Accurate Evaluation
of Arithmetic Expressions

Matthieu Martel

DALI - Université de Perpignan Via Domitia
LIRMM - CNRS: UMR 5506 - Université Montpellier 2, France

matthieu.martel@univ-perp.fr

NUMERICAL AND SYMBOLIC ABSTRACT DOMAINS (NSAD 2012)
Some static analyzers based on abstract interpretation:

**Astrée:** ANSI C, RTEs, integer and floating-point arithmetics

**Code Contracts:** Visual Studio, contract violations, integer arithmetic

**Fluctuat:** ANSI C, numerical inaccuracies, floating-point arithmetic

Compute subtle properties on codes

Industrial success stories


[M. Fahndrich and F. Logozzo, *Static contract checking with Abstract Interpretation*, FoVeOOS 2010]

Errors due to computer arithmetic are difficult to detect by hand. . .

. . . and to correct!

Static analyzers detect bugs

Natural extension: propose corrections to the programmer
Approach (1/2)

Capture the programmer’s intention

the expressions would return the expected results if the arithmetic were exact

Synthesize new expressions which implements the intention

the new expressions introduce less errors in the computer arithmetic

Correctness

The source and synthesized expressions are mathematically equal
Optimize expressions given ranges for the variables

\[ x^2 - 2x + 1 \text{ more accurate than } (x - 1) \times (x - 1) \text{ when } x \in [0.1, 1.0] \text{ in f.p. arithmetic} \]

Too many mathematically equivalent expressions: need for abstraction

\((2n - 1)!! \text{ ways to sum } n \text{ terms } \left( \frac{2n!}{n!(n+1)!} \right) \text{ parsings} \)

\[ e = (x - 1) \times \ldots \times (x - 1) \text{ } \overbrace{n \text{ times}}^{n \text{ times}} \]

\[ \begin{array}{c|c}
2.3 \cdot 10^6 \text{ equivalent expressions for } n = 5 \\
1.3 \cdot 10^9 \text{ equivalent expressions for } n = 6 \\
\end{array} \]

Computer arithmetics:

integer, floating-point, fixed-point and interval
Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Integer Arithmetic

Bounded integers

Example: \[ \text{int} = [m, M] \text{ with } m = -2^{31} \text{ and } M = 2^{31} - 1 \]

Operations (wrap up)

\[ M + 1 = m \quad m - 1 = M \]

Example with 32 bits signed integers: \( x = 2^{30} \) and \( y = -2^{15} \)

\[
\frac{2 \times x}{3} + y = -715860650 \quad \text{and} \quad 2 \times \frac{x}{3} + y = 715795114
\]

Synthesis of expressions

Generate expression mathematically equal to the original

And which minimizes the maximal intermediary result in absolute value

[F. Logozzo and T. Ball. Modular and verified automatic program repair. OOPSLA’12]
Floating-Point Arithmetic: the IEEE754 Standard

Binary64 normalized floating-point numbers: \[ \pm 1.d_1d_2\ldots d_p 2^e \]

Precision \( p = 52, \) \(-1022 \leq e \leq 1023\)

Example of distribution (simplified set, \( \beta = 2, \) \( p = 3, \) \(-1 \leq e \leq 1\)):

Special values: \( \pm\infty, \) NaN, denormalized numbers
IEEE754 Standard: Rounding Modes

4 rounding modes: towards $\pm\infty$, to the nearest, towards 0

$\circ_r : \mathbb{R} \rightarrow \mathbb{F}$ computes the roundoff of a real number in rounding mode $r$

For elementary operations $\odot \in \{+,-,\times,\div,\sqrt{\cdot}\}$:

$$x \odot_r y = \circ_r(x \ast y)$$

Floating-Point Arithmetic

Example

e = 2.7182818 \ldots \text{computed using Bernouilli's formula:}

\[ e = \lim_{n \to +\infty} u_n \quad \text{with} \quad u_n = \left(1 + \frac{1}{n}\right)^n, \quad n > 0 \]

In double precision

\[ u_8 = 2.718282 \quad u_{14} = 2.716110 \quad u_{16} = 3.035035 \quad u_{17} = 1.0 \]

Synthesis of expressions

Generate expression mathematically equal to the original

And which minimizes the roundoff error \(|r_{\text{exact}} - r_{\text{float}}|\) on the result
Fixed-Point Arithmetic

Values

In format $\langle w, i \rangle$, $b_{w-1} \ldots b_0$ represents:

$$-b_{w-1} \cdot 2^{i-1} + \sum_{j=2}^{j=w} b_{w-j} \cdot 2^{i-j}$$

Operations
Fixed-Point Arithmetic

Synthesis of expressions

Generate expression mathematically equal to the original

And which minimizes the sum of $w$ of the formats of the intermediary results

Example: $x^2 - 6x + 9$ with $x$ in the format $\langle 5, 3 \rangle$

$x^2 - 6x + 9$: 68 bits, \((x - 3) \times (x - 3)\): 40 bits
Interval Arithmetic

Values and operations

Intervals with floating-point bounds

\[ [x, \bar{x}] \boxplus [y, \bar{y}] = [x \ominus -\infty y, \bar{x} \oplus +\infty \bar{y}] \]

Absence of relation between variables: over-approximations

Example: \( f(x) = \frac{x}{x-2} \)

\( f([3, 4]) = [1.5, 4] \quad f(x) = g(x) = 1 + \frac{2}{x-2} \quad g([3, 4]) = [2, 3] \)

Synthesis of expressions

Generate expression mathematically equal to the original

And which minimizes the width of the resulting interval
Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Non-Standard Semantics: Overview

Non-standard values:

Record the results of the evaluation in computer and exact arithmetics

Pairs \((\hat{v}, v)\) with \(\hat{v}\) machine value and \(v\) exact value

Depending on the semantics \((\hat{v}, v)\) belongs to

\[
\text{int} \times \mathbb{Z} \quad \text{float} \times \mathbb{R} \quad \text{fixed} \times \mathbb{R} \quad \text{float} \times \text{float} \times \wp(\mathbb{R})
\]

Non-standard semantics

Record information related to the evaluation in the computer arithmetic

Helps to select an expression adapted to the computer arithmetic

Enables one to prove the correctness of the synthesis of a new expression
Objective: find a trace which minimizes $m$
Objective: find a trace which minimizes $W$
Floating-point and Interval Non-Standard Semantics

float: $\hat{v} = \hat{v}_1 \odot \sim \hat{v}_2 \quad v = v_1 \ast v_2$

$\langle (\hat{v}_1, v_1) \ast (\hat{v}_2, v_2), \rho \rangle \rightarrow_{\text{float}} \langle (\hat{v}, v), \rho \rangle$

interval: $\hat{v} = v_1 \ast \hat{v}_2 \quad v = \{ x \ast y : x \in v_1, y \in v_2 \}$

$v$ set of points

$\langle (\hat{v}_1, v_1) \ast (\hat{v}_2, v_2), \rho \rangle \rightarrow \llbracket \langle (\hat{v}, v), \rho \rangle \rrbracket$

Objective: find a trace which minimizes $|\hat{v} - v|$ or width($\hat{v}$)
Collecting Semantics $[e] \Theta$ for set $\Theta$ of environments

Observational abstractions $\alpha \circ$

Keeps exact results, discards computer results in non-standard values
Non-Standard Semantics: Correctness (2/2)

It is correct to replace $e$ by $e'$ if for any $\Theta$, $\llbracket e \rrbracket \Theta = S$, $\llbracket e' \rrbracket \Theta = S'$ and:

$$\{ \alpha_O(s) : s \in S \} = \{ \alpha_O(s') : s' \in S' \}$$

[P. Cousot and R. Cousot; Systematic design of program transformation frameworks by abstract interpretation, POPL'02]
Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Abstraction of Equivalent Expressions

Too many mathematically equivalent expressions: need for abstraction

\[(2n - 1)!!\] ways to sum \(n\) terms \(\left(\frac{2n!}{n!(n+1)!}\right)\) parsings

\[e = (x - 1) \times \ldots \times (x - 1)\] \(n\) times

- 2.3 \(\times\) 10^6 equivalent expressions for \(n = 5\)
- 1.3 \(\times\) 10^9 equivalent expressions for \(n = 6\)

Two abstractions:

**EUD-k:** Identify expressions whose syntactic trees are Equal Up to Depth \(k\)

[M. Martel, *Semantics-based transformation of arithmetic expressions*, SAS'07]

**APEGs:** Abstraction based on Abstract Program Expression Graphs

Expression Simplification

Definition of Mathematically Equivalent Expressions

\( \mathcal{R} \subseteq \text{Expr} \times \text{Expr} \) binary relation on the set of expressions

\( \mathcal{R} \) identifies mathematically equivalent expressions

For example, \( \mathcal{R} \) may contain associativity or distributivity:

\[
\left\{ (e_1 + (e_2 + e_3), (e_1 + e_2) + e_3) : e_1, e_2, e_3 \in \text{Expr} \right\} \subseteq \mathcal{R}
\]

\[
\left\{ (e_1 \times (e_2 + e_3), e_1 \times e_2 + e_1 \times e_3) : e_1, e_2, e_3 \in \text{Expr} \right\} \subseteq \mathcal{R}
\]
EUD-k Abstraction (2/3)

Generation of a set of equivalent expressions

\[ \rightarrow_k \text{ on states } \langle E, K \rangle \in \wp(\text{Expr}) \times \wp(\text{Expr}) \]

\[ e \in E \quad e \not\in e' \quad \neg e' \not\in K \]

\[ \langle E, K \rangle \rightarrow_k \langle \{e'\} \cup E, \{\neg e' \} \cup K \rangle. \]

Initial state \( \langle \{e\}, \{\neg e\} \rangle \)
Compute maximal set \( E \) of equivalent expressions such that
\[
e_1, e_2 \in E \Rightarrow \neg e_1^k \neq \neg e_2^k
\]

\( E \) under-approximation of the set of expressions \( \mathcal{R} \)-equivalent to \( e \)

Exponential in \( k \) (user-defined parameter)

Example

\( e = c \times ((a + a) + b) \)

\[
S_1 = \{ c \times ((a + a) + b), c \times (a + a) + c \times b \} \quad \text{if } k = 1,
\]

\[
S_2 = \left\{ (a + a) + b \times c, (a + (a + b)) \times c, (a + a) \times c + b \times c, a \times c + (a + b) \times c \right\} \quad \text{if } k = 2.
\]
Represent many equivalent expressions in polynomial size

APEGs contain equivalence classes

APEGs contain abstraction boxes: \( *(e_1, \ldots, e_n) \) (\( * \) assoc. and commut. )

\( *(e_1, \ldots, e_n) \) represents all the parsings of \( e_1 \ast \ldots \ast e_n \)

Box elements are constants, expressions or abstraction boxes
Abstract Program Expression Graphs (2/2)

Set $\mathcal{A}(p)$ of expressions contained inside an APEG $p$

\[\mathcal{A}(p) = \left\{ \begin{array}{l}
((a + a) + b) \times c, ((a + b) + a) \times c, ((b + a) + a) \times c,
((2 \times a) + b) \times c, c \times ((a + a) + b), c \times ((a + b) + a),
c \times ((b + a) + a), c \times ((2 \times a) + b), (a + a) \times c + b \times c,
(2 \times a) \times c + b \times c, b \times c + (a + a) \times c, b \times c + (2 \times a) \times c
\end{array} \right\} \]
APEG construction

Rewriting rules applied up to saturation
[R. Tate, M. Stepp, Z. Tatlock, and S. Lerner, Equality saturation: A new approach to optimization, POPL'09]

Specific polynomial algorithms
[A. Ioualalen and M. Martel, A new abstract domain for the representation of mathematically equivalent expressions, SAS’12]
Correctness (1/2)

Abstract sets contain only equivalent expressions

Set $\mathcal{E}(e)$ of expressions equivalent to $e$ generated by $\rightarrow \in \wp(\text{Expr}) \times \wp(\text{Expr})$:

$$
e \in E \quad e \mathcal{R} e' \quad \frac{}{E \rightarrow \{e'\} \cup E}$$

EUD-$k$: $\langle \{e\}, \{\neg e^\neg_k\} \rangle \rightarrow_k^* \langle E^\#, K \rangle$

APEGs: $\mathcal{A}(p)$ set of expressions contained inside an APEG $p$ built from $e$

$E^\#$ and $\mathcal{A}(p)$ are under-approximations of $\mathcal{E}(e)$
Correctness (2/2)

concrete set of equivalent expressions

EUD-k states: expressions $\times$ expressions of limited depth

set of APEGs

\[
\gamma_1(\langle E, K \rangle) = \bigcup_{e \in E, \{e\} \rightarrow E'} E'
\]

\[
\gamma_2(p) = \bigcup_{e \in A(p), \{e\} \rightarrow E'} E'
\]
Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Abstract value: interval of integers \( (\text{int} \times \text{int}) \)

\[
[v, \bar{v}] = [v_1, \bar{v}_1] \star_{\text{int}} [v_2, \bar{v}_2] \\
m' = \max(|v_1|, |\bar{v}_1|, |v_2|, |\bar{v}_2|, |v|, |\bar{v}|, m)

\langle [v_1, \bar{v}_1] \star [v_2, \bar{v}_2], m, \rho \rangle \rightarrow^{\#}_{\text{int}} \langle [v, \bar{v}], m', \rho \rangle
\]
Abstract value: pair of intervals of floating-point numbers

First interval: abstracts the computer result (bounds rounded to the nearest)

second interval: safe approx. of the range of the exact result (under-approx.)
Generation of new Expressions

Select the expression which minimize $m$, $\hat{v} - v$, $W$ or $\text{width}(\hat{v})$

**EUD-k:**

Apply abstract semantics to all the expressions of the under-approximation

**APEGs:**

Algorithms to search inside the structure and for boxes
APEGs and Formula Synthesis: The Case of Boxes

An abstraction box represents \((2n - 1)!!\) expressions

Greedy heuristic:

At each step, select the terms \(a\) and \(b\) such that \(\text{error}(a \times b)\) is minimal

Complexity: \(O(n^2)\)

Example:

\[ +\left(a, b, c, d, e\right) \rightarrow +\left(a, c, e, +\left(b, d\right)\right) \rightarrow +\left(e, +\left(a, c\right), +\left(b, d\right)\right) \rightarrow +\left( +\left(e, +\left(a, c\right)\right), +\left(b, d\right)\right) \]

We synthetize \((e + (a + c)) + (b + d)\)
Simplest approach:

For each class, select the operation which yields the smallest error.

Complexity: $O(n)$
APEGs and Formula Synthesis: Improvement

Not recording only the operation which yields the smallest error

Minimize the error for one operator using the classes below

Generalization: Consider all the classes up to $k$ levels
Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Experimental Results (with Arnault Ioualalen)

Experiments performed using the IEEE754 Binary64 format

Summations

Developed univariate polynomials

Taylor Series

Method:

Generate as many equivalent source expressions as possible (all)

Select (abstract) datasets

Compare the synthetized implementation to the direct implementation for any source expression
Datasets for Summations

4 datasets:

> 0, 20% of large values $\approx 10^{16}$ among small values $\approx 10^{-16}$

> 0, 20% of large values among small and medium values $\approx 1$

20% of large values, both signs, among small and medium values

> 0 and < 0, few small values, as many medium and large values

2 interval widths:

Width = 10% of the central value of the interval

Width = $10^{-12}$ smaller than the central value of the interval
Summation of 9 Termes : 2 Millions Cases (Dataset 1)

> 0, 20% of large values $\approx 10^{16}$ among small values $\approx 10^{-16}$

Large intervals. (similar results for small intervals)
Summation of 9 Termes : 2 Millions Cases (Dataset 2)

> 0, 20% of large values among small and medium values \(\approx 1\)

Large intervals. (similar results for small intervals)
Summation of 9 Termes : 2 Millions Cases (Dataset 3)

20% of large values, both signs, among small and medium values
Summation of 9 Termes : 2 Millions Cases (Dataset 4)

> 0 and < 0, few small values, as many medium and large values
We consider the polynomial:

$$(x - 1)^n = \sum_{k=0}^{n} (-1)^k \times \binom{n}{k} \times x^k, \quad n \in [2, 6]$$

(1)

As $n$ increases, the roundoff error increases around the multiple root

Source expressions: all the parsings of (1), no factorization
Developed Polynomials with $n = 5$, 5670 cases

Red: initial error bounds
Developed Polynomials with $n = 6, 374220$ cases

Red: initial error bounds
Taylor Series Developments

\[ \cos x = \sum_{n=0}^{+\infty} (-1)^n \frac{x^{2n}}{(2n)!} \]

\[ \sin x = \sum_{n=0}^{+\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \]

\[ \ln(2 + x) = \sum_{n=1}^{+\infty} (-1)^{n-1} \frac{x^n}{n \times 2^n} \]

Development orders for \( \cos \): \( n \in \{4, 6, 8\} \)

Development orders for \( \sin \): \( n \in \{5, 7, 9\} \)

Development orders for \( \ln(2+x) \): \( n \in \{4, 5\} \)

Intervals centered on the root, width = 10% of central value
Result for \( \cos \) with \( n = 8 \), 30240 Cases

Red: initial error bounds

Blue: error bounds on synthetized expressions
Result for \( \cos \) with \( n = 9 \), 162855 Cases

Red: initial error bounds

Blue: error bounds on synthetized expressions
Result for $\ln(2 + x)$ with $n = 5$, 5670 Cases

Red: initial error bounds

Blue: error bounds on synthetized expressions
Fixed-Point Arithmetic (1/2)

Digital Filters

\[ o = \sum_{r=1}^{n} \sum_{l=1}^{m} l(i + r - 1, j + l - 1) \times k_{rl} \]

8 bits image: value of pixels between 0 and 255
Fixed-Point Arithmetic (2/2)

<table>
<thead>
<tr>
<th>Filter</th>
<th>$\text{Filter}_{\text{sum}}$</th>
<th>$\text{Filter}_{\text{opt}}^{\text{sum}}$</th>
<th>%Gain</th>
<th>Avg-IP</th>
<th>$\text{Filter}_{\text{sum}}$</th>
<th>$\text{Filter}_{\text{opt}}^{\text{sum}}$</th>
<th>%Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>183</td>
<td>176</td>
<td>3.8%</td>
<td>$\sim$ 139</td>
<td>442</td>
<td>393</td>
<td>11%</td>
</tr>
<tr>
<td>Laplacian 1</td>
<td>204</td>
<td>180</td>
<td>11.7%</td>
<td>$\sim$ 152</td>
<td>713</td>
<td>660</td>
<td>7.4%</td>
</tr>
<tr>
<td>Laplacian 2</td>
<td>248</td>
<td>246</td>
<td>0.8%</td>
<td>$\sim$ 204</td>
<td>779</td>
<td>723</td>
<td>7.1%</td>
</tr>
<tr>
<td>Prewit 1</td>
<td>178</td>
<td>163</td>
<td>8.4%</td>
<td>$\sim$ 133</td>
<td>638</td>
<td>569</td>
<td>10.8%</td>
</tr>
<tr>
<td>Prewit 2</td>
<td>184</td>
<td>169</td>
<td>8.1%</td>
<td>$\sim$ 136</td>
<td>644</td>
<td>571</td>
<td>11.3%</td>
</tr>
<tr>
<td>Rehauss 1</td>
<td>259</td>
<td>253</td>
<td>2.3%</td>
<td>$\sim$ 212</td>
<td>841</td>
<td>794</td>
<td>5.5%</td>
</tr>
<tr>
<td>Rehauss 2</td>
<td>249</td>
<td>242</td>
<td>2.8%</td>
<td>$\sim$ 201</td>
<td>724</td>
<td>678</td>
<td>6.3%</td>
</tr>
<tr>
<td>Robert 1</td>
<td>263</td>
<td>233</td>
<td>11.4%</td>
<td>$\sim$ 161</td>
<td>778</td>
<td>694</td>
<td>10.7%</td>
</tr>
<tr>
<td>Robert 2</td>
<td>266</td>
<td>233</td>
<td>12.4%</td>
<td>$\sim$ 153</td>
<td>781</td>
<td>694</td>
<td>11.1%</td>
</tr>
<tr>
<td>Sobel 1</td>
<td>198</td>
<td>176</td>
<td>11.1%</td>
<td>$\sim$ 145</td>
<td>769</td>
<td>678</td>
<td>11.8%</td>
</tr>
<tr>
<td>Sobel 2</td>
<td>194</td>
<td>176</td>
<td>9.2%</td>
<td>$\sim$ 144</td>
<td>752</td>
<td>678</td>
<td>10.6%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fractional part size</th>
<th>Error bound generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$1.2 \cdot 10^2$</td>
</tr>
<tr>
<td>6</td>
<td>$2.8 \cdot 10^1$</td>
</tr>
<tr>
<td>8</td>
<td>5.1</td>
</tr>
<tr>
<td>10</td>
<td>1.9</td>
</tr>
<tr>
<td>12</td>
<td>$3.4 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>14</td>
<td>$3.1 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>16</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Summary

- Introduction
- Computer Arithmetics
- Correctness of the Synthesis
- Abstraction of Sets of Equivalent Expressions
- Generation of New Expressions
- Experimental results
- Conclusion
Conclusion

Inter-expression transformation:

Transformation of several expressions

Control structures (conditions and loops), relations

Multi-criteria optimization (accuracy and time)

Insertion of additional computations to improve even more accuracy

Error free transformations

Example: Knuth’s TwoSum: \(\text{tmp} = a-(a+b); \ err = \text{tmp}+b\)

Tool under development: SARDANA

Open PhD position
Questions?