sqrt{\frac{200}{|\overline{r}_1| \cdot |\overline{r}_3|}} \approx 16$ *and* $t_2 \approx |\overline{r}_3| \cdot \sqrt{\frac{200}{|\overline{r}_1| \cdot |\overline{r}_3|}} \approx 12$.

Narrowing Input Domains for Next Round

Example 8. *In Example 7, no test case from* D_{max} *violates the RE threshold bound. Since* ε_1 *is the most sensitive symbol in* D_{max}*, the range of* $x_f = [-1, 3]$ *is divided into two new subranges* $[-1, 1]$ *and* $[1, 3]$*, which compose two subdomains* D^1_{max} *and* D^2_{max}*. CANA reports that:*

- *the REs over* D^1_{max} *lie in* $[-0.078914, 0.187500]$
- *the REs over* D^2_{max} *lie in* $[0.017318, 0.250977]$

CANA also reports similarly for D_{min}.

5. PRELIMINARY EXPERIMENTS

The first column in Table 1 shows the names of 4 programs, with the following 40 settings for each.

Input program	CANA and Random test				CANAT			
	CANA	Random test	Time (sec)	%Checked	Analysis	Test	Time(sec)	%Checked
P2	15	11	7	65.00%	20	18	13	95.00%
P5	9	15	14	60.00%	12	19	24	77.50%
Sine	19	7	37	65.00%	21	8	81	72.50%
subMpeg	11	11	65	55.00%	11	19	121	75.00%

Table 1: Experimental results of CANAT

1. **P2** (Example 1): The initial range $([-1, 3], [-10, 10])$, $fp \in \{7, 8, 9, 10\}$, and $\theta \in \{0.001 + 0.002i \mid 0 \leqslant i \leqslant 9\}$.
2. **P5** $(1-x+3x^2-2x^3+x^4-5x^5)$: The initial range $[0, 1]$, $fp \in \{7, 8, 9, 10\}$, and $\theta \in \{0.001 + 0.01i \mid 0 \leqslant i \leqslant 9\}$.
3. **Sine** (by Taylor expansion up to degree 21): The initial range $[0, 1]$, $fp \in \{7, 8, 9, 10\}$, and $\theta \in \{0.001 + 0.005i \mid 0 \leqslant i \leqslant 9\}$.
4. **subMpeg** (a fragment taken from the mpeg4 decoder reference algorithm, consisting of an 8×8 loop): The initial range $[0, 30]$, $fp \in \{7, 8, 9, 10\}$, and $\theta \in \{0.001 + 0.05i \mid 0 \leqslant i \leqslant 9\}$.

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 for each are shown in **Time** columns.

6. RELATED WORK

We obey the semantics of propagation of REs as shown in [4]. Affine interval (AI) is proposed in [8]. RE analyses, which over approximate REs between real numbers and floating point numbers, are designed based on AI [2, 5]. They are implemented as a tool FLUCTUAT. To sandwich REs from both sides, an RE analysis for under approximation is proposed based on EAI, the mean value theorem, and Kaucher arithmetic [3].

For the floating point-to-fixed-point conversion, Fang [1] proposed an RE analysis based on AI, intended for DSP applications. We focus on the same problem, but with EAI. We also adapted sophisticated weighted model checking, whereas they adapted direct bit-vector encoding. They also applied probabilistic reduction of the search space.

There are many works on sophisticated floating-point-to-fixed-point translations, aimed at hard-wired implementations. A dataflow graph is traced in a backward manner to propagate the output requirement by CI-based estimation, to optimize the fixed point number formats [9]. Then, the translation is verified by testing. Our work is complementary, and can be a substitute for the final verification.

7. CONCLUSION

The contribution of this paper is twofold:

- Counterexample-guided narrowing, in which RE analyses and testing refine each other.
- A prototype tool, CANAT, and its preliminary experiments. We would like to emphasize that, although our experiments are still small, the results are encouraging.

For future work, we plan:

- Experiments on more realistic C programs. We hope that the compositional nature of DSP reference algorithms (Figure 1) would help scalability.
- More sophisticated input domain decomposition strategy, e.g., the use of differences between the RE threshold and the maximum RE during testing.
- Automatic source code correction to reduce REs, e.g., exchanging the ordering of arithmetic operations by associative and distributive laws.

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

[1] C. F. Fang. Probabilistic Interval-Valued Computation: Representing and Reasoning about Uncertainty in DSP and VLSI Design. Ph.D.Diserration, Carnegie Mellon University, 2005.

[2] E. Goubault and S. Putot. Static analysis of numerical algorithms. *13th Int. Static Analysis Symposium*, Springer LNCS 4134, pp.18-34, 2006.

[3] E. Goubault and S.Putot. Under approximations of computations in real numbers based on generalized affine arithmetic. *14th Int. Static Analysis Symposium*, Springer LNCS 4634, pp.137-152, 2007.

[4] M. Martel. Semantics of roundoff error propagation in finite precision calculations. *Higher-Order and Symbolic Computation*, 19(1), pp.7-30, 2006.

[5] M. Martel. Static analysis of the numerical stability of loops. *9th Int. Static Analysis Symposium*, Springer LNCS 2477, pp.133-150, 2002.

[6] D. T. B. Ngoc and M. Ogawa. Overflow and Roundoff Error Analysis via Model Checking. *7th IEEE Int. Conf. on Software Engineering and Formal Methods*, pp.105-114, 2009.

[7] D. T. B. Ngoc and M. Ogawa. Combining Testing and Static Analysis to Overflow and Roundoff Error Detection. JAIST Tech. Reports IS-RR-2010-004, 2010.

[8] J. Stolfi and L.H. de Figueiredo. An introduction to affine arithmetic. *TEMA Tend. Mat. Apl. Comput.*, 3(4), pp. 297-312, 2003.

[9] S. Wijaya and A. Cantoni. A Java Simulation Tool for Fixed-Point System Design. 2nd Int. ICST Conf. on Simulation Tools and Techniques, pp. 1-10, 2009.