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)

Introduction (1/2)

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

[D. Kästner, S. Wilhelm, S. Nenova, P. Cousot, R. Cousot, J. Feret, A. Miné, L. Mauborgne, and X. Rival. *Astrée: Proving the absence of runtime errors*, ERTSS 2011]

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

[D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, and F. Vedrine. *Towards an industrial use of Fluctuat on safety-critical avionics software*, FMICS'12]

Introduction (2/2)

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

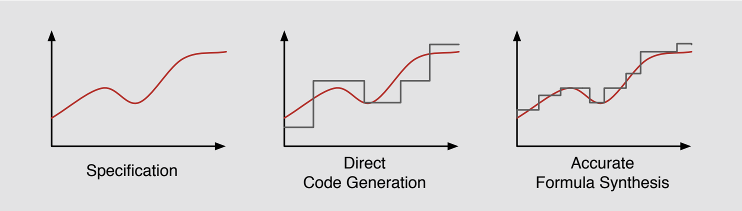
the expressions would return the expected results if the arithmetic were exact

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



Approach (2/2)

Optimize expressions given ranges for the variables

$x^2 - 2x + 1$ more accure than $(x-1) \times (x-1)$ when $x \in [0.1, 1.0]$ in f.p. arithmetic

Too many mathematically equivalent expressions: need for abstraction

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

$$\underbrace{e = (x-1) \times \ldots \times (x-1)}_{n \text{ times}}$$

$2.3 \cdot 10^6$ equivalent expressions for $n = 5$

$1.3 \cdot 10^9$ equivalent expressions for $n = 6$

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: `int` $= [\mathbf{m}, \mathbf{M}]$ with $\mathbf{m} = -2^{31}$ and $\mathbf{M} = 2^{31} - 1$

Operations (wrap up)

$$\mathbf{M} + 1 = \mathbf{m} \qquad \mathbf{m} - 1 = \mathbf{M}$$

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

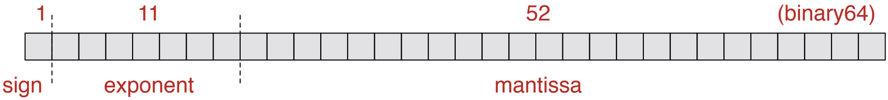
And which minimizes the maximal intermediary result in absolute value

[F. Logozzo and T. Ball. *Modular and verified automatic program repair*. OOPSLA'12]

Floating-Point Arithmetic: the IEEE754 Standard

Binary64 normalized floating-point numbers: $\pm \underbrace{1.d_1 d_2 \ldots d_p}_{\textit{mantissa}} 2^e$

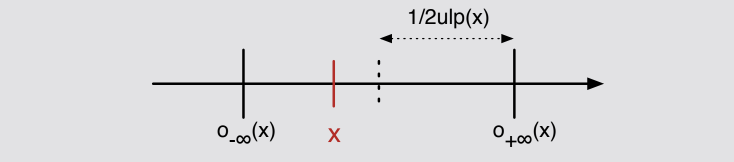
Precision $p = 52$, $-1022 \leq e \leq 1023$



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 $\odot \in \{+, -, \times, \div, \sqrt{}\}$:

$$x \circledast_r y = \circ_r(x * y)$$

[J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2010]

Floating-Point Arithmetic

Example

$e = 2.7182818\ldots$ computed using Bernouilli's formula:

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

In double precision

$$u_8 = 2.718282 \quad u_{14} = 2.716110 \quad u_{16} = 3.0.35035 \quad u_{17} = 1.0$$

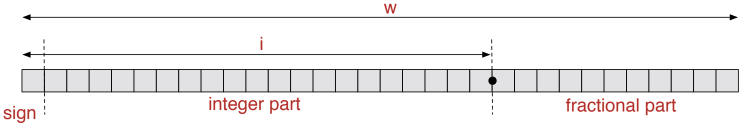
Synthesis of expressions

Generate expression mathematically equal to the original

And which minimizes the roundoff error $|r_{exact} - r_{float}|$ on the result

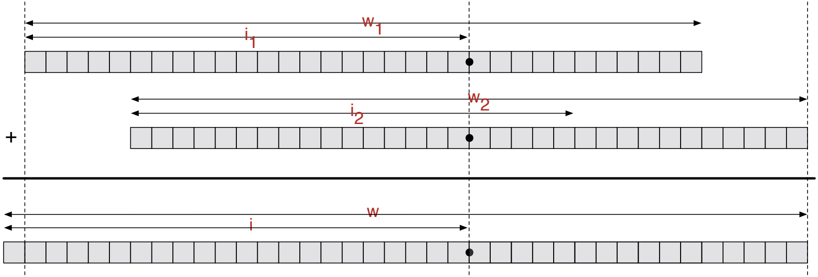
Fixed-Point Arithmetic

Values



In format $\langle w, i \rangle$, $b_{w-1} \ldots b_0$ reprensents: $-b_{w-1} \cdot 2^{i-1} + \sum_{j=2}^{j=w} b_{w-j} \cdot 2^{i-j}$

Operations



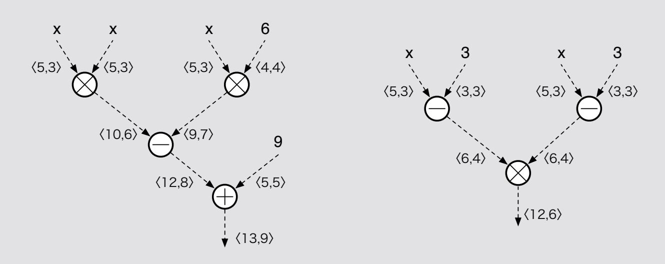
Fixed-Point Arithmetic

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

$$[\underline{x}, \overline{x}] \boxplus [\underline{y}, \overline{y}] = [\underline{x} \oplus_{-\infty} \underline{y}, \overline{x} \oplus_{+\infty} \overline{y}]$$

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

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

$$\texttt{int} \times \mathbb{Z} \qquad \texttt{float} \times \mathbb{R} \qquad \texttt{fixed} \times \mathbb{R} \qquad \texttt{float} \times \texttt{float} \times \wp(\mathbb{R})$$

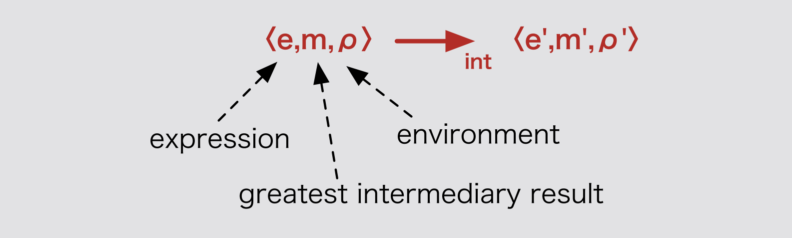
Non-standard semantics

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

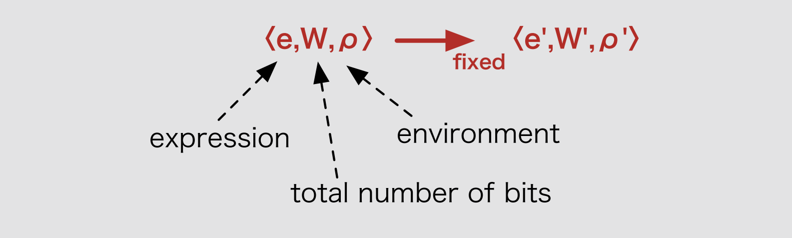
Integer Non-Standard Semantics



$$\frac{\hat{v} = \hat{v}_1 \circledast \hat{v}_2 \quad v = v_1 * v_2 \quad m' = \max(|v_1|, |v_2|, |v|, m)}{\langle (\hat{v}_1, v_1) * (\hat{v}_2, v_2), m, \rho \rangle \rightarrow_{\texttt{int}} \langle (\hat{v}, v), m', \rho \rangle}$$

Objective: find a trace which minimizes m

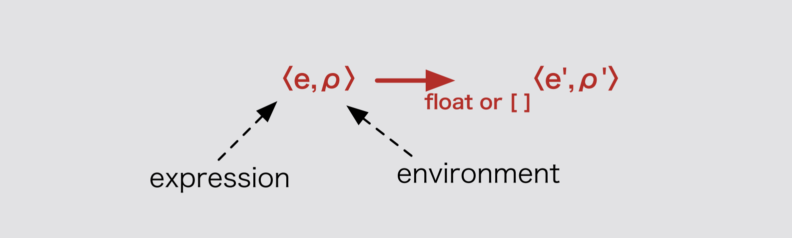
Fixed-Point Non-Standard Semantics



$$\frac{v^{\langle w,i\rangle} = v_1^{\langle w_1,i_1\rangle} \circledast v_2^{\langle w_2,i_2\rangle} \quad v = v_1 * v_2 \quad W' = W + w_1 + w_2 + w}{\langle (v_1^{\langle w_1,i_1\rangle}, v_1) * (v_2^{\langle w_2,i_2\rangle}, v_2), W, \rho\rangle \to_{\texttt{fixed}} \langle (v^{\langle w,i\rangle}, v), W', \rho\rangle}$$

Objective: find a trace which minimizes W

Floating-point and Interval Non-Standard Semantics



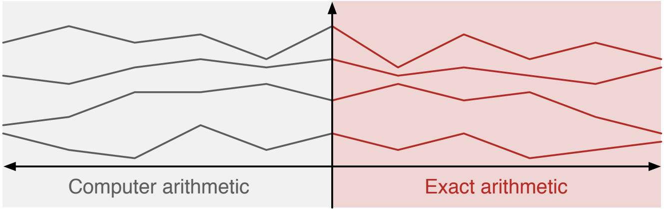
$$\text{float: } \frac{\hat{v} = \hat{v}_1 \circledast_{\sim} \hat{v}_2 \quad v = v_1 * v_2}{\langle (\hat{v}_1, v_1) * (\hat{v}_2, v_2), \rho \rangle \rightarrow_{\texttt{float}} \langle (\hat{v}, v), \rho \rangle}$$

$$\text{interval: } \frac{\hat{v} = v_1 \boxast \hat{v}_2 \quad v = \{ x * y \ : \ x \in v_1,\ y \in v_2 \}}{\langle (\hat{v}_1, v_1) * (\hat{v}_2, v_2), \rho \rangle \rightarrow_{[]} \langle (\hat{v}, v), \rho \rangle}$$ v set of points

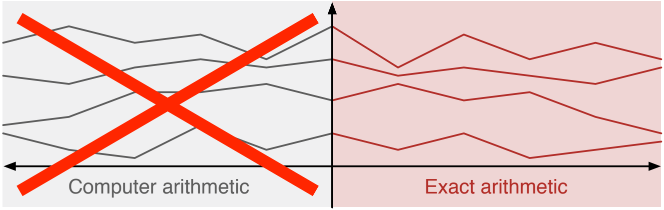
Objective: find a trace which minimizes $|\hat{v} - v|$ or $\mathit{width}(\hat{v})$

Non-Standard Semantics: Correctness (1/2)

Collecting Semantics $[\![e]\!]\,\Theta$ for set Θ of environments



Observational abstractions $\alpha_{\mathcal{O}}$

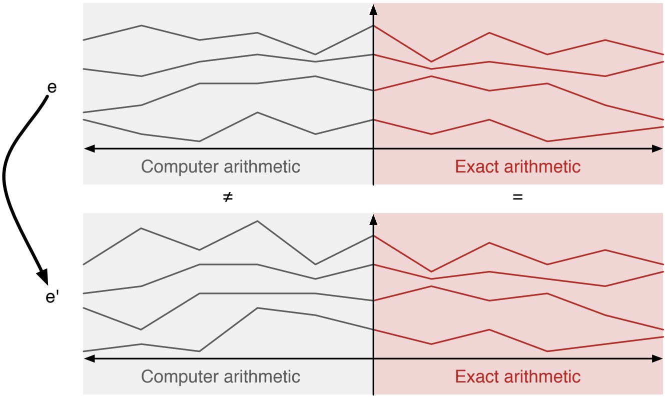


Keeps exact results, discards computer results in non-standard values

Non-Standard Semantics: Correctness (2/2)

It is correct to replace e by e' if for any Θ, $[\![e]\!]\ \Theta = S$, $[\![e']\!]\ \Theta = S'$ and:

$$\{\alpha_{\mathcal{O}}(s)\ :\ s \in S\} = \{\alpha_{\mathcal{O}}(s')\ :\ s' \in S'\}$$



[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 ($\frac{2n!}{n!(n+1)!}$ parsings)

$$\underbrace{e = (x-1) \times \ldots \times (x-1)}_{n \text{ times}}$$

$2.3 \cdot 10^6$ equivalent expressions for $n = 5$

$1.3 \cdot 10^9$ equivalent expressions for $n = 6$

Two abstractions:

EUD-k: Identify expressions whose syntactic trees are Equal Up to Depth k

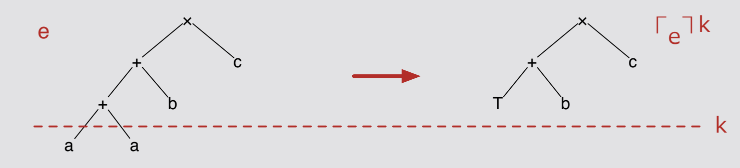
[M. Martel, *Semantics-based transformation of arithmetic expressions*, SAS'07]

APEGs: Abstraction based on Abstract Program Expression Graphs

[A. Ioualalen and M. Martel, *A new abstract domain for the representation of mathematically equivalent expressions*, SAS'12]

EUD-k Abstraction (1/3)

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:

$$\{(e_1 + (e_2 + e_3), (e_1 + e_2) + e_3) \ : \ e_1,\ e_2,\ e_3 \in \text{Expr}\} \subseteq \mathcal{R}$$

$$\{(e_1 \times (e_2 + e_3), e_1 \times e_2 + e_1 \times e_3) \ : \ e_1,\ e_2,\ e_3 \in \text{Expr}\} \subseteq \mathcal{R}$$

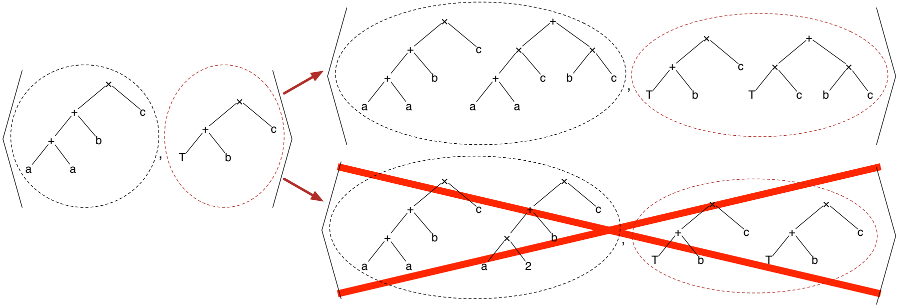
EUD-k Abstraction (2/3)

Generation of a set of equivalent expressions

$\rightarrow_k$ on states $\langle E, K\rangle \in \wp(Expr) \times \wp(Expr)$

$$\frac{e \in E \quad e \,\mathcal{R}\, e' \quad \ulcorner e' \urcorner^k \notin K}{\langle E, K\rangle \rightarrow_k \langle \{e'\} \cup E, \{\ulcorner e' \urcorner^k\} \cup K\rangle}.$$

Initial state $\langle \{e\}, \{\ulcorner e \urcorner^k\}\rangle$



EUD-k Abstraction (3/3)

Compute maximal set E of equivalent expressions such that

$$e_1, e_2 \in E \;\Rightarrow\; \ulcorner e_1 \urcorner^k \neq \ulcorner e_2 \urcorner^k$$

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

Exponential in k (user-defined parameter)

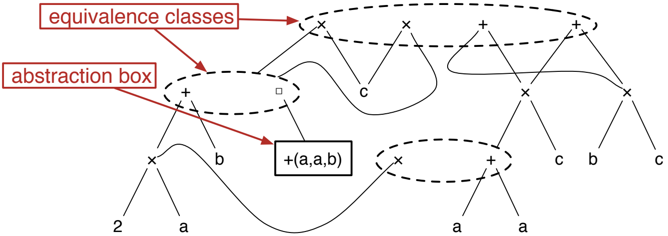
Example

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

$$S_1 = \big\{c \times \big((a + a) + b\big),\; c \times (a + a) + c \times b\big\} \quad \text{if } k = 1,$$

$$S_2 = \left\{ \begin{array}{c} ((a + a) + b) \times c,\; (a + (a + b)) \times c, \\ (a + a) \times c + b \times c,\; a \times c + (a + b) \times c \end{array} \right\} \quad \text{if } k = 2.$$

Abstract Program Expression Graphs (1/2)



Represent many equivalent expressions in polynomial size

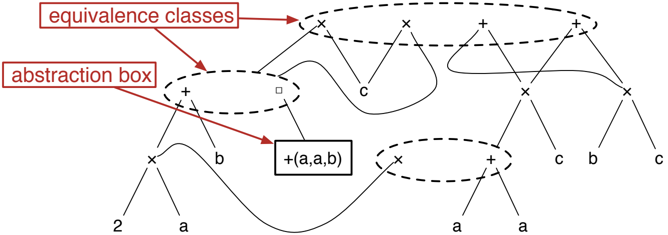
APEGs contain equivalence classes

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

$\boxed{*(e_1, \ldots, e_n)}$ represents all the parsings of $e_1 * \ldots * e_n$

Box elements are constants, expressions or abstraction boxes

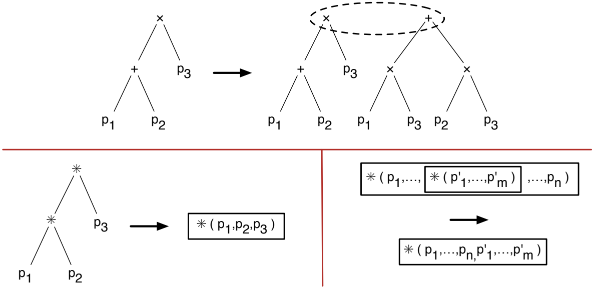
Abstract Program Expression Graphs (2/2)



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

$$\mathcal{A}(p) = \left\{ \begin{array}{c} ((a+a)+b)\times c,\ ((a+b)+a)\times c,\ ((b+a)+a)\times c, \\ ((2\times a)+b)\times c,\ c\times((a+a)+b),\ c\times((a+b)+a), \\ c\times((b+a)+a),\ c\times((2\times a)+b),\ (a+a)\times c+b\times c, \\ (2\times a)\times c+b\times c,\ b\times c+(a+a)\times c,\ b\times c+(2\times a)\times c \end{array} \right\}$$

APEG Construction



APEG construction

Rewritting rules applyed 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 contains only equivalent expressions

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

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

EUD-k: $\langle \{e\}, \{\ulcorner e \urcorner^k\} \rangle \rightarrow_k^* \langle E^\sharp, K \rangle$

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

$E^\sharp$ and $\mathcal{A}(p)$ are under-approximations of $\mathcal{E}(e)$

Correctness (2/2)

concrete set of equivalent expressions

EUD-k states: expressions × expressions of limited depth

$$\langle \wp(Expr), \subseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \wp(Expr) \times \wp(Expr), \subseteq_\times \rangle$$

$$\langle \wp(Expr), \subseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \Pi, \subseteq_\Pi \rangle$$

concrete set of equivalent expressions

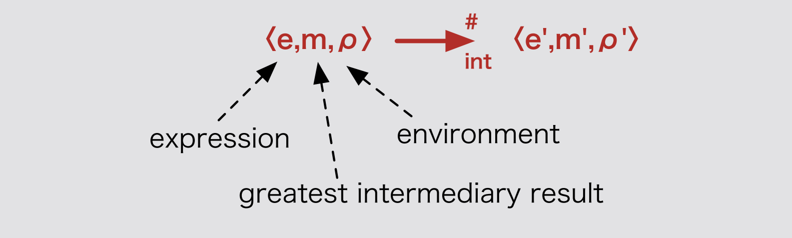
set of APEGs

$$\gamma_1(\langle E, K \rangle) = \bigcup_{e \in E,\ \{e\} \to E'} E' \qquad \gamma_2(p) = \bigcup_{e \in \mathcal{A}(p),\ \{e\} \to E'} E'.$$

Summary

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

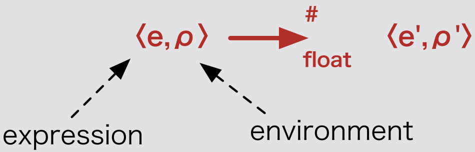
Integer Abstract Semantics



Abstract value: interval of integers ($\texttt{int} \times \texttt{int}$)

$$\frac{[\underline{v}, \overline{v}] = [\underline{v_1}, \overline{v_1}] \boxast_{\texttt{int}} [\underline{v_2}, \overline{v_2}] \quad m' = \max(|\underline{v_1}|, |\overline{v_1}|, |\underline{v_2}|, |\overline{v_2}|, |\underline{v}|, |\overline{v}|, m)}{\langle [\underline{v_1}, \overline{v_1}] * [\underline{v_2}, \overline{v_2}], m, \rho \rangle \rightarrow^{\sharp}_{\texttt{int}} \langle [\underline{v}, \overline{v}], m', \rho \rangle}$$

Floating-Point Abstract Semantics



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.)

$$\frac{[\underline{\hat{v}}, \overline{\hat{v}}] = [\underline{\hat{v}_1}, \overline{\hat{v}_1}] \boxed{\ast}_{\sim} [\underline{\hat{v}_2}, \overline{\hat{v}_2}] \quad [\underline{v}, \overline{v}] = [\underline{v_1}, \overline{v_1}] \boxed{\ast}_{\downarrow} [\underline{v_2}, \overline{v_2}]}{\langle([\underline{\hat{v}_1}, \overline{\hat{v}_1}],[\underline{v_1},\overline{v_1}]) * ([\underline{\hat{v}_2}, \overline{\hat{v}_2}],[\underline{v_2},\overline{v_2}]), \rho\rangle \rightarrow^{\sharp}_{\mathtt{float}} \langle([\underline{\hat{v}}, \overline{\hat{v}}],[\underline{v},\overline{v}]),\rho\rangle}$$

Generation of new Expressions

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

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 $error(a * b)$ is minimal

Complexity: $O(n^2)$

Example:

$$\boxed{+(a,b,c,d,e)} \longrightarrow \boxed{+(a,c,e,\boxed{+(b,d)})} \longrightarrow \boxed{+(e,\boxed{+(a,c)},\boxed{+(b,d)})} \longrightarrow \boxed{+(\boxed{+(e,\boxed{+(a,c)})},\boxed{+(b,d)})}$$

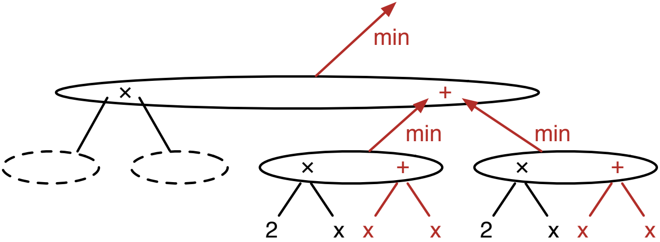
We synthetize $(e + (a + c)) + (b + d)$

APEGs and Formula Synthesis: Equivalence Classes

Simplest approach:

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

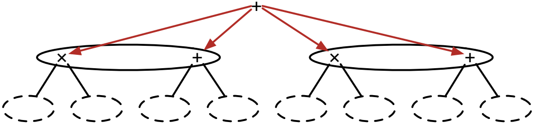
Complexity: $O(n)$



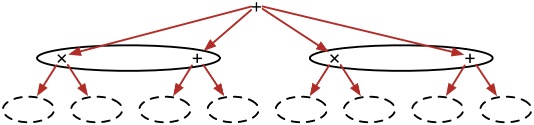
APEGs and Formula Synthesis: Improvement

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 ≈ 1

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

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

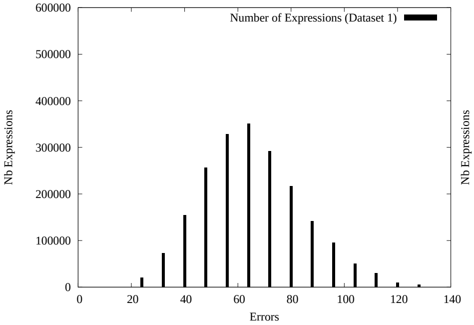
2 interval widths:

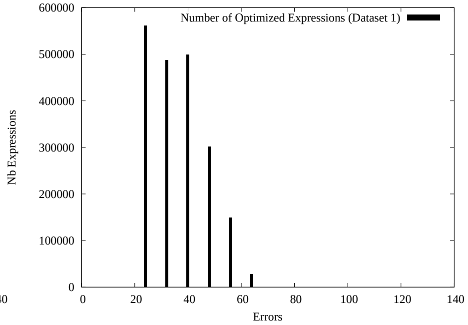
Width $=$ 10% of the central value of the interval

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 $\approx 10^{-16}$

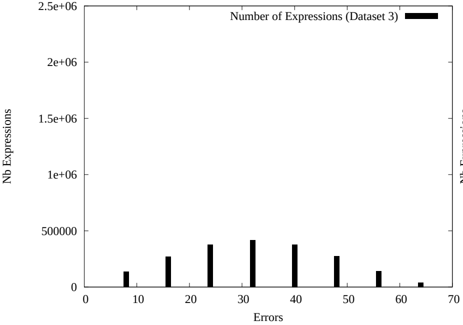


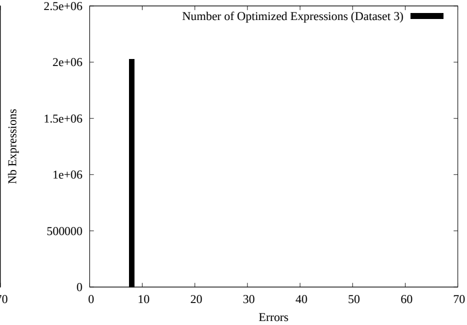


Large intervals. (similar results for small intervals)

Summation of 9 Termes : 2 Millions Cases (Dataset 2)

> 0, 20% of large values among small and medium values ≈ 1

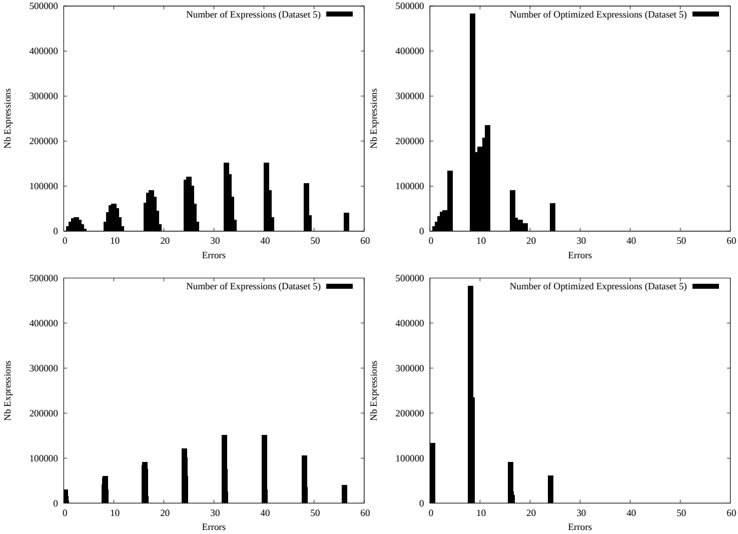




Large intervals. (similar results for small intervals)

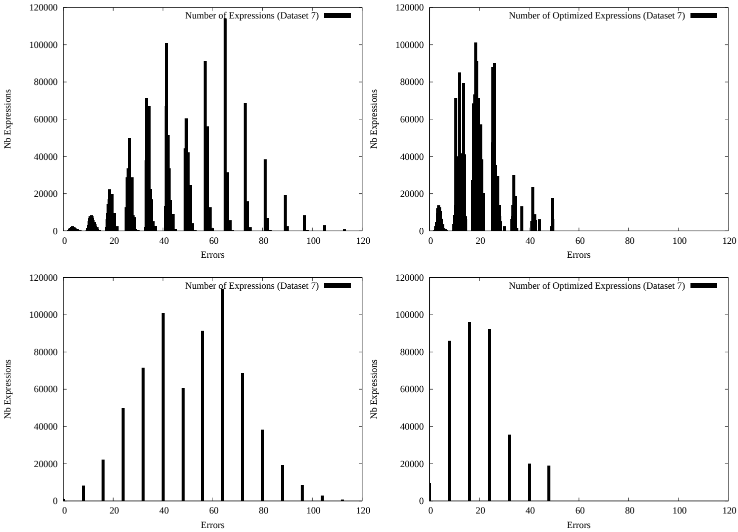
Summation of 9 Termes : 2 Millions Cases (Dataset 3)

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



Developed Polynomials

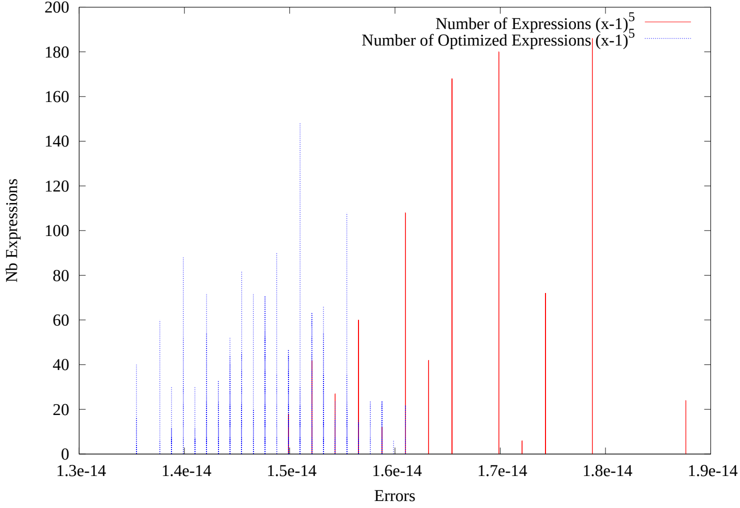
We consider the polynomial:

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

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

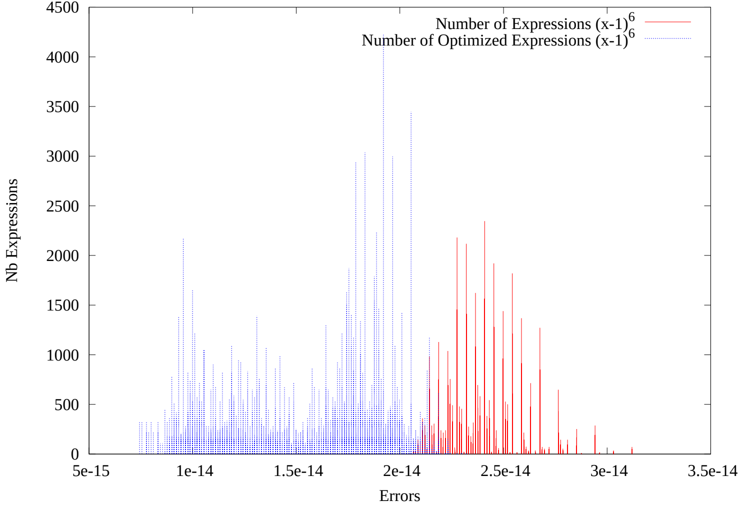
Source expressions: all the parsings of (1), no factorization

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

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

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

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

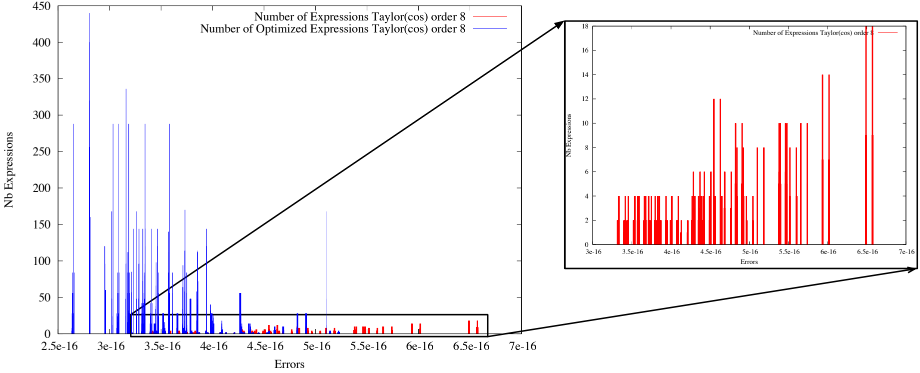
Development orders for cos: $n \in \{4, 6, 8\}$

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

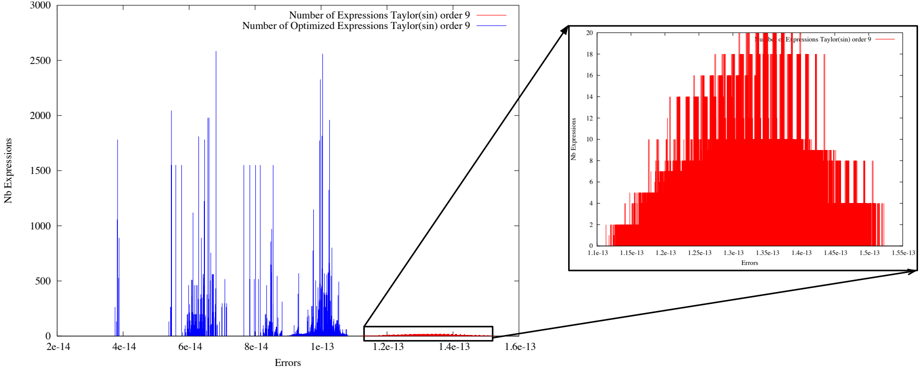
Result for cos with $n = 8$, 30240 Cases



Red: initial error bounds

Blue: error bounds on synthetized expressions

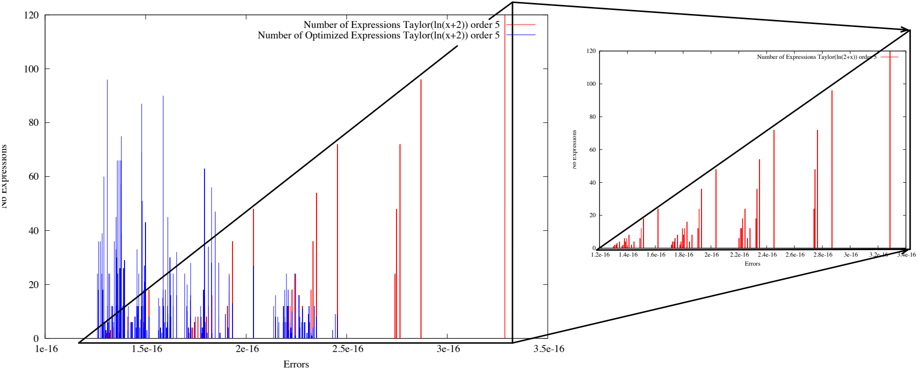
Result for cos with $n = 9$, 162855 Cases



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

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

8 bits image: value of pixels between 0 and 255

Fixed-Point Arithmetic (2/2)

	size 3x3				size 5x5		
Filter	$Filter_{sum}$	$Filter_{sum}^{opt}$	%*Gain*	Avg-IP	$Filter_{sum}$	$Filter_{sum}^{opt}$	%*Gain*
Gaussian	183	176	3,8%	$\sim$ 139	442	393	11%
Laplacian 1	204	180	11,7%	$\sim$ 152	713	660	7,4%
Laplacian 2	248	246	0,8%	$\sim$ 204	779	723	7,1%
Prewit 1	178	163	8,4%	$\sim$ 133	638	569	10,8%
Prewit 2	184	169	8,1%	$\sim$ 136	644	571	11,3%
Rehauss 1	259	253	2,3%	$\sim$ 212	841	794	5,5%
Rehauss 2	249	242	2,8%	$\sim$ 201	724	678	6,3%
Robert 1	263	233	11,4%	$\sim$ 161	778	694	10,7%
Robert 2	266	233	12,4%	$\sim$ 153	781	694	11,1%
Sobel 1	198	176	11,1%	$\sim$ 145	769	678	11,8%
Sobel 2	194	176	9,2%	$\sim$ 144	752	678	10,6%

Fractional part size	Error bound generated
4	$1.2 \cdot 10^2$
6	$2.8 \cdot 10^1$
8	5.1
10	1.9
12	$3.4 \cdot 10^{-1}$
14	$3.1 \cdot 10^{-2}$
16	0.0

[A. Ioualalen and M. Martel, *Synthesis of Arithmetic Expressions for the Fixed-Point Arithmetic: The Sardana Approach*, DASIP'2012]

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: `tmp = a-(a+b); err = tmp+b`

Tool under development: SARDANA

Open PhD position

Questions?