YalAA - Yet Another Library for Affine Arithmetic

Stefan Kiel
University of Duisburg-Essen

June 14, 2011

Table of Contents

Affine Arithmetic

- Arithmetic for verified numerics [Comba and Stolfi, 1990]
- Linear dependency tracking
- Partially unknown quantity as the affine form $\hat{x}$
  $\hat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \ldots + x_n\epsilon_n$, where

x_0	Central value
x_i $i \geq 1$	Partial deviations x_i (errors or uncertainties)
ϵ_i, $i \geq 1$	(Symbolic) noise symbols $\epsilon_i \in [-1, 1]$

Operations in Affine Arithmetic

Consider two affine forms
$\hat{x} = x_0 + x_1\epsilon_1 + ... + x_n\epsilon_n$ and $\hat{y} = y_0 + y_1\epsilon_1 + ... + y_n\epsilon_n$

Affine operations

$$\begin{aligned}
\hat{x} + \hat{y} &= (x_0 + y_0) + (x_1 + y_1)\epsilon_1 + ... + (x_n + y_n)\epsilon_n \\
\alpha\hat{x} &= (\alpha x_0) + (\alpha x_1)\epsilon_1 + ... + (\alpha x_n)\epsilon_n \\
\alpha + \hat{x} &= (x_0 + \alpha) + x_1\epsilon_1 + ... + x_n\epsilon_n
\end{aligned}$$

Non affine operation f

- Pick a good affine approximation f^a to f over its domain
- Introduce a new noise symbol ϵ_{n+1} and a add an extra error term $x_{n+1}\epsilon_{n+1}$ enclosing the approximation error.

Improvements

Several improvements to the original model have been proposed

AF1, AF2 [Messine, 2002]
Reduce number of noise symbols

QFT, ... [Messine and Touhami, 2006], [Bilotta, 2008], [Shou et al., 2003]
Preserve higher order correlations

Posteriori error corrections [Jordan Ninin, 2011]
Use COSY-like approach for error correction

Implementations: State of the Art

Two publicly available implementations

libaa [Stolfi]

Implements the standard model

Not object oriented (C)

Limited set of elementary functions

libaffa [Gay et al.]

Implements the standard model

Object oriented (C++)

More complete set of elementary functions

Not always verified

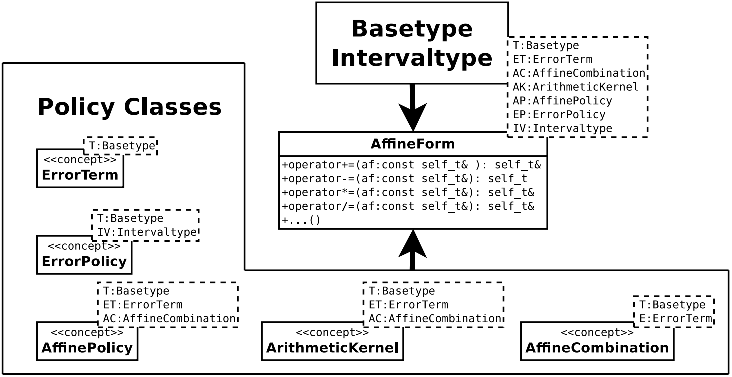
Our goal:

A library supporting both standard and the extended affine models.

YalAA

Goal	Realization
Object oriented interface	C++
Most elementary functions	Functions from P1788 draft
Several affine models	Policy based design
Custom base type	Specialization of operations
Integration with IA libraries	Trait classes Basic combinations with intervals
Verified implementation	*Extended Stolfi model*

Basic Structure



The main class `AffineForm` is customized through 2 types and 5 polices.

ErrorTerm and AffineCombination

```
                                         T:Basetype
<<concept>>
ErrorTerm
+ErrorTerm(base_t:dev)
+<<const>> operator==(other:const ET<T>&): bool
+<<const>> operator<(other:const ET<T>&): bool
+<<const>> dev(): T
+set_dev(new_dev:base_t): void
+...()
```

Tasks

- Models error terms $x_i \epsilon_i$
- Provides new noise variables
- Defines ordering on symbols

Why policy?

- Thread safe, not thread safe
- Custom maximum number of ϵ_i

```
                                   T:Basetype
                                   ET:ErrorTerm<T>
<<concept>>
AffineCombination
+AffineComb(central:base_t)
+<<const>> central(): base_t
+set_central(new_central:base_t): void
+insert(term:const ET<T>&): void
+remove(term:const ET<T>&): void
+begin(): aff_comb_iter
+end(): aff_comb_iter
+<<const>> size(): size_t
+...()
```

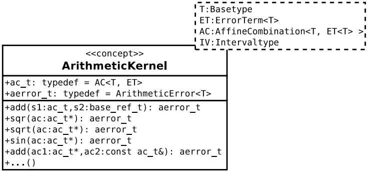
Tasks

Models $x_0 + x_1\epsilon_1 + \ldots x_n\epsilon_n$

Why Policy?

Different data structures for storage

ArithmeticKernel



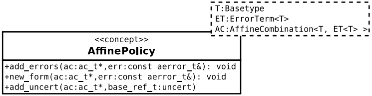
Tasks

- Implements arithmetic operations
- Carries out the affine part of an operation
- Returns rounding/approx. errors
- Returns status flags

Why policy?

- Specialization for every base type
- Different implementations

AffinePolicy and ErrorPolicy

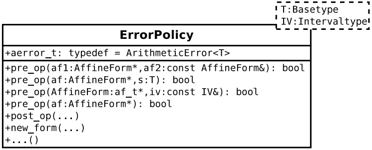


Tasks

- Handles rounding/approx. errors
- Introduces new forms
- Introduces extra uncertainty

Why policy?

Realization of AF1, AF2 forms



Tasks

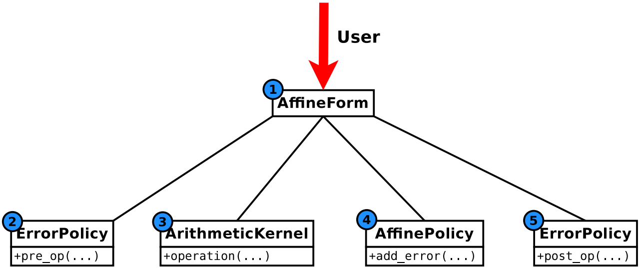
- Handles special values $\rightarrow \pm\infty, \emptyset$
- Handles errors during calculation

Why policy?

Allows different approaches

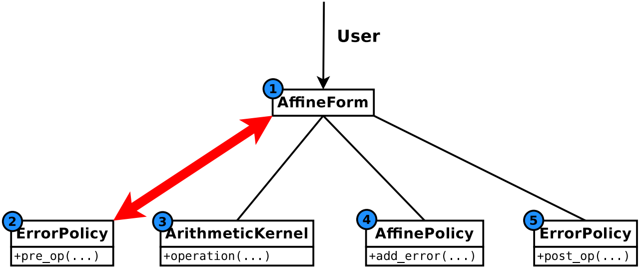
e.g. *Stolfi's approach, decorations* ...

Interaction of Policies



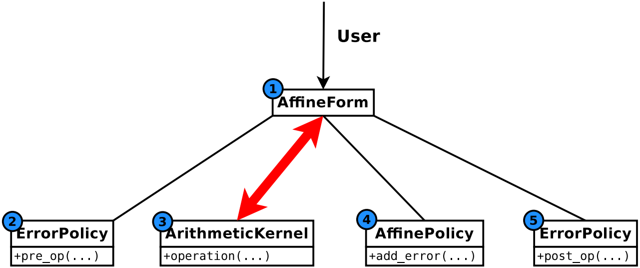
1. Call an operation
2. Check for special forms in the input arguments → `ErrorPolicy`
3. Perform actual operation → `ArithmeticKernel`
4. Add approximation/rounding errors to result → `AffinePolicy`
5. Check for errors during operation → `ErrorPolicy`

Interaction of Policies



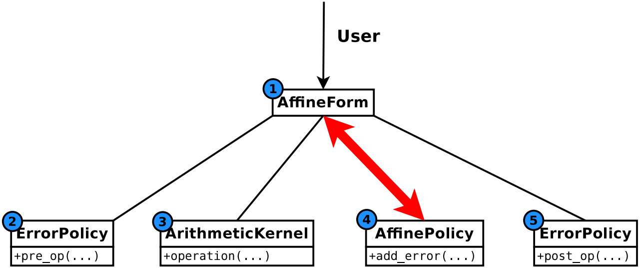
1. Call an operation
2. Check for special forms in the input arguments → `ErrorPolicy`
3. Perform actual operation → `ArithmeticKernel`
4. Add approximation/rounding errors to result → `AffinePolicy`
5. Check for errors during operation → `ErrorPolicy`

Interaction of Policies



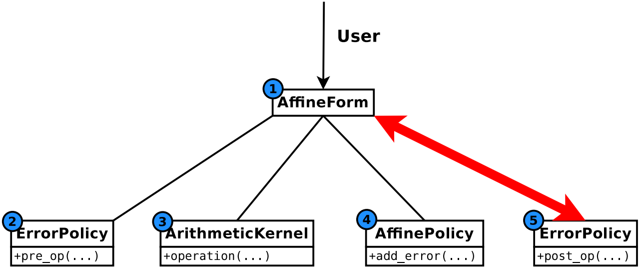
1. Call an operation
2. Check for special forms in the input arguments → `ErrorPolicy`
3. Perform actual operation → `ArithmeticKernel`
4. Add approximation/rounding errors to result → `AffinePolicy`
5. Check for errors during operation → `ErrorPolicy`

Interaction of Policies



1. Call an operation
2. Check for special forms in the input arguments → `ErrorPolicy`
3. Perform actual operation → `ArithmeticKernel`
4. Add approximation/rounding errors to result → `AffinePolicy`
5. Check for errors during operation → `ErrorPolicy`

Interaction of Policies



1. Call an operation
2. Check for special forms in the input arguments → `ErrorPolicy`
3. Perform actual operation → `ArithmeticKernel`
4. Add approximation/rounding errors to result → `AffinePolicy`
5. Check for errors during operation → `ErrorPolicy`

Signed Errors

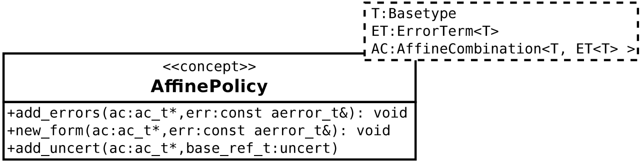
Approximation errors are possibly positive or negative
The sign is lost, if we simply map them to an error symbol.

Example

Let $\hat{x} = x_0 + x_1\epsilon_1$.
Then $(\hat{x})^2 = x_0^2 + 2x_0x_1\epsilon_1 + \underbrace{x_1^2\epsilon_1^2}_{(1)}$

The term (1) is non affine and to be replaced by new error symbol.
However, $\epsilon_1^2 \in [0, 1] \rightarrow$ that is the error is positively signed error.

Affine Policy in Detail



Three operations

`add_errors` Adds rounding or approximation errors to the affine form

`add_uncert` Introduces extra uncertainty (combination with intervals)

`new_form` Generates a new affine form

$\rightarrow$ `AffinePolicy` enables us to implement the AF0, AF1 and AF2.

Common Affine Policy

Idea

Map all errors *in each* computation step into a new symbol

In YalAA

Every `AffinePolicy` operation introduces a new ϵ_{n+1}.

Signed errors

Let e^+ the positive error.
Then $x_{n+1} = 0.5e^+$ and scale $x_0 = x_0 + 0.5x_0$
For negative analogously.

AF1 [Messine, 2002]

Problem

The number of noise symbols grows due to rounding/approximation errors.

Idea

Introduce a noise symbol for every input variable

Map *all* errors into the same noise symbol ϵ_e

$\rightarrow \hat{x} = x_o + \left(\sum_{i=0}^{n} x_i \epsilon_i \right) + x_e \epsilon_e$

Policy

`add_errors` $x_e = x_e + |e|$

`add_uncert` $x_e = x_e + |u|$

`new_form` Introduces new variable ϵ_{n+1}

Signed errors

Like in common policy

AF2 [Messine, 2002]

Problem

Some error are either positive or negative

Error terms in standard affine arithmetic always lie in $[-1,1]$

Idea

Extension of AF1

Split $x_e\epsilon_e$ accounting for positive, negative and general errors

$$\rightarrow \hat{x} = x_o + \left(\sum_{i=0}^{n} x_i\epsilon_i\right) + x_{e^+}\epsilon_{e^+} + x_{e^-}\epsilon_{e^-} + x_e\epsilon_e$$

with $\epsilon_{e^+} \in [0,1], \epsilon_{e^-} \in [-1,0]$

Policy

`add_errors` $x_e = x_e + |e|, x_{e^+} = x_{e^+} + |e^+|, \ldots$

`add_uncert` $x_e = x_e + |u|$

`new_form` Introduces new variable ϵ_{n+1}

Standard Error Policy

Common error handling strategy [de Figueiredo and Stolfi, 1997]

Introduces two special forms

R Complete real line

$\emptyset$ Empty affine form

Combination of special forms.

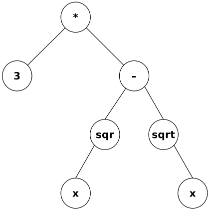
$\circ$	NONE	R	EMPTY
NONE	NONE	R	EMPTY
R	R	R	EMPTY
EMPTY	EMPTY	EMPTY	EMPTY

Interval Decorations

Decorations [Hayes and Neumaier], [Hayes, 2010], [Kreinovich, 2011]

Concept currently discussed by the P1788 interval standardization group
Store an interval and a decoration as a pair $\rightarrow$ *decorated interval*
Propagation of error states and features of an *inductively defined function*

Function



$f(x) = 3(sqr(x) - sqrt(x))$

Combination of elementary functions
Propagate decorations through graph
Decoration of an operation depends on

- Operands
- Operation performed
- Decoration of operands

$\rightarrow$ Statements about inductively defined functions

Properties

Domain tetrit [Hayes, 2010]
Let $D_f \subseteq \mathbb{R}^n$ the natural domain of $f : \mathbb{R}^n \to \mathbb{R}$. Then the domain tetrit $D(f, \boldsymbol{x}) = (D^+, D^-)$ for an interval $\boldsymbol{x} \in \mathbb{IR}^n$ is defined as

$$D^+ \Leftrightarrow (\exists x \in \boldsymbol{x}) : \boldsymbol{x} \in D_f$$
$$D^- \Leftrightarrow (\exists x \in \boldsymbol{x}) : \neg(\boldsymbol{x} \in D_f)$$

Defined and Continuous [Hayes, 2011]
$C(f, \boldsymbol{x})$: the restriction of f to $\boldsymbol{x}$ is defined and continuous

Defined and Bounded [Hayes, 2010]
$B(f, \boldsymbol{x})$: the restriction of f to $\boldsymbol{x}$ is defined and bounded

Decorations in YalAA

Get for the interval box $\boldsymbol{x}$ enclosing an affine form $\hat{x}$
Keeping track of the properties $D(f, \boldsymbol{x}), B(f, \boldsymbol{x}), C(f, \boldsymbol{x})$

This results in 7 decorations

D^+	D^-	C	B	Dec.	Meaning
T	F	T	T	$\mathbb{D}_5$	f is cert. defined, cont. and bounded over $\boldsymbol{x}$
T	F	T	F	$\mathbb{D}_4$	f is cert. defined and cont. over $\boldsymbol{x}$
T	F	F	F	$\mathbb{D}_3$	f is cert. defined over $\boldsymbol{x}$
T	T	F	F	$\mathbb{D}_2$	f is possibly defined over $\boldsymbol{x}$
F	T	F	F	$\mathbb{D}_1$	f is certainly undefined over $\boldsymbol{x}$
F	F	F	F	$\mathbb{D}_0$	$\boldsymbol{x}$ is the empty set
?	?	?	?	$\mathbb{D}_{-1}$	an error occurred

The decorations are ordered, that is, $\mathbb{D}_i < \mathbb{D}_{i-1}$ for $5 \geq i \geq 0$
Decoration of $f(x)$ is the minimum of $f's$ and its arguments decorations.

Interaction with the Affine Part

Decoration	Meaning for the affine part	Central value[1]
$\mathbb{D}_5$	Has result	*valid*
$\mathbb{D}_4$	Overflowed	$\pm\infty$
$\mathbb{D}_3$	Possibly has result	*valid* or $\pm\infty$
$\mathbb{D}_2$	Possibly has result	*valid* or $\pm\infty$
$\mathbb{D}_1$	Is empty set	NaN
$\mathbb{D}_0$	Is empty set	NaN
$\mathbb{D}_{-1}$	Undefined	NaN

Problems

No distinction for overflows or unbounded

Possible overflow in $\mathbb{D}_3$ and $\mathbb{D}_2$

Using NaN for empty sets and undefined is not optimal.

[1] in case of IEEE754 base type

Interaction with Intervals and Scalars

Affine forms can be combined with intervals and scalars.
Manual conversion is possible

Rules for automatic conversion

Type	Value	Dec.
Scalar	$\neg$`special`	$\mathbb{D}_5$
Scalar	`special`	$\mathbb{D}_{-1}$
Interval	$\neg$(`empty` $\vee$ `special`)	$\mathbb{D}_5$
Interval	`empty` $\wedge \neg$`special`	$\mathbb{D}_0$
Interval	`special`	$\mathbb{D}_{-1}$

Remarks

With IEEE754 base types: `special` is NaN or $\pm\infty$
Affine decorations are valid for the box enclosure of the affine form
$\rightarrow$ conversion to decorated intervals (currently not supported)

Conclusion and Outlook

Conclusion

- New library for affine arithmetic
- Extended set of elementary functions
- Decoration support
- Tries to integrate different models for affine computation

Intended features

- Other error correction modes (COSY like)
- Generalized interval arithmetic [Hansen, 1975]
- Higher order forms

Thank You for Your Attentation

Download available soon

http://www.scg.inf.uni-due.de

Bibliography

G. Bilotta. Self-verified extension of affine arithmetic to arbitrary order. *Le Matematiche*, 63(1):15–30, 2008.

J.L.D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)*, pages 9–18. Citeseer, 1990.

L.H. de Figueiredo and J. Stolfi. *Self-Validated Numerical Methods and Applications*. IMPA, Rio de Janeiro, 1997.

O. Gay, D. Coeurjolly, and N.J. Hurst. libaffa. `http://www.nongnu.org/libaffa/`, accessed on 15.11.2010.

E. Hansen. A generalized interval arithmetic. In Karl Nickel, editor, *Interval Mathematics*, volume 29 of *Lecture Notes in Computer Science*, pages 7–18. Springer Berlin / Heidelberg, 1975.

N. T. Hayes. Exception handling for interval arithmetic, 2010. P1788, Position Paper.

N. T. Hayes. Property tracking with decorations, May 2011. P1788, Proposed Motion.

N. T. Hayes and A. Neumaier. Exception handling for interval arithmetic. P1788, Motion 8.

Frédéric Messine Jordan Ninin. Reliable affine arithmetic, 2011. International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2010).

Vladik Kreinovich. How to check, while performing straightforward interval computations, whether the resulting function is continuous, everywhere defined, etc.: Towards foundations of decorations, 2011.

F. Messine. Extentions of affine arithmetic: Application to unconstrained global optimization. *Journal of Universal Computer Science*, 8(11):992–1015, 2002.

Frédéric Messine and Ahmed Touhami. A general reliable quadratic form: An extension of affine arithmetic. *Reliable Computing*, 12:171–192, 2006. ISSN 1385-3139. URL `http://dx.doi.org/10.1007/s11155-006-7217-4`. 10.1007/s11155-006-7217-4.

Huahao Shou, Hongwei Lin, Ralph Martin, and Guojin Wang. Modified affine arithmetic is more accurate than centered interval arithmetic or affine arithmetic. In Michael Wilson and Ralph Martin, editors, *Mathematics of Surfaces*, volume 2768 of *Lecture Notes in Computer Science*, pages 355–365. Springer Berlin / Heidelberg, 2003.

J. Stolfi. libaa. `http://www.ic.unicamp.br/~stolfi/`, accessed on 08.06.2011.