The SARDANES Project

Matthieu Martel

matthieu.martel@univ-perp.fr

ES_PASS Workshop, Toulouse, February 2009
Scientific Objectives of SARDANES

To compile safely SCADE numerical codes

To generate a program as close as possible of the specifications

To validate the translation

Techniques:

To optimize the numerical precision of the object code

To bound the errors introduced by the translation
SARDANES Summary

Funded by FNRAE (Fondation de Recherche pour l’Aéronautique et l’Espace)

Starts February 2009, lasts 36 months

**Partners:**

Ecole Normale Supérieure (P. and R. Cousot)

Université de Bretagne Occidentale (D. Massé)

Université de Perpignan Via Domitia (M. Martel)
Overview

Floating point arithmetic

Program transformation for numerical precision

Examples

Conclusion
**Floating point arithmetic:**

Not intuitive (usual algebraic laws do not hold)

Difficult to predict the precision of a computation

No method to improve the numerical accuracy of an implementation

A few empirical rules (Horner scheme, sort, etc.)
Example 1

\[ f: \mathbb{F} \rightarrow \mathbb{F} \]
\[ x \mapsto (1 - x) - 1 \]
Example 2

\[ f : \mathbb{F} \rightarrow \mathbb{F} \]
\[ x \mapsto (x - 2)^7 \]
Objective

Semantics-based transformation to improve the numerical precision

**SCADE semantics:**
- Computations are carried out using real numbers
- Infinite precision
- Algebraic laws (associativity, distributativity, etc.)

**SCADE implementation:**
- Object code uses floating point numbers
- Roundoff errors
- No algebraic laws
Overview

Floating point arithmetic

**Program transformation for numerical precision**

Examples

Conclusion
Principle of the Transformation: expressions

Allow algebraic rules in the semantics $\rightarrow$ many ways to evaluate the expressions

- $a + (b + c) = (a + b) + c$
- $a(b + c) = ab + ac$
- $a \times 1 = a$ etc.

Errors computed using abstract domains for numerical precision [Fluctuat]
Principle of the Transformation: programs

Allow to delay the evaluation of assignments → many ways to evaluate programs (either eager or lazy evaluation)

Also use standard unfolding techniques
Abstract Interpretation of the Traces

Concrete semantics: combinatorial explosion

Principle of the abstraction:
Overview

Floating point arithmetic

Program transformation for numerical precision

Examples

Conclusion
**Polynomials:** \((x \in [0,2], \text{ initial error on } x \in [0,0.0005])\)

<table>
<thead>
<tr>
<th>Case</th>
<th>Expression</th>
<th>Error bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source expression</td>
<td>(x+(x\times x))</td>
<td>([-1.800074334E-3,1.001074437E-3])</td>
</tr>
<tr>
<td>(k = 2)</td>
<td>((1.0+x)\times x)</td>
<td>([-9.000069921E-4,1.010078437E-4])</td>
</tr>
<tr>
<td>Source expression</td>
<td>((x\times (x\times x)) + (x\times x))</td>
<td>([-1.802887642E-3,3.191200091E-3])</td>
</tr>
<tr>
<td>(k = 3)</td>
<td>((x+1.0)\times (x\times x))</td>
<td>([-1.818142851E-4,1.390014781E-3])</td>
</tr>
<tr>
<td>(k = 4)</td>
<td>(((1.0+x)\times x)\times x)</td>
<td>([-9.091078216E-5,1.100112212E-3])</td>
</tr>
</tbody>
</table>

**Sums:**

\[
S = \sum_{i=0}^{4} x_i \quad \text{with} \quad x_i = [2^i, 2^{i+1}]
\]

<table>
<thead>
<tr>
<th>Case</th>
<th>Expression</th>
<th>Error bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source expression</td>
<td>(((e+d)+c)+b)+a)</td>
<td>([-7.6293E-6,7.6294E-6])</td>
</tr>
<tr>
<td>(k = 1)</td>
<td>((b+a)+(c+(e+d)))</td>
<td>([-5.9604E-6,5.9605E-6])</td>
</tr>
<tr>
<td>(k = 2)</td>
<td>((c+(b+a))+(e+d))</td>
<td>([-4.5299E-6,4.5300E-6])</td>
</tr>
<tr>
<td>(k = 3)</td>
<td>((d+(c+(a+b)))+e)</td>
<td>([-3.5762E-6,3.5763E-6])</td>
</tr>
</tbody>
</table>

\((a, b, c, d \text{ and } e \text{ stand for } x_0, x_1, x_2, x_3 \text{ and } x_4)\)
Example: loops

\[
S = \sum_{i=0}^{N} x_i \quad \text{with} \quad x_i = [z^{N-i}, z^{N-i+1}]
\]

i.e. \( x_N = [1,2] \), \( x_{N-1} = [2,4] \)...

Original program: [0.0, 2.861022950 \( E \) – 6]
Transformed program: [0.0, 2.145767212 \( E \) – 6]
Best bound (uses a sort): [0.0, 1.668930054 \( E \) – 6]

Possibility of Extension: Patriot Bug while(true) {t=t+0.1;}
Unfold 5 times and use 0.1+0.1+0.1+0.1+0.1 = 0.5
Example: conditionals

Automatic insertion of conditionals based on sensitivity analysis?
Specialization of programs w.r.t. some inputs to improve precision
Example: fixed point arithmetic

How many digits for the integer part, for a given implementation?

\[ e = (a + (b + (c + d))) \times e \]

\[ a = [-14, -13] \quad b = [-3, -2] \]
\[ c = [3, 3.5] \quad d = [12.5, 13.5] \quad e = 2 \]

\[ E'_\text{float} = ((a + b) \times e) + ((c + d) \times e) \]
\[ E'_\text{fixed} = e \times ((a + d) + (b + c)) \]

(can be extended to interval arithmetic)
Overview

Floating point arithmetic

Program transformation for numerical precision

Examples

Conclusion
SARDANES: Summary
Conclusion

**Current work (theory)**

- Definition of the transformation of full SCADE programs
- To improve the abstract semantics (global optimization)
- To select the relevant rules for the mathematical equivalence

**Current work (implementation)**

- Development of the tool

**Future work**

- Multi-criteria optimization (time)
References


- [http://gala.univ-perp.fr/~mmartel/sardanes.html](http://gala.univ-perp.fr/~mmartel/sardanes.html)
QUESTIONS?