The SARDANES Project

Matthieu Martel

matthieu.martel@univ-perp.fr

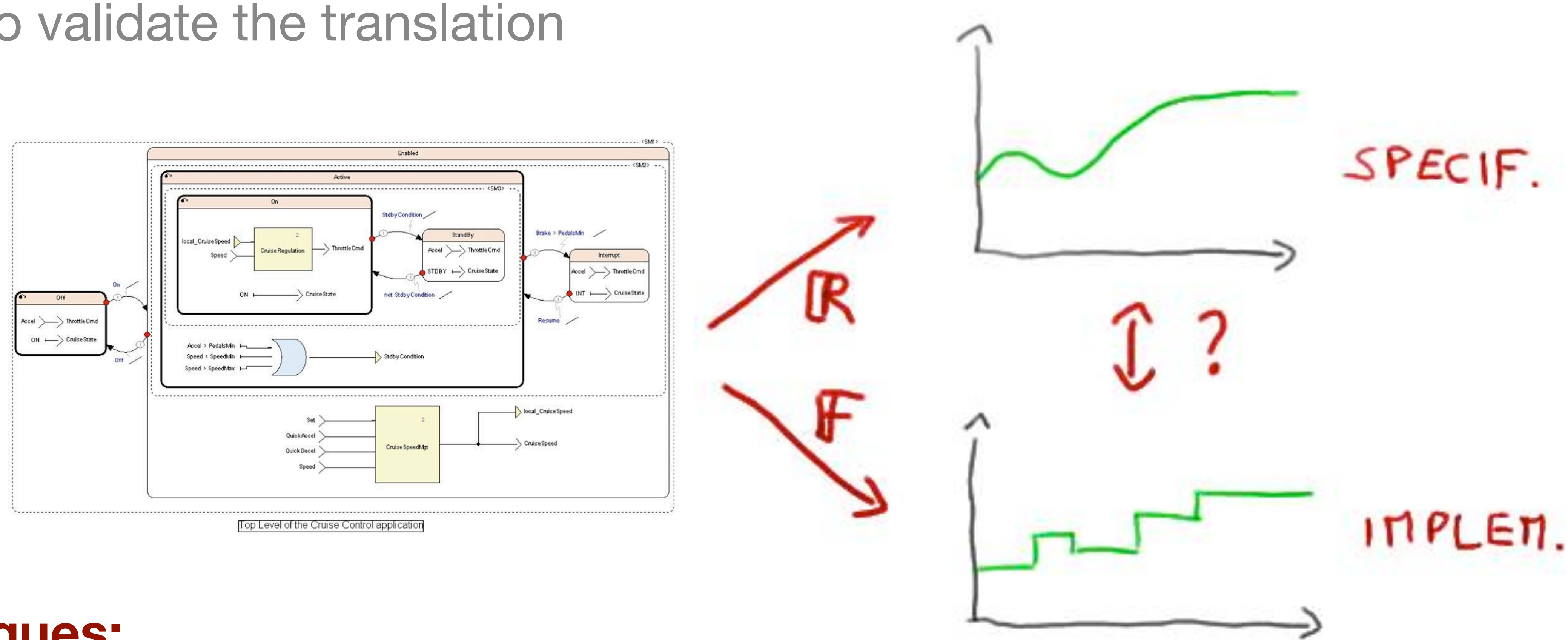
ES_PASS Workshop, Toulouse, February 2009

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)

Fondation de Recherche pour l' Aéronautique & l' Espace

université de bretagne occidentale

UPVD Université de Perpignan Via Domitia

Overview

Floating point arithmetic

Program transformation for numerical precision

Examples

Conclusion

Context

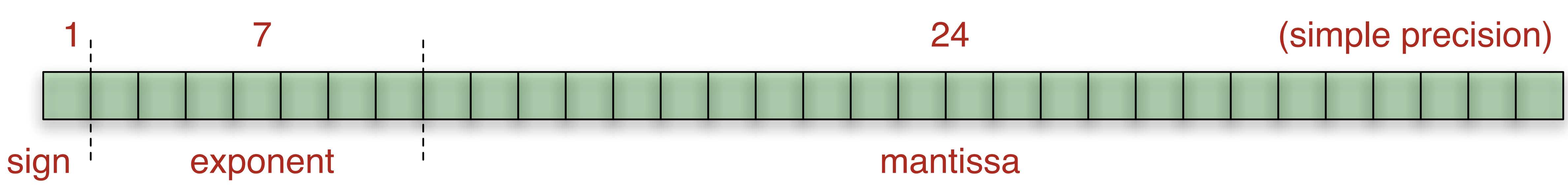
Floating point arithmetic:

Not intuitive (usual algebraic laws do not hold)

Difficult to predict the precision of a computation

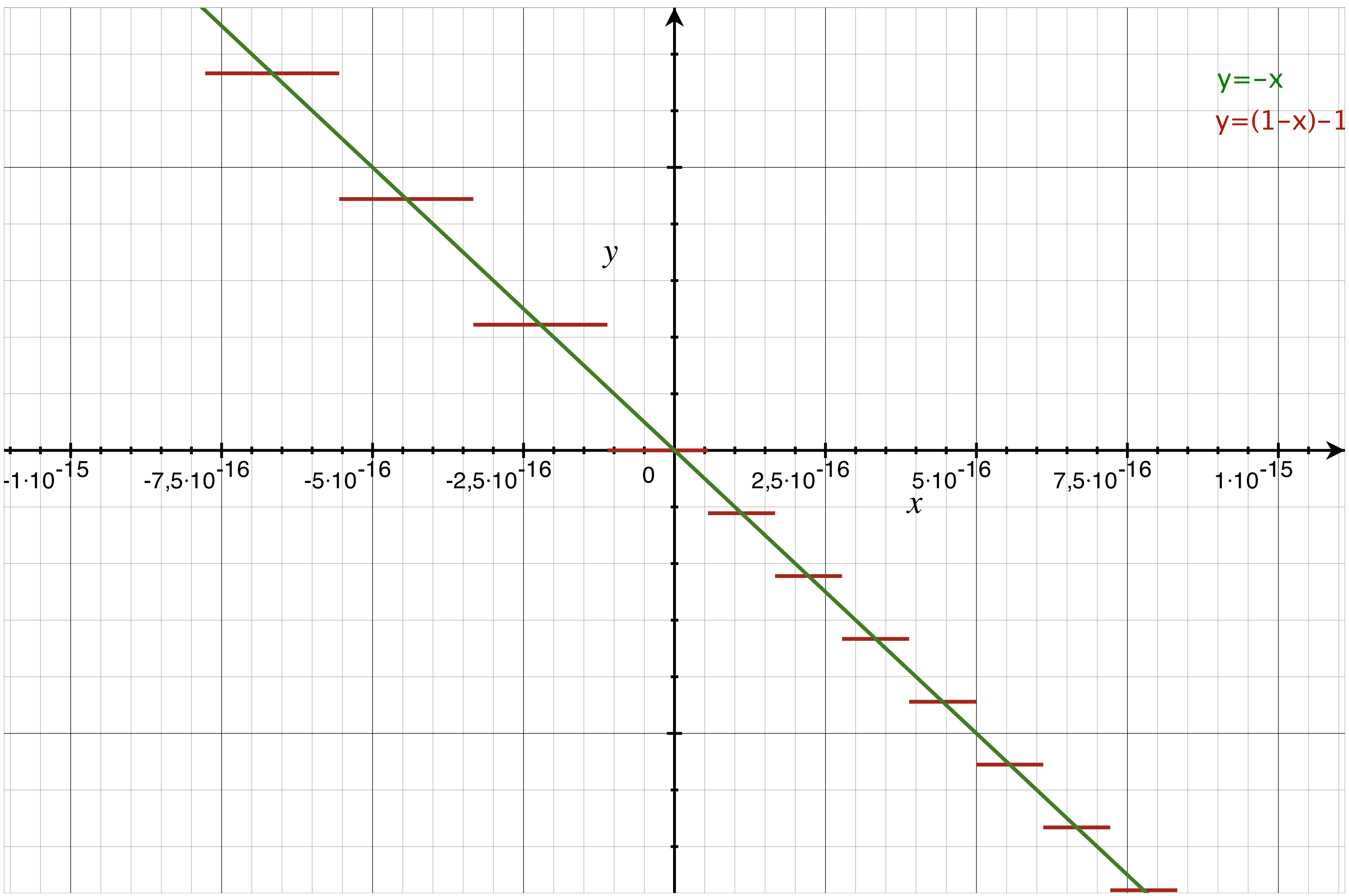
No method to improve the numerical accuracy of an implementation

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



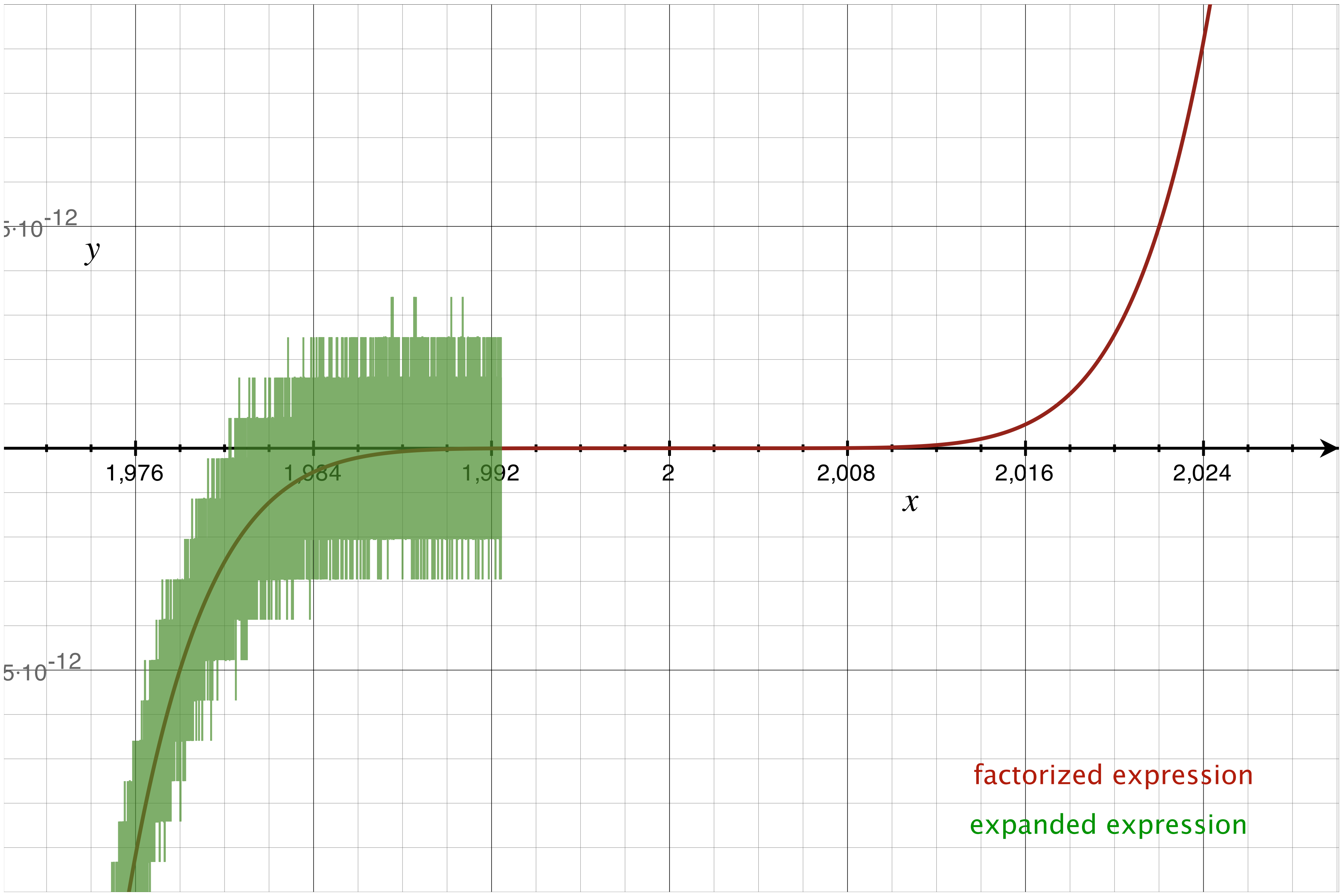
Example 1

$$f : \mathbb{F} \to \mathbb{F}$$
$$x \mapsto (1 - x) - 1$$



Example 2

$$f : \mathbb{F} \to \mathbb{F}$$
$$x \mapsto (x-2)^7$$



Objective

Semantics-based transformation to improve the numerical precision

SCADE semantics:

Computations are carried out using real numbers

Infinite precision

Algebraic laws (associativity, distribututativity, etc.)

SCADE implementation:

Object code uses floating point numbers

Roundoff errors

No algebraic laws

Overview

Floating point arithmetic

Program transformation for numerical precision

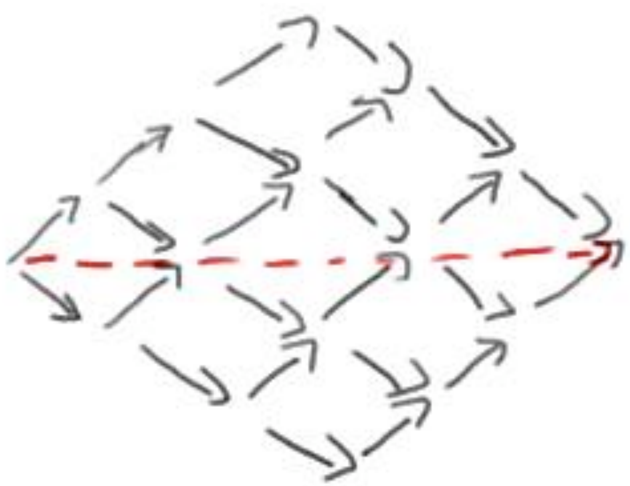
Examples

Conclusion

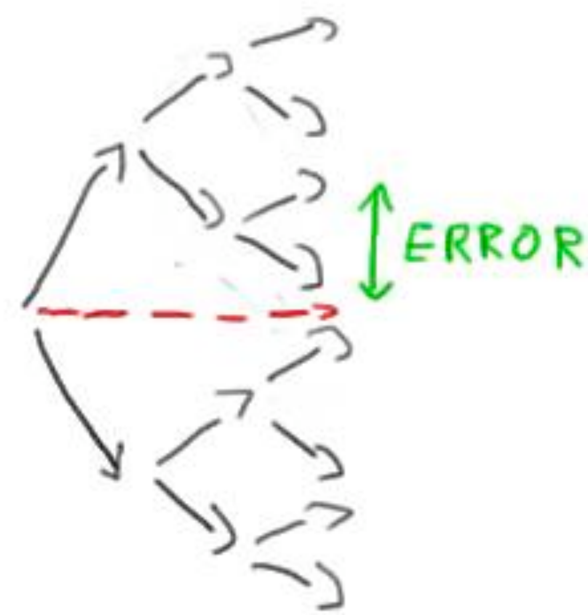
Principle of the Transformation: expressions

Allow algebraic rules in the semantics ➞ many ways to evaluate the expressions

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



REAL ARITHMETIC

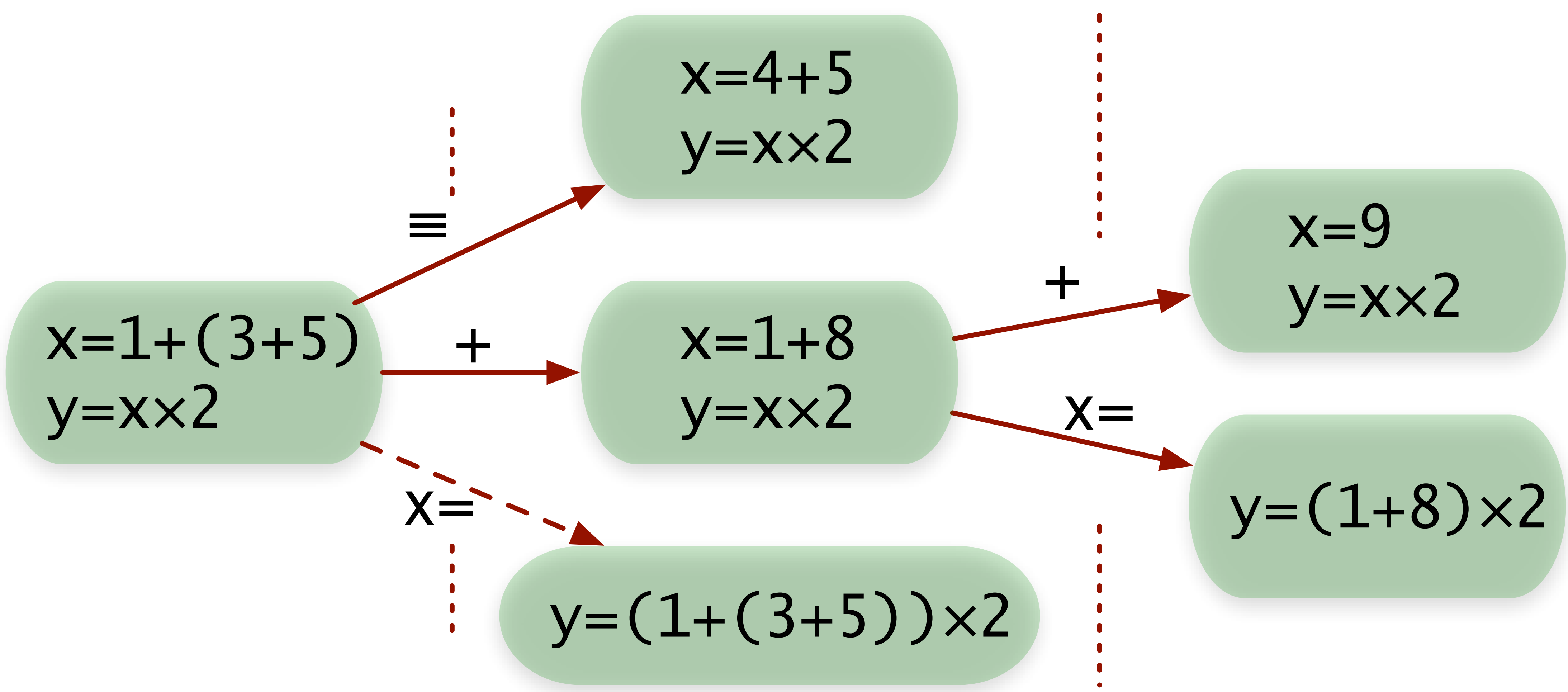


COMPUTER ARITHMETIC

Errors computed using abstract domains for numerical precision [Fluctuat]

Principle of the Transformation: programs

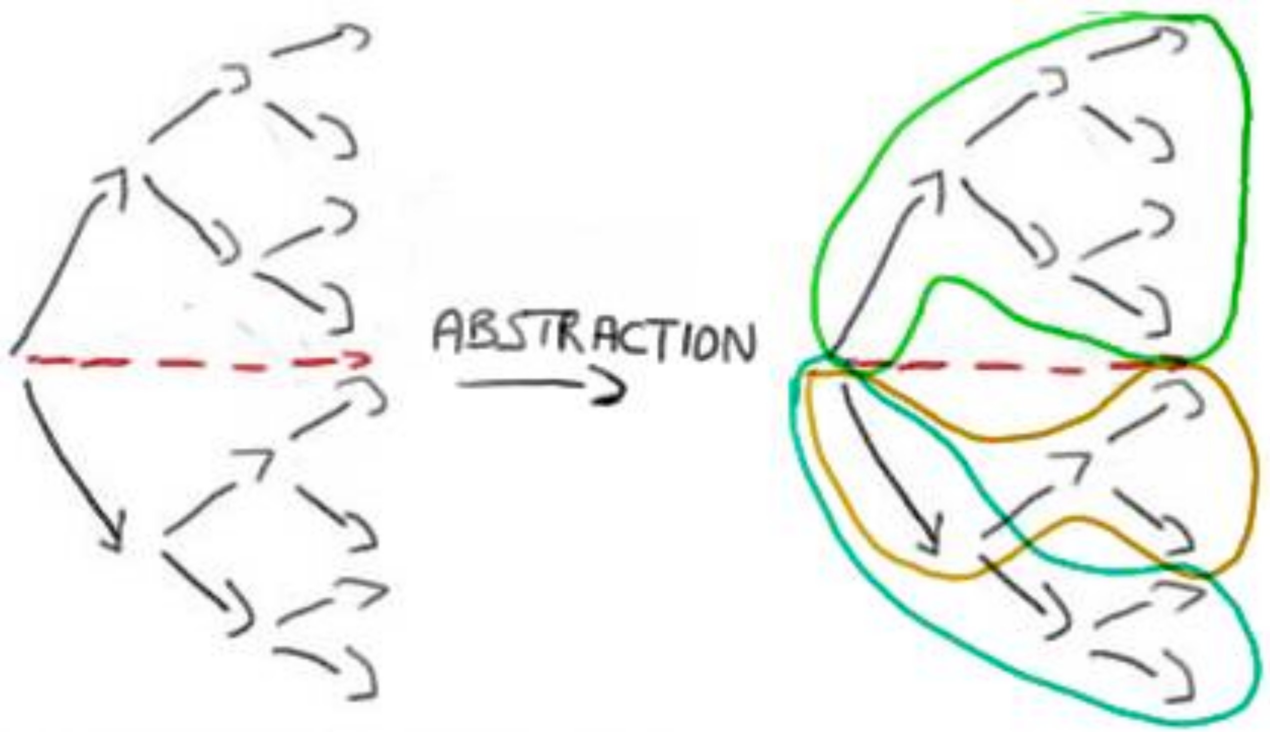
Allow to delay the evaluation of assignments ➙ many ways to evaluate programs (either eager or lazy evaluation)



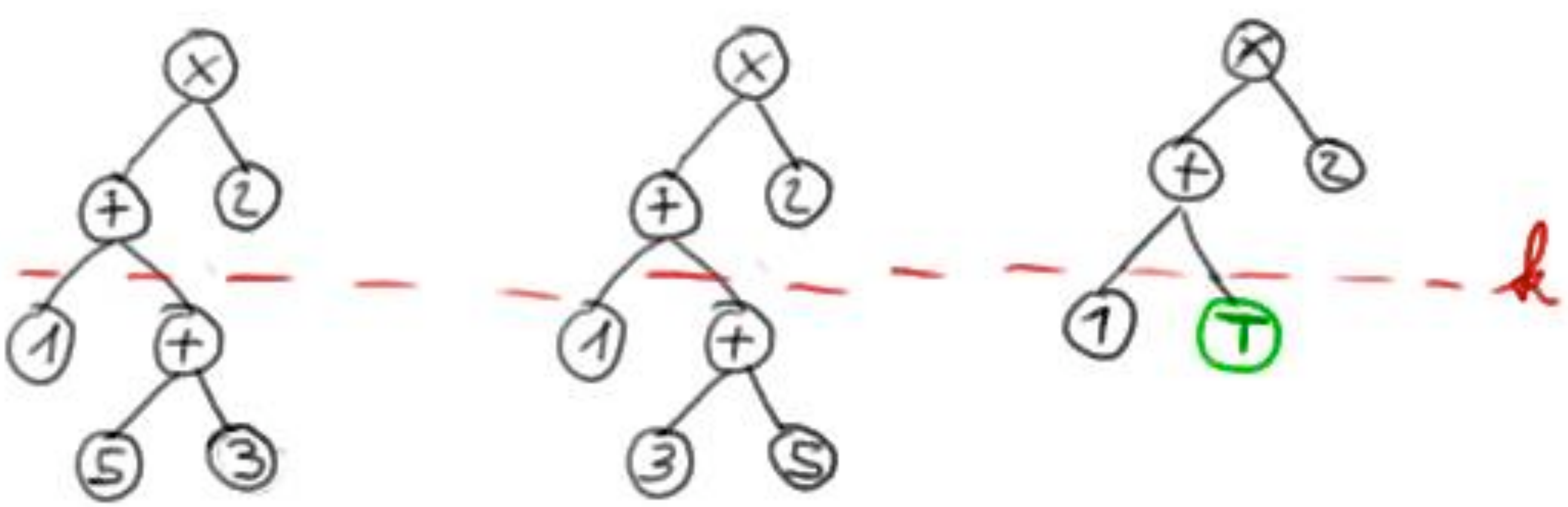
Also use standard unfolding techniques

Abstract Interpretation of the Traces

Concrete semantics: combinatorial explosion



Principle of the abstraction:



Overview

Floating point arithmetic

Program transformation for numerical precision

Examples

Conclusion

Example: expressions

Polynomials: (x∈[0,2], initial error on x∈[0,0.0005])

Case	Expression	Error bound
Source expression	`x+(x*x)`	[-1.800074334E-3,1.001074437E-3]
$k = 2$	`(1.0+x)*x`	[-9.000069921E-4,1.010078437E-4]
Source expression	`(x*(x*x))+(x*x)`	[-1.802887642E-3,3.191200091E-3]
$k = 3$	`(x+1.0)*(x*x)`	[-1.818142851E-4,1.390014781E-3]
$k = 4$	`((1.0+x)*x)*x`	[-9.091078216E-5,1.100112212E-3]

Sums:

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

Case	Expression	Error bound
Source expression	`(((e+d)+c)+b)+a`	[-7.6293E-6,7.6294E-6]
$k = 1$	`(b+a)+(c+(e+d))`	[-5.9604E-6,5.9605E-6]
$k = 2$	`(c+(b+a))+(e+d)`	[-4.5299E-6,4.5300E-6]
$k = 3$	`(d+(c+(a+b)))+e`	[-3.5762E-6,3.5763E-6]

(a, b, c, d and e stand for x_0, x_1, x_2, x_3 and x_4)

Example: loops

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

$$\text{i.e. } x_N = [1,2],\ x_{N-1} = [2,4] \ldots$$

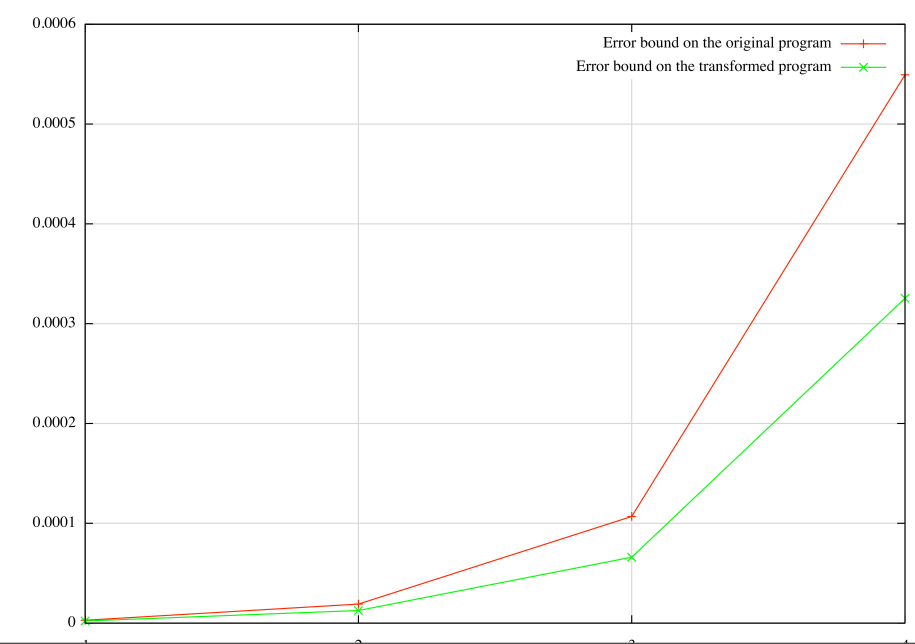
```
int i=0;
float s=0.0;
while (i<N) {
  s=s+x[i];
  i=i+1;
}
```

```
int i=0;
float s=0.0;
while (i<N) {
  s=s+x[i];
  s=s+x[i+1];
  i=i+2;
}
```

```
int i=0;
float s=0.0;
while (i<N) {
  l=x[i]+x[i+1];
  s=s+l;
  i=i+2;
}
```

For $N = 4$:

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



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

```
float a=[0.0,100.0];
float b=[0.0,0.001];
float c=[5.0,10.0];
float d=2.0;
if (a>0.1) {
  r=d*(a+(b+c));
  ...
} else {
  r=d*(a+(b+c));
  ...
}
```

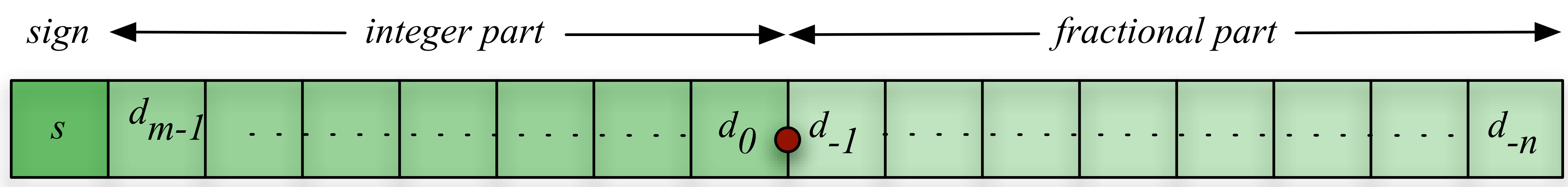
```
float a=[0.0,100.0];
float b=[0.0,0.001];
float c=[5.0,10.0];
float d=2.0;
if (a>0.1) {
  r=d*(a+(b+c));
  ...
} else {
  r=d*((a+b)+c);
  ...
}
```

Case	Error bound on r
$p,\ a > 0.1$	[-8.583068847E-6,-7.629394532E-6]
$p_t,\ a \leq 0.1$	[-1.907348632E-6,-9.536743165E-7]
$p,\ a > 0.1$	[-8.583068847E-6,-7.629394532E-6]
$p_t,\ a \leq 0.1$	[-9.611248970E-7,-9.536743165E-7]

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?

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

$a = [-14, -13]$ $\quad$ $b = [-3, -2]$

$c = [3, 3.5]$ $\quad$ $d = [12.5, 13.5]$ $\quad$ $e = 2$

$$\mathrm{E}'_{\textsf{float}} = ((a + b) \times e) + ((c + d) \times e)$$

$$\mathrm{E}'_{\textsf{fixed}} = e \times ((a + d) + (b + c))$$

(can be extended to interval arithmetic)

Overview

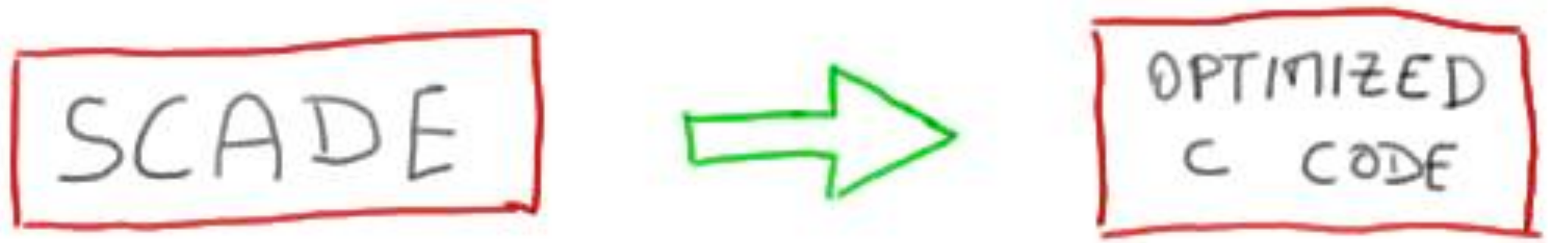
Floating point arithmetic

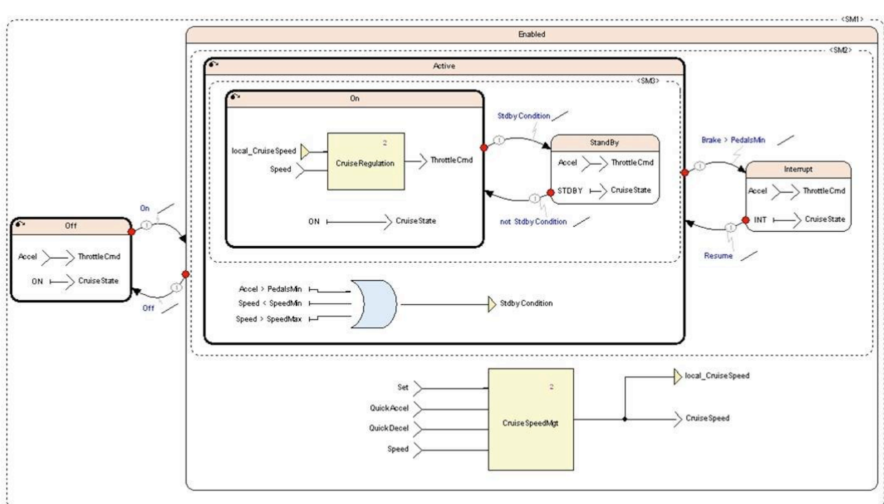
Program transformation for numerical precision

Examples

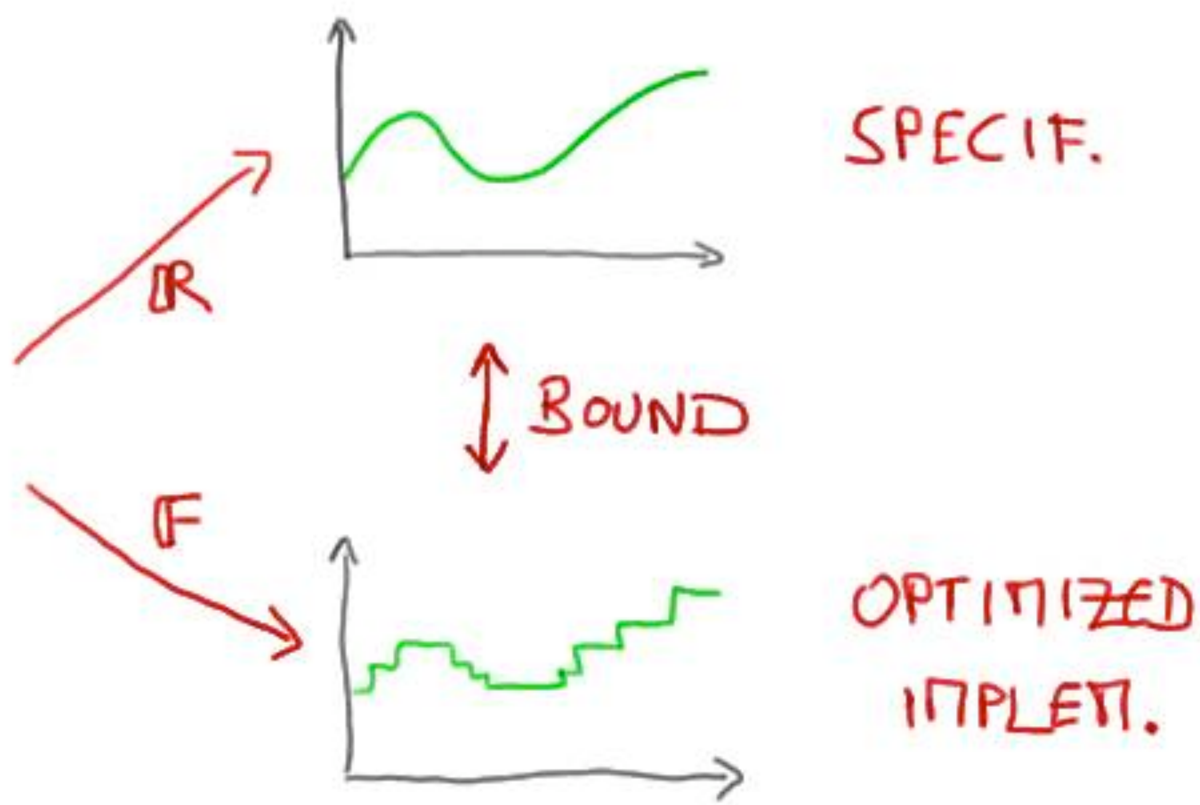
Conclusion

SARDANES: Summary





Top Level of the Cruise Control application



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

- Matthieu Martel, *Program Transformation for Numerical Precision*, Partial Evaluation and Program Manipulation, PEPM'09, ACM Press, 2009
- Matthieu Martel, *Enhancing the Implementation of Mathematical Formulas for Fixed-Point and Floating-Point Arithmetics*, International Workshop on Numerical Abstractions for Software Verification, invited paper, 2008
- Matthieu Martel, *Semantics-Based Transformation of Arithmetic Expressions*, Static Analysis Symposium, SAS'07, Lecture Notes in Computer Science, 4634, Springer-Verlag, 2007
- Patrick Cousot and Radhia Cousot, *Systematic Design of Program Transformation Frameworks by Abstract Interpretation*, Principles of Programming Languages, POPL'02, ACM Press, 2002

- http://gala.univ-perp.fr/~mmartel/sardanes.html

QUESTIONS?