

Static analysis of numerical programs

Sylvie Putot

with Eric Goubault, Franck Védrine and Karim Tekkal (Digiteo)

Laboratory for the Modelling and Analysis
of Interacting Systems, CEA LIST

RAIM'09: 3es Rencontres Arithmétique de l'Informatique
Mathématique - LIP, ENS Lyon

Validation of numerical programs

- ▶ Conception designed in real numbers: what is the impact of a finite precision implementation, what is a correct program ?
 - ▶ No run-time error, such as division by 0, overflow, etc
 - ▶ The program does compute something “not too far” from what is expected (=the result computed in real numbers)
 - ▶ No problematic control-flow difference between real and floating-point computation (same nb of iterations)
 - ▶ Can we also prove the algorithm correct (method error) ?

Householder scheme for square root computation

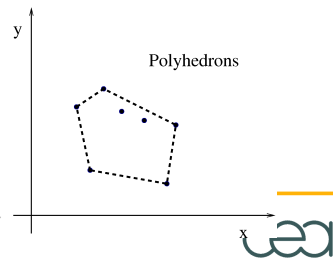
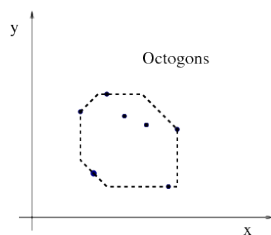
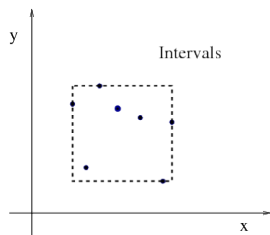
Static Analysis

- ▶ Analysis of the source code, for a set of inputs and parameters, without executing it:
 - ▶ does the program always terminate?
 - ▶ does the program ever reach a bad state?
 - ▶ is there a possibility of run-time error, such as division by zero?
 - ▶ synchronization errors (deadlocks, data races)?
 - ▶ does the program compute accurately?
- ▶ The ideal automatic static analyzer (eg for run-time error) is
 - ▶ sound: if there is an error, the analyzer reports it
 - ▶ complete: if the analyzer reports an error, it is a genuine one
- ▶ Interesting program properties are undecidable in general:
 - ▶ in general choose sound (but not complete)
 - ▶ then focus on trade-off between performance and accuracy

Static Analysis by Abstract Interpretation (Cousot 77)

Computable over-approximations of sets of values at any point in the program, for any possible execution:

- ▶ the program is considered as a discrete dynamical system
- ▶ invariants are computed as solution of a fix-point equation



FLUCTUAT static analyzer: models float as real + error

```
float x,y,z;  
x = 0.1; // [1]  
y = 0.5; // [2]  
z = x+y; // [3]  
t = x*z; // [4]
```

$$f^x = 0.1 + 1.49e^{-9} [1]$$

$$f^y = 0.5$$

$$f^z = 0.6 + 1.49e^{-9} [1] + 2.23e^{-8} [3]$$

$$f^t = 0.06 + 1.04e^{-9} [1] + 2.23e^{-9} [3] - 8.94e^{-10} [4] - 3.55e^{-17} [ho]$$

⇒ Then abstraction for each term (real value and errors)

Sound abstraction based on Affine Arithmetic

- ▶ The *real value* of variable x is represented by an affine form \hat{x} :

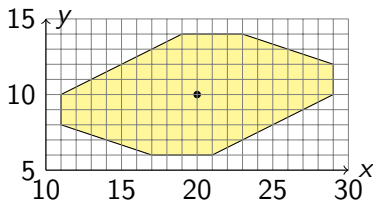
$$\hat{x} = x_0 + x_1\varepsilon_1 + \dots + x_n\varepsilon_n,$$

where $x_i \in \mathbb{R}$ and the ε_i are independent symbolic variables with unknown value in $[-1, 1]$.

- ▶ Sharing ε_i between variables expresses *implicit dependency*: concretization as a zonotope

$$\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$$

$$\hat{y} = 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$$



Abstract domain based on affine arithmetic

- ▶ *Assignment* of a variable x whose value is given in a range $[a, b]$ introduces a noise symbol ε_i :

$$\hat{x} = \frac{(a + b)}{2} + \frac{(b - a)}{2} \varepsilon_i.$$

- ▶ functional abstraction: link to the inputs via the noise symbols, allowing sensitivity analysis and worst case generation
- ▶ *Addition* is computed componentwise (no new noise symbol):

$$\hat{x} + \hat{y} = (\alpha_0^x + \alpha_0^y) + (\alpha_1^x + \alpha_1^y)\varepsilon_1 + \dots + (\alpha_n^x + \alpha_n^y)\varepsilon_n$$

- ▶ *Non linear operations* : approximate linear form (Taylor expansion), new noise term for the approximation error
- ▶ Efficient join operator (on-going work for a better meet operator)

Union over affine forms

We define $z = x \cup y$ by $z = \alpha_0^z + \sum_i \alpha_i^z \varepsilon_i + \beta^z \varepsilon_U$ with

$$\begin{cases} \alpha_0^z = \text{mid}([\hat{x}] \cup [\hat{y}]) \\ \alpha_i^z = \underset{\alpha_i^x \wedge \alpha_i^y \leq \alpha \leq \alpha_i^x \vee \alpha_i^y}{\text{argmin}} |\alpha|, \forall i \geq 1 \\ \beta^z = \sup \gamma(\hat{x}) \cup \gamma(\hat{y}) - \alpha_0^z - \|z\|_1 \end{cases}$$

$$\underset{u \wedge v \leq \alpha \leq u \vee v}{\text{argmin}} |\alpha| = \{\alpha \in [u \wedge v, u \vee v], |\alpha| \text{ minimal}\}$$

Intuitively, α_i^z expresses the common dependency to symbol ε_i , and the remainder is associated to a new noise symbol ε_U

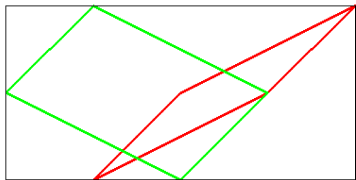
- ▶ efficient (linear in the number of symbols, and eliminates part of the symbols)
- ▶ range of values taken by the union is the union of the ranges

Example

$$\hat{x} = 3 + \varepsilon_1 + 2\varepsilon_2$$

$$\hat{y} = 1 - 2\varepsilon_1 + \varepsilon_2$$

$$\hat{u} = \varepsilon_1 + \varepsilon_2$$



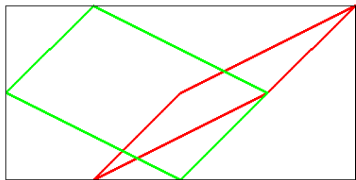
\hat{x} et \hat{y} functions of \hat{u}

Example

$$\hat{x} = 3 + \varepsilon_1 + 2\varepsilon_2$$

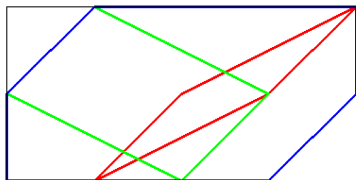
$$\hat{y} = 1 - 2\varepsilon_1 + \varepsilon_2$$

$$\hat{u} = \varepsilon_1 + \varepsilon_2$$



\hat{x} et \hat{y} functions of \hat{u}

$$\hat{z} = \hat{x} \cup \hat{y} = 2 + \varepsilon_2 + 3\varepsilon_U$$



\hat{x} , \hat{y} et \hat{z} functions of \hat{u}

$$\gamma(\hat{z}) = [-2, 6] = \gamma(\hat{x}) \cup \gamma(\hat{y})$$

- ▶ Takes source C code (most of ANSI C, except union types and malloc most notably), with assertions (for instance range of values and imprecision on input, but also range of gradient of evolution of values)
- ▶ Gives, fully automatically, characterization of ranges/errors, and describe the origins of errors: identification of pieces of code with numerical difficulties
- ▶ But also, in some cases, weak functional proof of algorithms
- ▶ Is/has been used for a wide variety of codes (automotive, nuclear industry, aeronautics, aerospace) of size up to about 50000 LOCs (on laptop PCs 1Gb)



Back to the Householder scheme

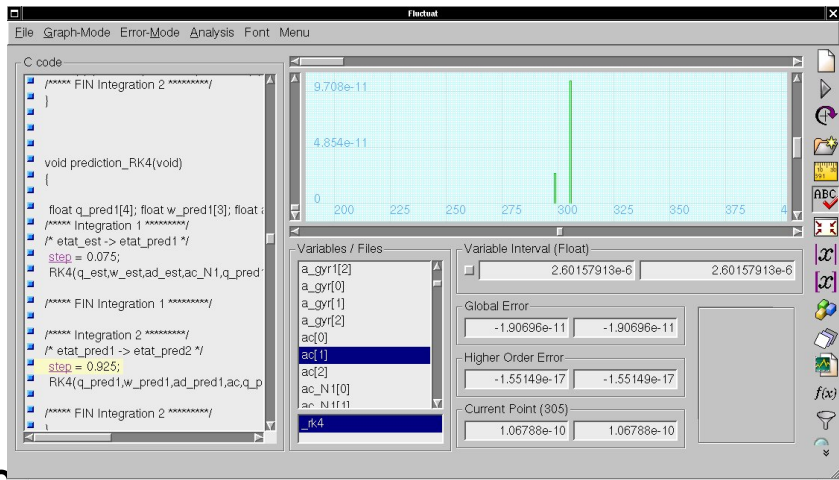
Second order filters

Astrium ST case study for ATV (33KLoc)

- ▶ Central control module that monitors the behaviour of the spacecraft while docking ISS station: Kalman filter
- ▶ Iterate, up to 1200 seconds:
 - ▶ Computation of the estimated state (based on data from the navigation module and from previous predicted states) and commanded acceleration
 - ▶ Filtering and saturation of the acceleration (8th order linear filter)
 - ▶ Control of the *real* acceleration of the engine (using inputs from sensors)
 - ▶ Computation of the predicted state : integration of the motion equations using 4th order Runge-Kutta scheme



Identifying sources of errors



Explanation

- ▶ Each global step of 1 second is divided in two integration steps $step_1 = 0.075$ and $step_2 = 0.925$,
- ▶ $step_1$ and $step_2$ are not exactly represented as float (error on $step_1$ is about $3e^{-9}$ and error on $step_2$ is about $1.2e^{-8}$).
- ▶ The integration is thus computed on more than one second at each step. Naturally the error on $step_2$ prevails, as found by FLUCTUAT.
- ▶ Relative error insignificant considering the maximum duration of integration ($1.8e^{-5}$ for 1200s), but exactly representable steps with sum equal to 1, such as 0.078125 and 0.921875 would be much better.



- ▶ Abstract domains based on affine forms for the computation of invariants on numerical programs:
 - ▶ Static Analysis of Numerical Algorithms, *SAS 2006 (Static Analysis Symposium)*
 - ▶ Under-Approximations of Computations in Real Numbers Based on Generalized Affine Arithmetic, *SAS 2007 (Static Analysis Symposium)*
 - ▶ Perturbed affine arithmetic for invariant computation in numerical program analysis, *arXiv:0807.2961, july 2008*
 - ▶ The Zonotope Abstract Domain Taylor1+, *CAV 2009 (Computer Aided Verification)*
 - ▶ A Zonotopic Framework for Functional Abstractions, *arXiv:0910.1763, october 2009*

- ▶ Implementation and use of FLUCTUAT; industrial case studies
 - ▶ Static Analysis of the Accuracy in Control Systems: Principles and Experiments (with IRSN - Institut de Radioprotection et de Sécurité Nucléaire - and Hispano Suiza), *FMICS 2007 (Formal Methods for Industrial Critical Systems)*
 - ▶ Validation using Abstract Interpretation (with ESA, ASTRIUM SAS, ENS), *DASIA 2009 (DATA Systems In Aerospace Space Software)*
 - ▶ HybridFluctuat: A Static Analyzer of Numerical Programs within a Continuous Environment, *CAV 2009*
 - ▶ Towards an industrial use of FLUCTUAT on safety-critical avionics software (with Airbus), *FMICS 2009*

