DE LA RECHERCHE À L'INDUSTRIE

DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

Static Analysis of Numerical Programs and Systems

Sylvie Putot

Habilitation à Diriger les Recherches | Saclay Nanoinnov, December 13, 2012

During the last ten years...

Some steps

- Starting point (E. Goubault, SAS 2001): bound and propagate rounding errors (intervals) and indicate provenance
  - hired Jan. 01 (PhD Num. Analysis): first implement these ideas (ESOP'02)
- Zonotopic abstract domains
  - first ideas for the error quite early (Dagstuhl Seminar 2002)
  - first focus on the real value (SAS 2006, arxiv 2008, arxiv 2009)
  - back to errors (VMCAI 2011, much sooner in Fluctuat...)
  - extensions (SAS 2007, Computing 2012)
- Meanwile fixpoint computation (CAV 2005), development of FLUCTUAT

Environment

- Academic projects: IST Daedalus (2000-2002), ANR Eva-Flo (2003-2005), ANR ASOPT (2009-2011), ANR CPP (2009-2012), ANR DEFIS (2012-2015) Digiteo PASO (2009-2011), SANSCRIT (2010-2013)
  - most of them: interactions between program analysis and numerical computation, optimisation, control, probabilities
- Industrial collaborations: many projects and some publications (FMICS 2007, FMICS 2009, DASIA 2009)

What is correction for numerical computations?

- No run-time error (division by 0, overflow, etc)
- The program computes a result close to what is expected
  - accuracy (and behaviour) of finite precision computations
  - method error

Context: safety-critical programs

- Typically flight control or industrial installation control (signal processing, instrumentation software)

Sound and automatic methods

- Guaranteed methods, that prove good behaviour or else try to give counter-examples
- Automatic methods, given a source code, and sets of (possibly uncertain) inputs and parameters

Abstract interpretation based static analysis

Automatic invariant synthesis

- Program seen as system of equations $X^{n+1} = F(X^n)$
  - Based on a notion of control points in the program
  - Equations describe how values of variables are collected at each control point, for all possible executions (collecting semantics)

Example

```
int x=[-100,50]; [1]
while [2] (x < 100)
  [3] x=x+1; [4]
 [5]
```

$$\begin{cases} x_1 = [-100, 50] \\ x_2 = x_1 \cup x_4 \\ x_3 =]-\infty, 99] \cap x_2 \\ x_4 = x_3 + [1, 1] \\ x_5 = [100, +\infty[\cap x_2 \end{cases}$$

Automatic invariant synthesis

- Program seen as a system of equations $X^{n+1} = F(X^n)$
- Want to compute reachable or invariant sets at control points
- Least fixpoint computation on partially ordered structure
  - classically computed as the limit of the Kleene iteration

$$X^0 = \bot, X^1 = F(X^0), \ldots, X^{k+1} = X^k \cup F(X^k)$$

  - or policy iteration (Newton-like method)
- Generally not computable

Automatic invariant synthesis

- Program seen as a system of equations $X^{n+1} = F(X^n)$
- Want to compute reachable or invariant sets at control points
- Least fixpoint computation on partially ordered structure
  - classically computed as the limit of the Kleene iteration
    $$X^0 = \bot, X^1 = F(X^0), \ldots, X^{k+1} = X^k \cup F(X^k)$$
  - or policy iteration (Newton-like method)
- Generally not computable

Sound abstractions heavily relying on set-based methods

- Choose a computable abstraction that defines an over or under-approximation of set of values
- Need a partially ordered structure, with join and meet operators
- Transfer concrete fixpoint computation in the abstract world

Outline

- Affine sets: zonotopic abstraction of real computations
- A word on fixpoint computations
- Extensions of the zonotopic approach
- Analyzing finite precision computations: Fluctuat

Affine forms

- Affine form for variable $\mathtt{x}$:

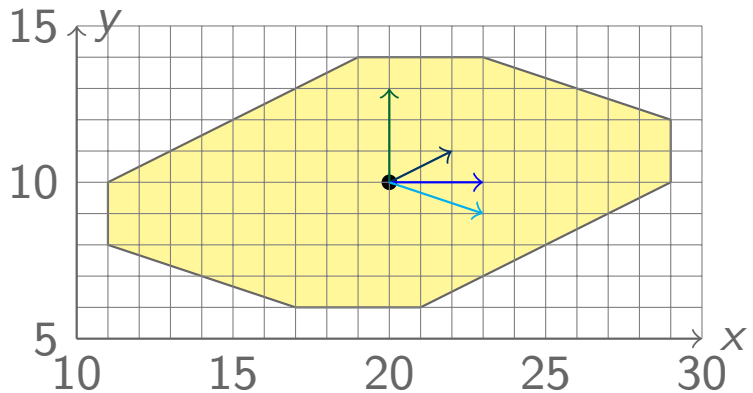
$$\hat{x} = x_0 + x_1\varepsilon_1 + \ldots + x_n\varepsilon_n,\ x_i \in \mathbb{R}$$

  where the ε_i are symbolic variables (*noise symbols*), with value in $[-1, 1]$.
- Sharing ε_i between variables expresses *implicit dependency*
- Interval concretization of affine form $\hat{x}$:

$$\left[x_0 - \sum_{i=0}^{n} |x_i|, x_0 + \sum_{i=0}^{n} |x_i|\right]$$

Geometric concretization as zonotopes in $\mathbb{R}^p$

$$\begin{aligned} \hat{x} &= 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4 \\ \hat{y} &= 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4 \end{aligned}$$



Affine arithmetic

- Assignment $x := [a, b]$ introduces a noise symbol:

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

- Addition/subtraction are exact:

$$\hat{x} + \hat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + \ldots + (x_n + y_n)\varepsilon_n$$

- Non linear operations : approximate linear form, new noise term bounding the approximation error
- Close to Taylor models of low degree (large ranges for static analysis)

Numerical abstract domain (in short!)

Concretization-based analysis

- Machine-representable abstract values X (affine sets)
- A concretization function γ_f defining the set of concrete values represented by an abstract value
- A partial order on these abstract values, induced by γ_f:
  $X \sqsubseteq Y \iff \gamma_f(X) \subseteq \gamma_f(Y)$

Abstract transfer functions

- Arithmetic operations: F is a sound abstraction of f iff

$$\forall x \in \gamma_f(X),\ f(x) \in \gamma_f(F(X))$$

- Set operations: join ($\cup$), meet ($\cap$), widening
  - no least upper bound / greatest lower bound on affine sets
  - (minimal) upper bounds / over-approximations of the intersection ...

and ... hopefully accurate and effective to compute!!!

Affine sets as an abstract domain (Arxiv 2008 & 2009)

Two kinds of noise symbols

- Input noise symbols (ε_i): created by uncertain inputs
- Perturbation noise symbols (η_j): created by uncertainty in analysis

Perturbed affine sets $X = (C^X, P^X)$

$$\begin{pmatrix} \hat{x_1} \\ \hat{x_2} \\ \dots \\ \hat{x_p} \end{pmatrix} = {}^tC^X \begin{pmatrix} 1 \\ \varepsilon_1 \\ \dots \\ \varepsilon_n \end{pmatrix} + {}^tP^X \begin{pmatrix} \eta_1 \\ \eta_2 \\ \dots \\ \eta_m \end{pmatrix}$$

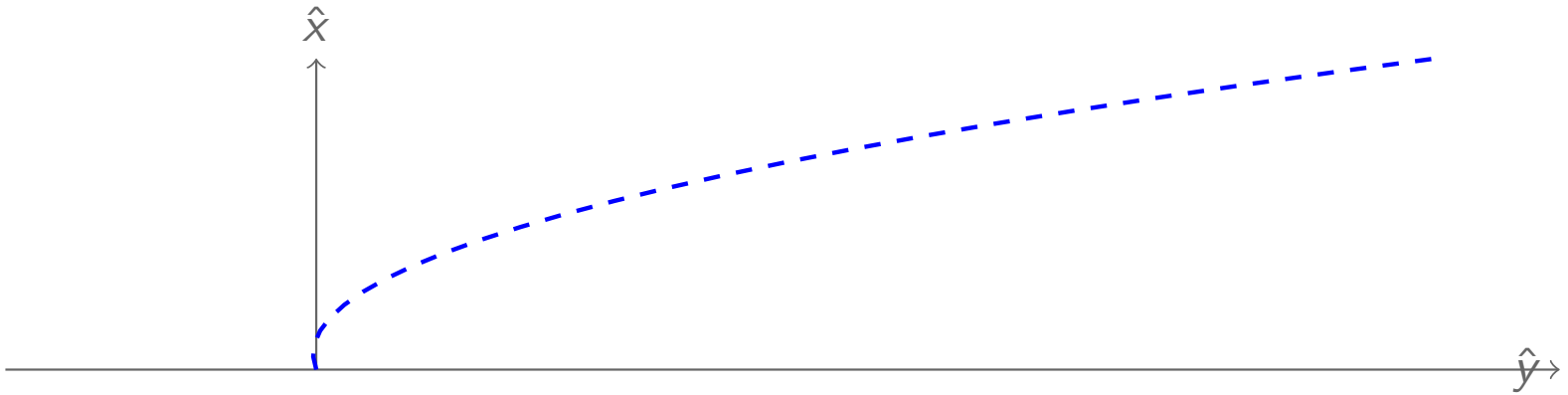
- Central part links the current values of the program variables to the initial values of the input variables (linear functional)
- Perturbation part encodes the uncertainty in the description of values of program variables due to non-linear computations (multiplication, join etc.)

Zonotopes define input-output relations (parameterization by the ε_i)

- Want an order that preserves these input-output relations

A simple example: functional interpretation

```
real x = [0,10];
real y = x*x - x;
```

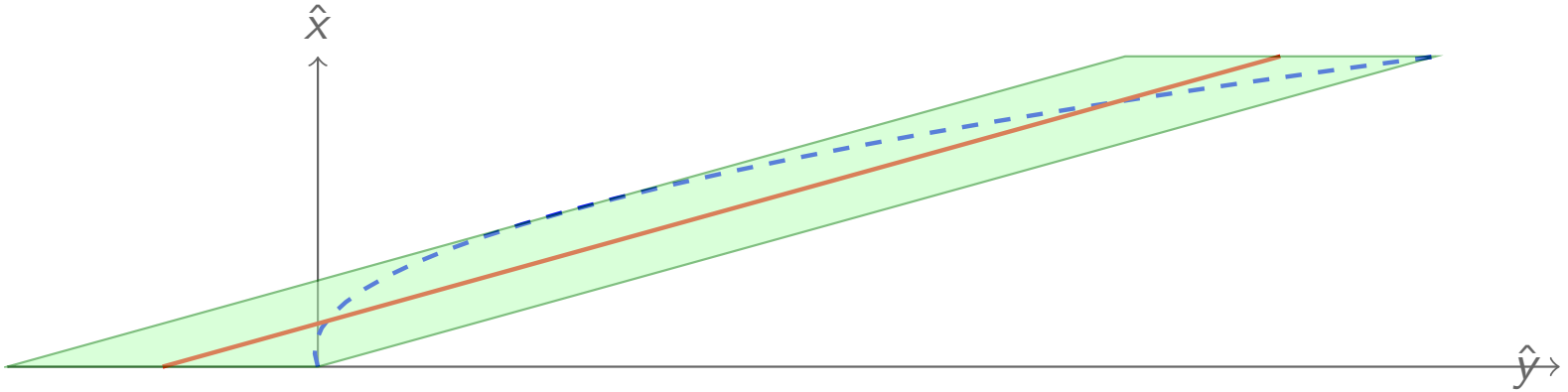


Abstraction of function $x \to y = x^2 - x$ as

$$y = 32.5 + 45\varepsilon_1 + 12.5\eta_1$$

A simple example: functional interpretation

```
real x = [0,10];
real y = x*x - x;
```



Abstraction of function $x \to y = x^2 - x$ as

$$\begin{aligned} y &= 32.5 + 45\varepsilon_1 + 12.5\eta_1 \\ &= -12.5 + 9x + 12.5\eta_1 \end{aligned}$$

(since $x = 5 + 5\varepsilon_1$)

Functional order relation

Concretization in terms of sets of functions from $\mathbb{R}^n$ to $\mathbb{R}^p$:

$$\gamma_f(X) = \left\{ f : \mathbb{R}^n \to \mathbb{R}^p \mid \forall \epsilon \in [-1,1]^n, \exists \eta \in [-1,1]^m, f(\varepsilon) = {}^tC^X \begin{pmatrix} 1 \\ \varepsilon \end{pmatrix} + {}^tP^X \eta \right\}.$$

- Equivalent to the geometric ordering on augmented space $\mathbb{R}^{p+n}$: zonotopes enclosing current values of variables + their initial values ε_i

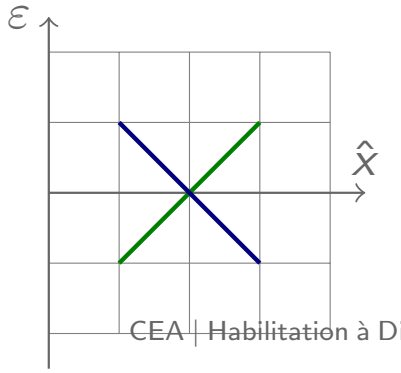
$$\gamma(\tilde{X}) \subseteq \gamma(\tilde{Y}), \text{ where } \tilde{X} = \begin{pmatrix} \begin{pmatrix} 0_n \\ I_{n\times n} \end{pmatrix} & C^X \\ 0_{m\times n} & P^X \end{pmatrix}$$

  - Implies the geometric ordering in $\mathbb{R}^p$
- Computable inclusion test:

$$X \sqsubseteq Y \iff \sup_{u\in\mathbb{R}^p} \left(\|(C^Y - C^X)u\|_1 + \|P^X u\|_1 - \|P^Y u\|_1 \right) \leq 0$$

Example

- $x_1 = 2 + \varepsilon_1$, $x_2 = 2 - \varepsilon_1$
- x_1 and x_2 are incomparable



Functional order relation

Concretization in terms of sets of functions from $\mathbb{R}^n$ to $\mathbb{R}^p$:

$$\gamma_f(X) = \left\{ f : \mathbb{R}^n \to \mathbb{R}^p \mid \forall \epsilon \in [-1,1]^n, \exists \eta \in [-1,1]^m, f(\varepsilon) = {}^tC^X \begin{pmatrix} 1 \\ \varepsilon \end{pmatrix} + {}^tP^X \eta \right\}.$$

- Equivalent to the geometric ordering on augmented space $\mathbb{R}^{p+n}$: zonotopes enclosing current values of variables + their initial values ε_i

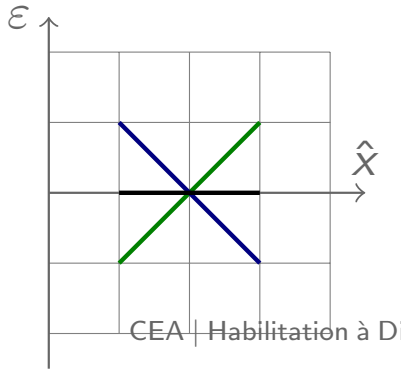
$$\gamma(\tilde{X}) \subseteq \gamma(\tilde{Y}), \text{ where } \tilde{X} = \begin{pmatrix} \begin{pmatrix} 0_n \\ I_{n\times n} \end{pmatrix} & C^X \\ 0_{m\times n} & P^X \end{pmatrix}$$

  - Implies the geometric ordering in $\mathbb{R}^p$
- Computable inclusion test:

$$X \sqsubseteq Y \iff \sup_{u\in\mathbb{R}^p} \left(\|(C^Y - C^X)u\|_1 + \|P^X u\|_1 - \|P^Y u\|_1 \right) \leq 0$$

Example

- $x_1 = 2 + \varepsilon_1$, $x_2 = 2 - \varepsilon_1$ (geometric concretization [1, 3])
- x_1 and x_2 are incomparable



Functional order relation

Concretization in terms of sets of functions from $\mathbb{R}^n$ to $\mathbb{R}^p$:

$$\gamma_f(X) = \left\{ f : \mathbb{R}^n \to \mathbb{R}^p \mid \forall \epsilon \in [-1,1]^n, \exists \eta \in [-1,1]^m, f(\varepsilon) = {}^tC^X \begin{pmatrix} 1 \\ \varepsilon \end{pmatrix} + {}^tP^X \eta \right\}.$$

- Equivalent to the geometric ordering on augmented space $\mathbb{R}^{p+n}$: zonotopes enclosing current values of variables + their initial values ε_i

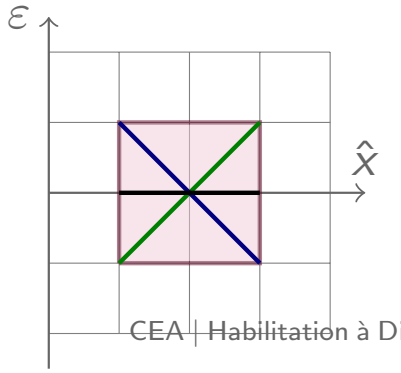
$$\gamma(\tilde{X}) \subseteq \gamma(\tilde{Y}), \text{ where } \tilde{X} = \begin{pmatrix} \begin{pmatrix} 0_n \\ I_{n\times n} \end{pmatrix} & C^X \\ 0_{m\times n} & P^X \end{pmatrix}$$

  - Implies the geometric ordering in $\mathbb{R}^p$
- Computable inclusion test:

$$X \sqsubseteq Y \iff \sup_{u\in\mathbb{R}^p} \left(\|(C^Y - C^X)u\|_1 + \|P^X u\|_1 - \|P^Y u\|_1 \right) \leq 0$$

Example

- $x_1 = 2 + \varepsilon_1$, $x_2 = 2 - \varepsilon_1$, $x_3 = 2 + \eta_1$ (geometric concretization $[1, 3]$)
- x_1 and x_2 are incomparable , both are included in x_3.



Set operations on affine sets / zonotopes

The choice of "home-made" functional join and meet operations

- Keep the parameterization by the ε_i
- These operations should not be expensive

A lot of litterature on zonotopes

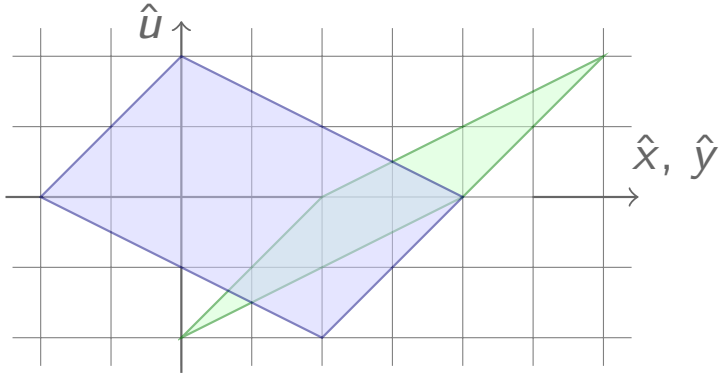
- Control theory and hybrid systems analysis: same problem of intersection of zonotopes with guards (Girard, Le Guernic etc)
- But these methods are geometrical
- Still, could be used on the perturbation part

Now: our join operator (2006-present, with E. Goubault)

- Join on coefficients of the forms (interval coefficients): no!
- Central form plus deviation (SAS 2006): $\gamma(\hat{x} \cup \hat{y})$ in general larger than $\gamma(\hat{x}) \cup \gamma(\hat{y})$, bad for fixpoint computation
- Arxiv 2008 and 2009: the componentwise upper bound presented here

Component-wise upper bound

$$\left(\begin{array}{l}\hat{x} = 3 + \varepsilon_1 + 2\varepsilon_2 \\ \hat{u} = 0 + \varepsilon_1 + \varepsilon_2\end{array}\right) \cup \left(\begin{array}{l}\hat{y} = 1 - 2\varepsilon_1 + \varepsilon_2 \\ \hat{u} = 0 + \varepsilon_1 + \varepsilon_2\end{array}\right) = \left(\begin{array}{lcl}\hat{x} \cup \hat{y} & = & 2 + \varepsilon_2 + 3\eta_1 \\ \hat{u} & = & 0 + \varepsilon_1 + \varepsilon_2\end{array}\right)$$



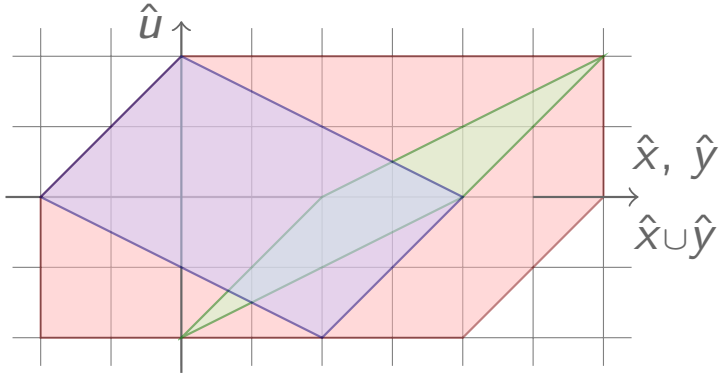
Construction (cost $\mathcal{O}(n \times p)$)

- Keep "minimal common dependencies"

$$z_i = \underset{x_i \wedge y_i \leq r \leq x_i \vee y_i}{\operatorname{argmin}} |r|, \ \forall i \geq 1$$

Component-wise upper bound

$$\left(\begin{array}{l}\hat{x} = 3 + \varepsilon_1 + 2\varepsilon_2 \\ \hat{u} = 0 + \varepsilon_1 + \varepsilon_2\end{array}\right) \cup \left(\begin{array}{l}\hat{y} = 1 - 2\varepsilon_1 + \varepsilon_2 \\ \hat{u} = 0 + \varepsilon_1 + \varepsilon_2\end{array}\right) = \left(\begin{array}{lcl}\hat{x} \cup \hat{y} & = & 2 + \varepsilon_2 + 3\eta_1 \\ \hat{u} & = & 0 + \varepsilon_1 + \varepsilon_2\end{array}\right)$$



Construction (cost $\mathcal{O}(n \times p)$)

- Keep "minimal common dependencies"

$$z_i = \underset{x_i \wedge y_i \leq r \leq x_i \vee y_i}{\operatorname{argmin}} |r|, \ \forall i \geq 1$$

- For each dimension, concretization is the interval union of the concretizations: $\gamma(\hat{x} \cup \hat{y}) = \gamma(\hat{x}) \cup \gamma(\hat{y})$

Fixpoint computation with componentwise join (arxiv 2008, with E. Goubault)

General result on recursive linear filters, pervasive in embedded programs:

$$x_{k+n+1} = \sum_{i=1}^{n} a_i x_{k+i} + \sum_{j=1}^{n+1} b_j e_{k+j}, \quad e[*] = input(m, M);$$

- Suppose this concrete scheme has bounded outputs (zeros of $x^n - \sum_{i=0}^{n-1} a_{i+1} x^i$ have modules stricty lower than 1).
- Then there exists q such that the Kleene abstract scheme "unfolded modulo q" converges towards a finite over-approximation of the outputs

$$\hat{X}_i = \hat{X}_{i-1} \cup f^q(E_i, \ldots, E_{i-k}, \hat{X}_{i-1}, \ldots, \hat{X}_{i-k})$$

in finite time, potentially with a widening partly losing dependency information

  - The abstract scheme is a perturbation (by the join operation) of the concrete scheme
  - Uses the stability property of our join operator: for each dimension $\gamma(\hat{x} \cup \hat{y}) = \gamma(\hat{x}) \cup \gamma(\hat{y})$

Some results with the APRON library (K. Ghorbal's Taylor1+ abstract domain, CAV 2009)

without widening (for $p = 5$ and $p = 16$)

filter o2	fixpoint	t(s)
Boxes	$\top$	0.12
Octagons	$\top$	2.4
Polyhedra	$[-1.3,2.82]$	0.53
T.1+(5)	$[-8.9,10.6]$	0.18
T.1+(16)	$[-5.3,6.95]$	0.13

filter o8	fixpoint	t(s)
Boxes	$\top$	0.41
Octagons	$\top$	450
Polyhedra	abort	$>24h$
T.1+(5)	$\top$	360
T.1+(16)	$\top$	942

with widening after 10 Kleene iterations (for $p = 5$ and $p = 20$)

filter o2	fixpoint	t(s)
Boxes	$\top$	<0.01
Octagons	$\top$	0.02
Polyhedra	$\top$	0.59
T.1+(5)	$[-8.9,10.6]$	0.1
T.1+(20)	$[-5.4,7.07]$	0.2

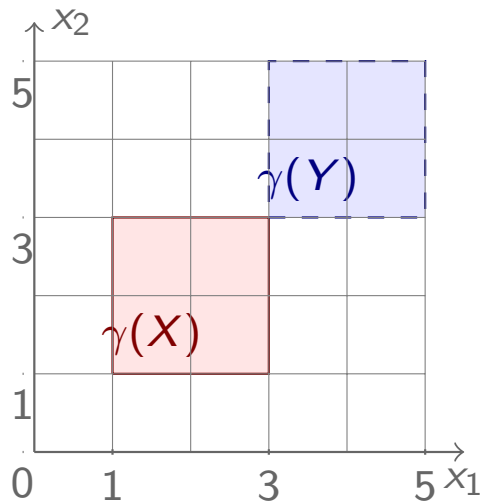
filter o8	fixpoint	t(s)
Boxes	$\top$	<0.01
Octagons	$\top$	2.56
Polyhedra	abort	$>24h$
T.1+(5)	$[-8.9,10.6]$	0.1
T.1+(20)	$[-5.4,7.07]$	0.2

Compute and preserve relations between variables and noise symbols

- Compute $k < p$ independent affine relations common to the 2 abstract values joined (solving a linear system)
- Componentwise join on $p - k$ components
- Global join for the k remaining components using the relations
- mub on the $p - k$ components $\Rightarrow$ mub on all components (else ub)

Example

$$\left(\begin{array}{l} \hat{x_1} = 2 + \varepsilon_1 \\ \hat{x_2} = 2 + \varepsilon_2 \end{array} \right) \cup \left(\begin{array}{l} \hat{y_1} = 4 + \varepsilon_1 \\ \hat{y_2} = 4 + \varepsilon_2 \end{array} \right)$$



Compute and preserve relations between variables and noise symbols

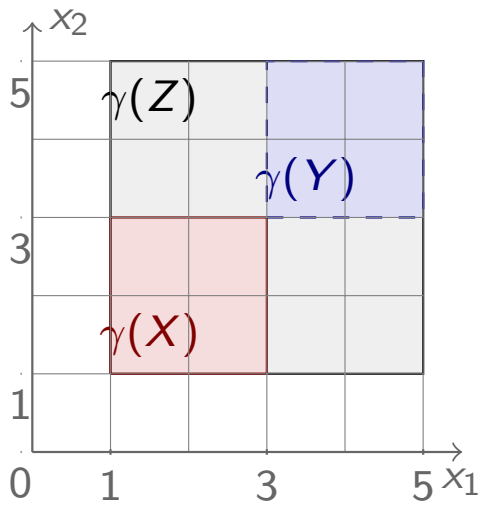
- Compute $k < p$ independent affine relations common to the 2 abstract values joined (solving a linear system)
- Componentwise join on $p - k$ components
- Global join for the k remaining components using the relations
- mub on the $p - k$ components $\Rightarrow$ mub on all components (else ub)

Example

$$\left(\begin{array}{l} \hat{x_1} = 2 + \varepsilon_1 \\ \hat{x_2} = 2 + \varepsilon_2 \end{array} \right) \cup \left(\begin{array}{l} \hat{y_1} = 4 + \varepsilon_1 \\ \hat{y_2} = 4 + \varepsilon_2 \end{array} \right)$$

Componentwise:

$$\left(\begin{array}{l} \hat{z_1} = 3 + \varepsilon_1 + \eta_1 \\ \hat{z_2} = 3 + \varepsilon_2 + \eta_2 \end{array} \right)$$



Compute and preserve relations between variables and noise symbols

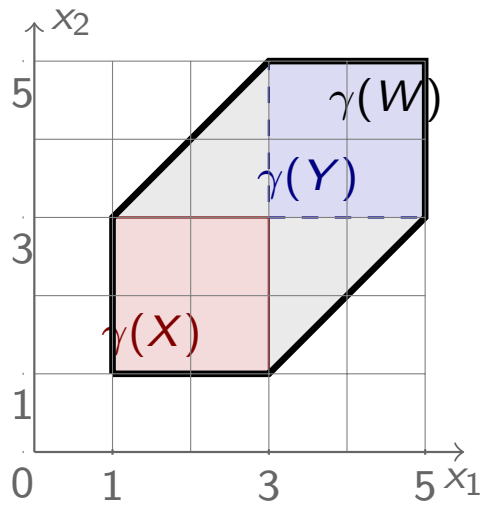
- Compute $k < p$ independent affine relations common to the 2 abstract values joined (solving a linear system)
- Componentwise join on $p - k$ components
- Global join for the k remaining components using the relations
- mub on the $p - k$ components $\Rightarrow$ mub on all components (else ub)

Example

$$\left(\begin{array}{l} \hat{x_1} = 2 + \varepsilon_1 \\ \hat{x_2} = 2 + \varepsilon_2 \end{array} \right) \cup \left(\begin{array}{l} \hat{y_1} = 4 + \varepsilon_1 \\ \hat{y_2} = 4 + \varepsilon_2 \end{array} \right)$$

Relation-preserving: $x_1 - x_2 = \varepsilon_1 - \varepsilon_2$

$$\left(\begin{array}{l} \hat{w_1} = 3 + \varepsilon_1 + \eta_1 \\ \hat{w_2} = 3 + \varepsilon_2 + \eta_1 \end{array} \right)$$



Compute and preserve relations between variables and noise symbols

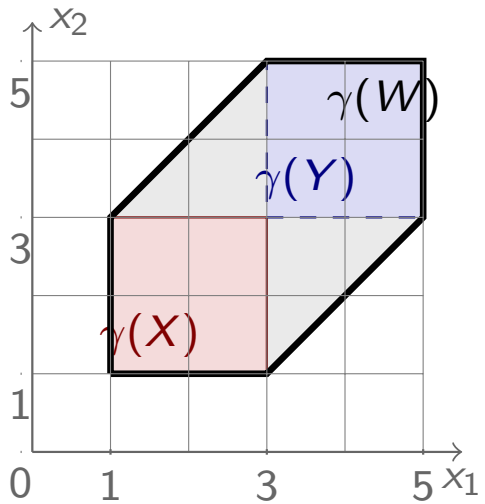
- Compute $k < p$ independent affine relations common to the 2 abstract values joined (solving a linear system)
- Componentwise join on $p - k$ components
- Global join for the k remaining components using the relations
- mub on the $p - k$ components $\Rightarrow$ mub on all components (else ub)

Example

$$\left(\begin{array}{l} \hat{x_1} = 2 + \varepsilon_1 \\ \hat{x_2} = 2 + \varepsilon_2 \end{array} \right) \cup \left(\begin{array}{l} \hat{y_1} = 4 + \varepsilon_1 \\ \hat{y_2} = 4 + \varepsilon_2 \end{array} \right)$$

Relation-preserving: $x_1 - x_2 = \varepsilon_1 - \varepsilon_2$

$$\left(\begin{array}{l} \hat{w_1} = 3 + \varepsilon_1 + \eta_1 \\ \hat{w_2} = 3 + \varepsilon_2 + \eta_1 \end{array} \right)$$



Future work: imprecise relations

Fixpoint computation: policy iteration

Policy iteration: Newton-like method to replace the widening process

- Kleene iteration with widening can be inefficient and/or inaccurate
  - especially for bounds on rounding errors (no narrowing)
- Policy iteration introduced for stochastic control problems (Howard 60)
- Pointed out to our attention by S. Gaubert and proposed together for static analysis on intervals (CAV 2005)
  - selection principle: fixpoint problem can be reduced to computing the inf of fixpoint of simpler problems
  - decreasing iteration from post-fixpoints: can stop at any time
  - often faster and more accurate than Kleene iteration, but complexity not well known (exponential worst case?)
- For relational domains (Adje-Gaubert-Goubault-Taly-Zennou, 2007-2012)
- Dual method by Gawlitza and Seidl (2007-2012)

Policy iteration and zonotopes ?

- Mainly future work, 2 approaches investigated (with intern M. Buchet and T. Le Gall): policies on the `argmin` operator (join) or on the constraints on noise symbols (meet)

Intersections (K. Ghorbal's PhD thesis, CAV 2010)

Main idea to interpret test, informally

- Translate the condition on noise symbols: constrained affine sets
- Abstract domain for the noise symbols: intervals, octagons, etc.
- Equality tests are interpreted by the substitution of one noise symbol of the constraint (cf summary instantiation for modular analysis)
- More general constraints in the future ?

Example

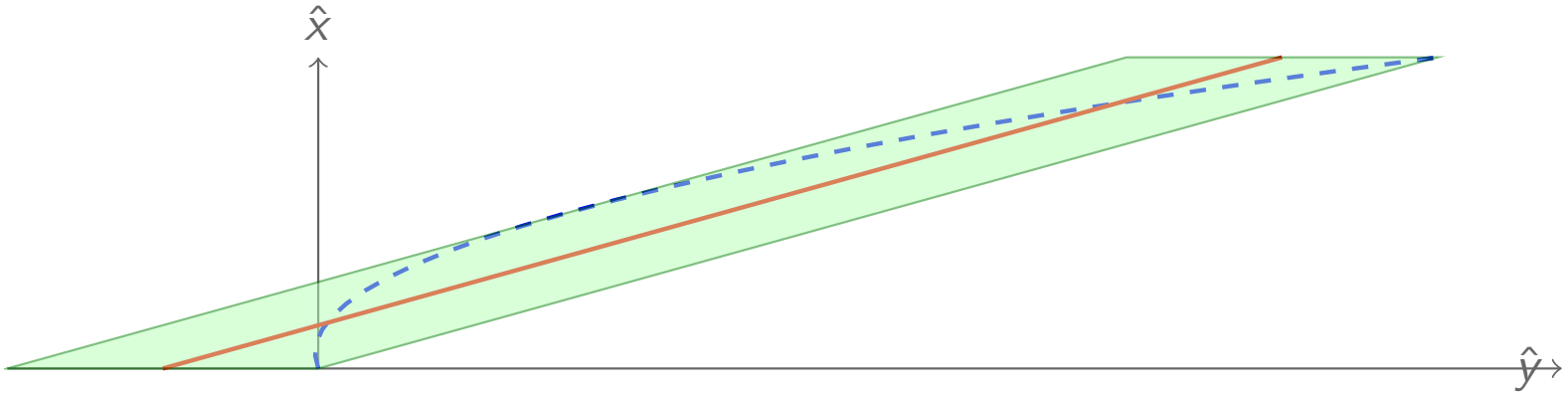
```
real x = [0,10];
real y = 2*x;
if (y >= 10)
  y = x;
```

- Affine forms before tests: $x = 5 + 5\varepsilon_1$, $y = 10 + 10\varepsilon_1$
- In the `if` branch $\varepsilon_1 \geq 0$: condition acts on both x and y

(Minimal) upper bound computation on constrained affine sets is difficult

Remember: example of functional abstraction

```
real x = [0,10];
real y = x*x - x;
```



Abstraction of function $x \to y = x^2 - x$ as

$$\begin{aligned} y &= 32.5 + 45\varepsilon_1 + 12.5\eta_1 \\ &= -12.5 + 9x + 12.5\eta_1 \end{aligned}$$

- Almost modular by construction!
- But valid only for inputs in [0,10] $\Rightarrow$ partition of contexts though a summary-based algorithm.

A summary of a program function is a pair of zonotopes (I, O)

- I abstracts the calling context, O the output
- Both zonotopes abstract functions of the inputs of the program (linear form of the ε_i)
- Output O can be instantiated for a particular calling context $C \subseteq I$ (constraints on the noise symbols that define a restriction of the function)

Summary instantiation: $[\![I == C]\!]O$

- Write the constraint $I == C$ in the space of noise symbols:

$$(c^I_{0i} - c^C_{0i}) + \sum_{r=1}^{n}(c^I_{ri} - c^C_{ri})\varepsilon_r + \sum_{r=1}^{m}(p^I_{ri} - p^C_{ri})\eta_r = 0 \ (i = 1, \ldots, p)$$

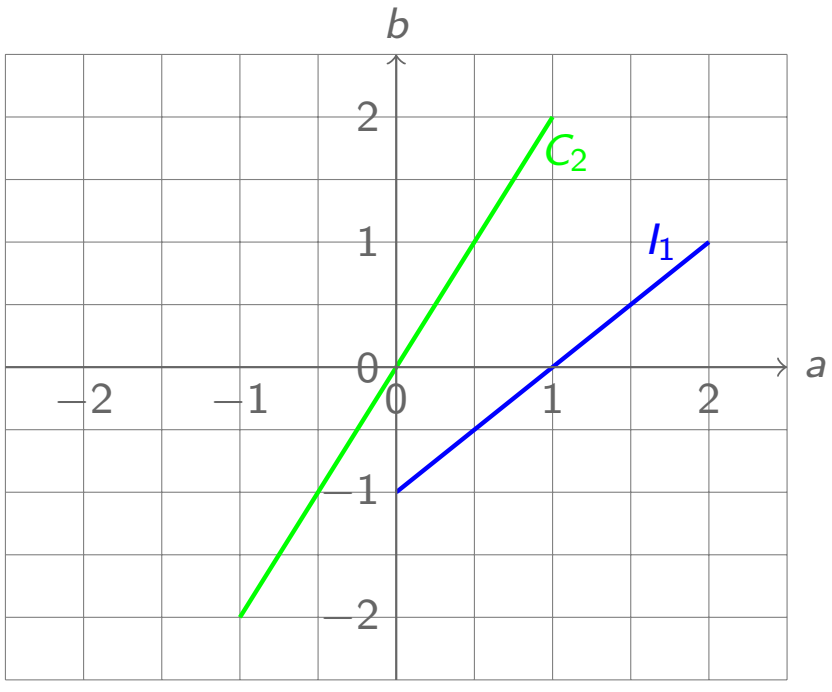
- We derive relations of the form (by Gauss elimination):

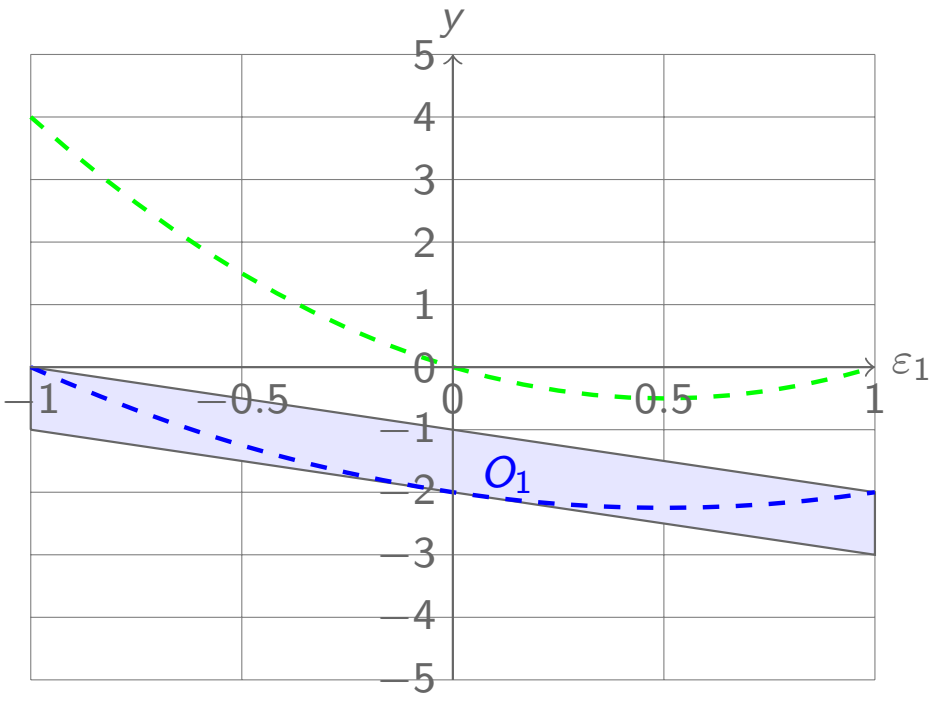
$$\eta_{k_{i+1}} = R_i(\eta_{k_i}, \ldots, \eta_1, \varepsilon_n, \ldots, \varepsilon_1) \ (i = 0, \ldots, r-1)$$

- We eliminate $\eta_{k_1}, \ldots, \eta_{k_r}$ in O using the relations above

Example: summary-based analysis

```
(0) mult(real a, real b) { return a*(b-2); }
(1) x := [-1,1];  // x = eps_1
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

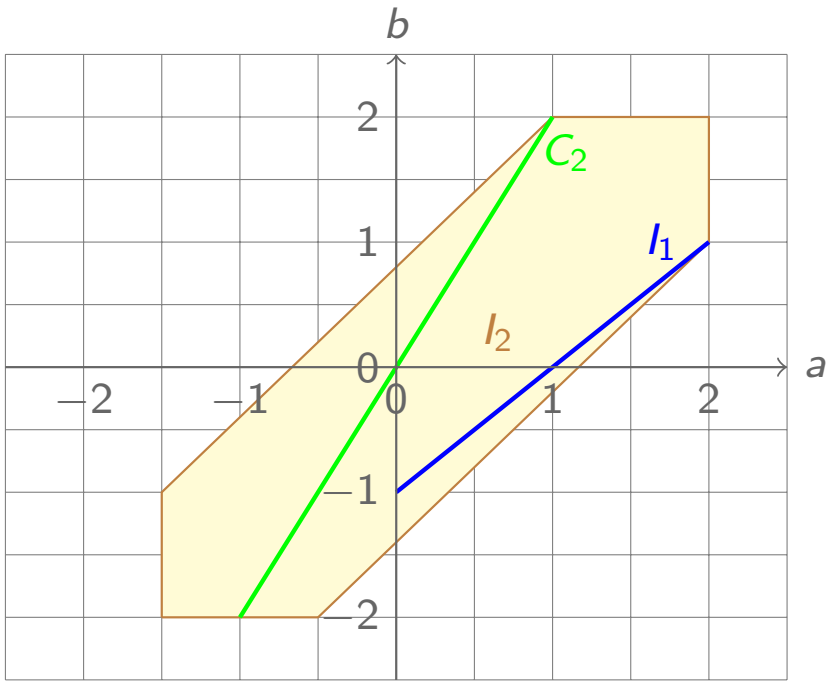


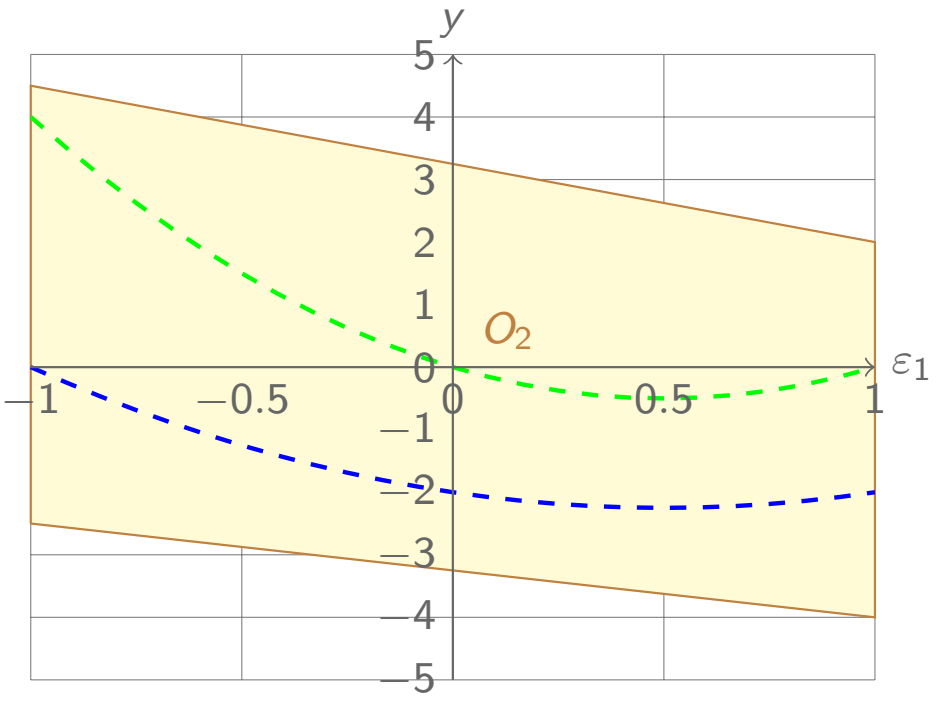


Summary construction

Example: summary-based analysis

```
(0) mult(real a, real b) { return a*(b-2); }
(1) x := [-1,1];  // x = eps_1
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```

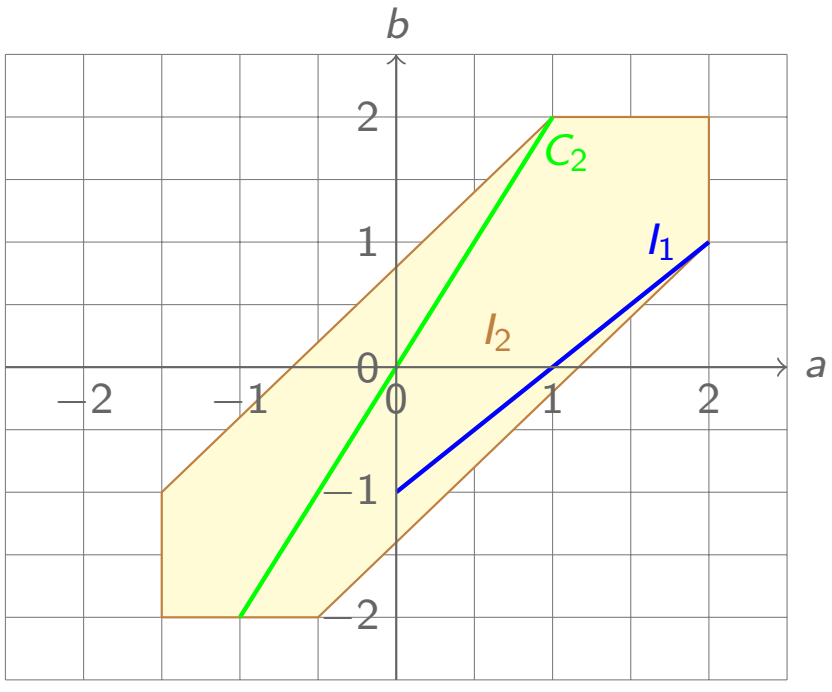


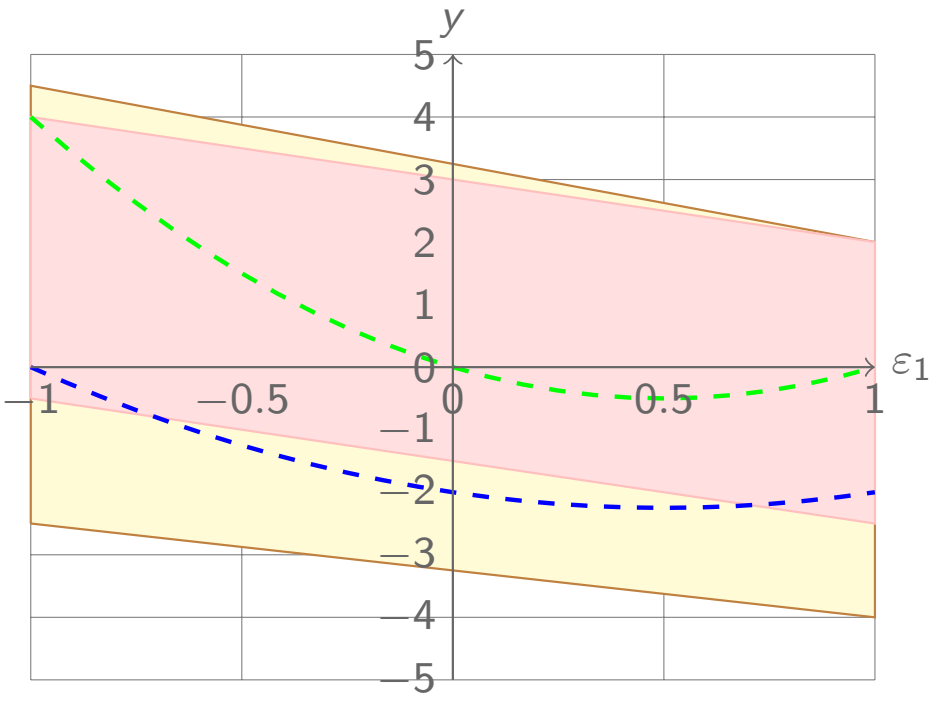


Summary construction

Example: summary-based analysis

```
(0) mult(real a, real b) { return a*(b-2); }
(1) x := [-1,1];  // x = eps_1
(2) real y1 = mult(x+1, x);
(3) real y2 = mult(x, 2*x);
```





Summary instantiation to context C_2

Affine sets: some current and future variations

Keep same parameterization $x = \sum_i x_i \varepsilon_i$ but with

- Interval coefficients x_i: generalized affine sets for under-approximation
  - under-approximation: sets of values of the outputs, that are sure to be reached for some inputs in the specified ranges
  - interval coefficients x_i, noise symbols in generalized intervals ($\varepsilon_\mathbf{i} = [-1, 1]$ or $\varepsilon_\mathbf{i}^* = [1, -1]$), Kaucher arithmetic extends classical interval arithmetic (SAS 2007, with E. Goubault)
  - extensions (higher order forms, under-approximations of discrete values, interval methods of Goldsztejn/Jaulin, application to robust control): O. Mullier's phD thesis (with E. Goubault, M. Kieffer)
- Noise symbols ε_i being no longer defined in intervals:
  - probabilistic affine forms: ε_i take values in p-boxes (Computing 2012, with O. Bouissou, E. Goubault, J. Goubault-Larrecq)
  - ellipsoids (clear potential for program invariants): $\|\varepsilon\|_2 \leq 1$ (instead of $\|\varepsilon\|_\infty \leq 1$) ?

But also, polynomial/rational invariants (extensions of changes of bases by Sankaranarayanan for instance)

Mean-value theorem (à la Goldsztejn 2005)

Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable, $(t_1, \ldots, t_n)$ a point in $[-1,1]^n$ and $\mathbf{\Delta_i}$ such that

$$\left\{ \frac{\partial f}{\partial \varepsilon_i}(\varepsilon_1, \ldots, \varepsilon_i, t_{i+1}, \ldots, t_n),\ \varepsilon_i \in [-1,1] \right\} \subseteq \mathbf{\Delta_i}.$$

Then

$$\tilde{f}(\varepsilon_1, \ldots, \varepsilon_n) = f(t_1, \ldots, t_n) + \sum_{i=1}^{n} \mathbf{\Delta_i}(\varepsilon_i - t_i),$$

is interpretable in the following way :

- if $\tilde{f}(\boldsymbol{\varepsilon_1^*}, \ldots, \boldsymbol{\varepsilon_n^*})$, computed with Kaucher arithmetic, is an improper interval, then pro $\tilde{f}(\boldsymbol{\varepsilon_1^*}, \ldots, \boldsymbol{\varepsilon_n^*})$ is an under-approx of $f(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$.
- $\tilde{f}(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$ is an over-approx of $f(\boldsymbol{\varepsilon_1}, \ldots, \boldsymbol{\varepsilon_n})$.

Generalized affine forms

- Affine forms with interval coefficients, defined on the ε_i (no η_j symbols)
- Under-approximation by over-approximation of dependencies
- Joint use of under-/over-approximation: quality of analysis results
- Extract scenarios giving extreme values

Example

$$f(x) = x^2 - x \text{ when } x \in [2,3] \text{ (real result } [2,6])$$

- Affine form

$$x = 2.5 + 0.5\varepsilon_1, \quad f^{\varepsilon}(\varepsilon_1) = (2.5 + 0.5\varepsilon_1)^2 - (2.5 + 0.5\varepsilon_1)$$

- Bounds on partial derivative

$$\frac{\partial f^{\varepsilon}}{\partial \varepsilon_1}(\varepsilon_1) = 2 * 0.5 * (2.5 + 0.5\varepsilon_1) - 0.5 \subseteq [1.5, 2.5]$$

- Mean value theorem with $t_1 = 0$

$$\tilde{f^{\varepsilon}}(\varepsilon_1) = 3.75 + [1.5, 2.5]\varepsilon_1$$

Under-approximating concretization
$3.75 + [1.5, 2.5][1, -1] = 3.75 + [1.5, -1.5] = [5.25, 2.25]$
Over-approximating concretization
$3.75 + [1.5, 2.5][-1, 1] = 3.75 + [-2.5, 2.5] = [1.25, 6.25]$

- Affine arithmetic (over-approximation)

$$x^2 - x = [3.75, 4] + 2\varepsilon_1 \text{ (concretization } [1.75, 6])$$

Square-root algorithm (Householder method)

```
double Input, x, xp1, residue, shouldbezero;
double EPS = 0.00002;

Input = __BUILTIN_DAED_DBETWEEN(16.0,20.0);
x = 1.0/Input; xp1 = x; residue = 2.0*EPS;
while (fabs(residue) > EPS) {
  xp1 = x*(1.875+Input*x*x*(-1.25+0.375*Input*x*x));
  residue = 2.0*(xp1-x)/(x+xp1);
  x = xp1;
}
shouldbezero = x*x-1.0/Input;
```

- With 32 subdivisions of the input
  - Stopping criterion of the Householder algorithm is satisfied after 5 iterations :

$$[0,0] \subseteq \text{residue}(x_4, x_5) \subseteq [-1.44e^{-5}, 1.44e^{-5}]$$

  - Tight enclosure of the iterate :

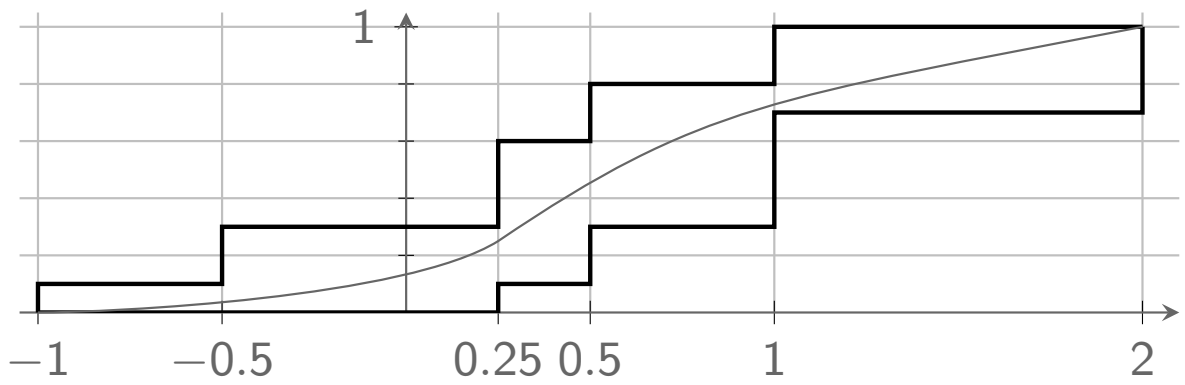
$$[0.22395, 0.24951] \subseteq x_5 \subseteq [0.22360, 0.25000]$$

  - Functional proof :

$$[0,0] \subseteq \text{shouldbezero} \subseteq [-1.49e^{-6}, 1.49e^{-6}]$$

Combining deterministic and probabilistic methods

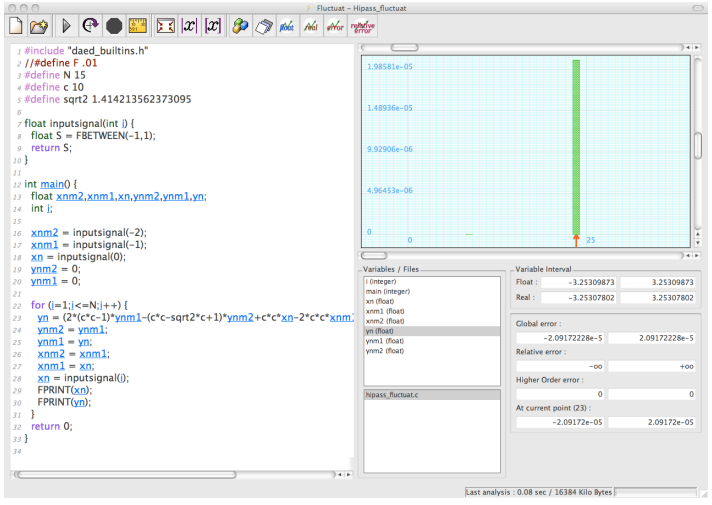
- Discrete p-boxes or Dempster-Shafer structures
  - Generalize probability distributions and interval computations: less pessimistic than intervals but still guaranteed
  - Represent sets of probability distributions: between an upper and a lower Cumulative Distribution Function $P(X \leq x)$

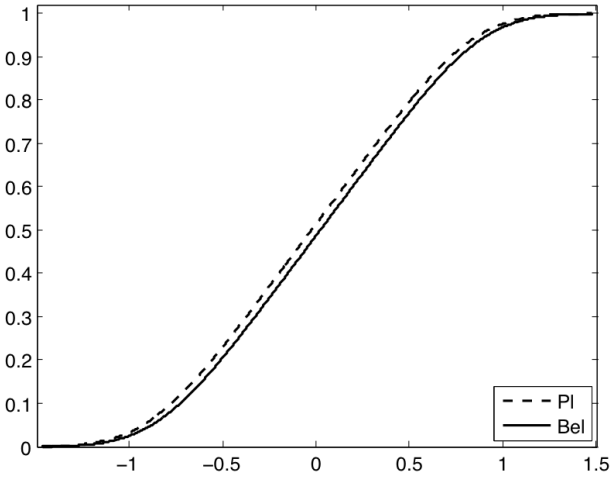


- Encode as much deterministic dependencies as possible by affine arithmetic
  - because arithmetic on p-boxes/DS not very efficient
  - associate a DS structure to each noise symbol
  - both more accurate and faster than direct DS arithmetic:

Example: recursive filter with independent inputs in [-1,1]

Prove that dangerous worst case occur with very low probability





- Deterministic analysis (left): outputs in [-3.25,3.25] (exact)
- Mixed probabilistic/deterministic analysis (right): outputs in [-3.25,3.25], and in [-1,1] with very strong probability

Some current work

- Set operations (extending the deterministic ones)
- Handling dependency information between noise symbols
- Efficient rounding error computation

Validate finite precision implementations

Prove that the program computes something close to what is expected (in real numbers)

- Accuracy of results
- Behaviour of the program (control flow, number of iterations)

Validate algorithms

Bound when possible the method/approximation error

- Check functional properties in real-number semantics

Ideally, find good match between method and implementation errors

Floating-point numbers (defined by the IEEE 754 norm)

- Normalized floating-point numbers

$$(-1)^s 1.x_1 x_2 \ldots x_n \times 2^e \quad \text{(radix 2 in general)}$$

  - implicit 1 convention ($x_0 = 1$)
  - $n = 23$ for simple precision, $n = 52$ for double precision
  - exponent e is an integer represented on k bits ($k = 8$ for simple precision, $k = 11$ for double precision)

- Denormalized numbers (gradual underflow),

$$(-1)^s 0.x_1 x_2 \ldots x_n \times 2^{e_{\min}}$$

- Consequences and difficulties:
  - limited range and precision: potentially inaccurate results, run-time errors
  - no associativity, representation error for harmless-looking reals such as 0.1
  - re-ordering by the compiler, use of registers with different precision, etc

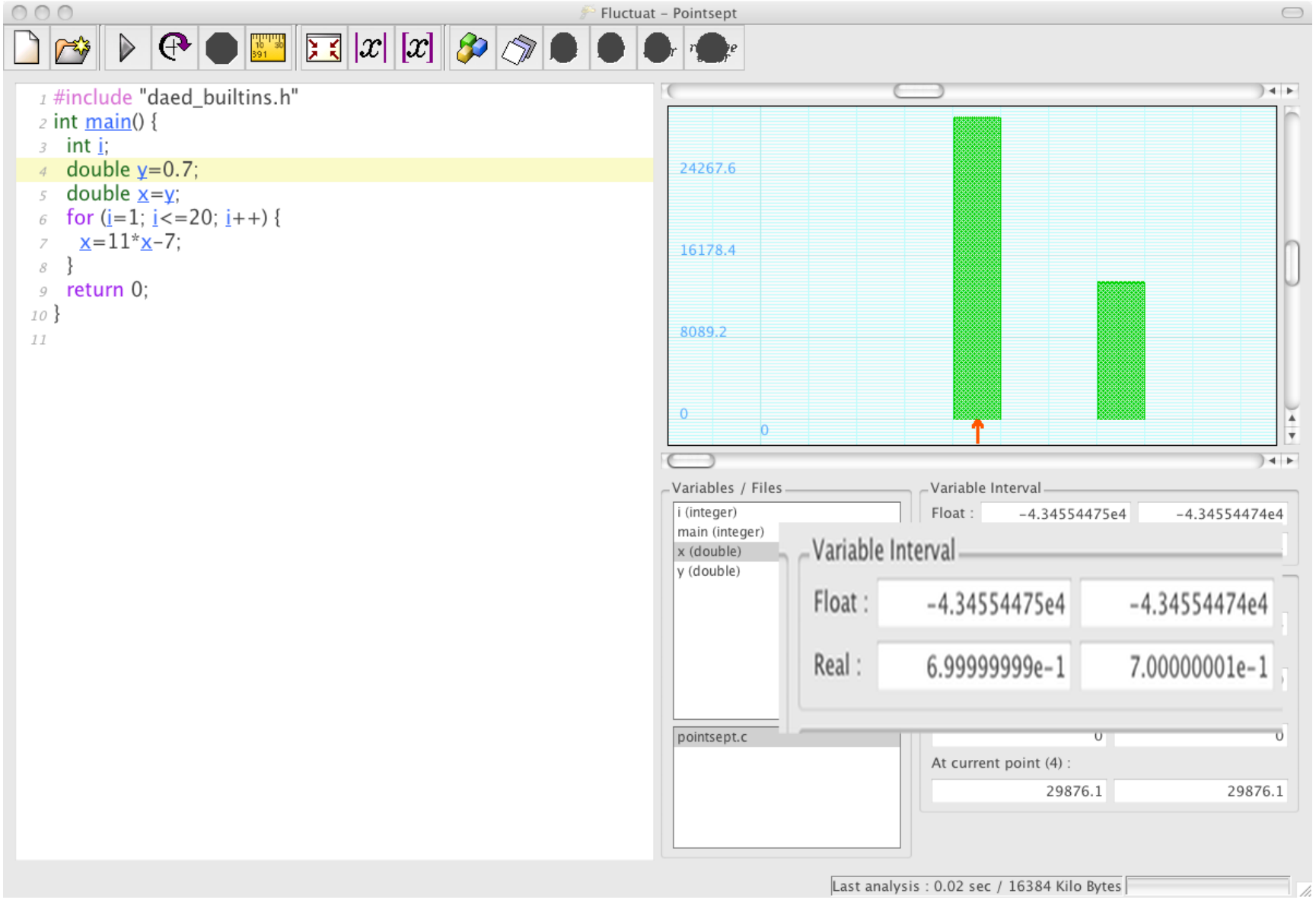
Concrete semantics

- Aim: compute rounding errors and their propagation
  - we need the floating-point values
  - relational (thus accurate) analysis more natural on real values
  - for each variable, we compute (f^x, r^x, e^x)
  - then we will abstract each term (real value and errors)

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

$$f^x = 0.1 + 1.49e^{-9}\ [1]$$
$$f^y = 0.5$$
$$f^z = 0.6 + 1.49e^{-9}\ [1] + 2.23e^{-8}\ [3]$$
$$f^t = 0.06 + 1.04e^{-9}\ [1] + 2.23e^{-9}\ [3] - 8.94e^{-10}\ [4] - 3.55e^{-17}\ [ho]$$

Example (Fluctuat)



Towards an abstract model

IEEE 754 norm on f.p. numbers specifies the rounding error:

- Elementary rounding error when rounding r^x to $\uparrow_\circ r^x$:

$$\exists(\delta_r > 0, \delta_a > 0),\ |r^x - \uparrow_\circ r^x| \leq max(\delta_r |\uparrow_\circ r^x|, \delta_a)$$

- The f.p. result of arithmetic elementary operations $+, -, \times, /, \sqrt{}$ is the rounded value of the real result
  - unit in the Last Place $ulp(x)$ = distance between two consecutive floating-point numbers around x = maximal rounding error around x

$$ulp(1) = 2^{-23} \sim 1.19200928955 * 10^{-7}$$

  - rounding error of elementary operation always less than the ulp of the result
  - also more refined properties, such as Sterbenz lemma (if x and y are two float such that $\frac{y}{2} \leq x \leq y$, then f.p. operation $x - y$ is exact)

Abstraction: for each variable x, a triplet (f^x, r^x, e^x)

Abstract value

- For each variable:
  - Interval $\mathbf{f}^x = [\underline{f^x}, \overline{f^x}]$ bounds the finite prec value, $(\underline{f^x}, \overline{f^x}) \in \mathbb{F} \times \mathbb{F}$,
  - Affine forms for real value and error; for simplicity no η symbols

$$f^x = \underbrace{(\alpha_0^x + \bigoplus_i \alpha_i^x \varepsilon_i^r)}_{\text{real value}} + (\underbrace{e_0^x}_{\text{center of the error}} + \underbrace{\bigoplus_l e_l^x \varepsilon_l^e}_{\text{uncertainty on error due to point } l} + \underbrace{\bigoplus_i m_i^x \varepsilon_i^r}_{\text{propag of uncertainty on value at pt } i})$$

- Constraints on noise symbols (interval + equality constraints)
  - for finite precision control flow
  - for real control flow

Test interpretation

Stable test assumption (old version)

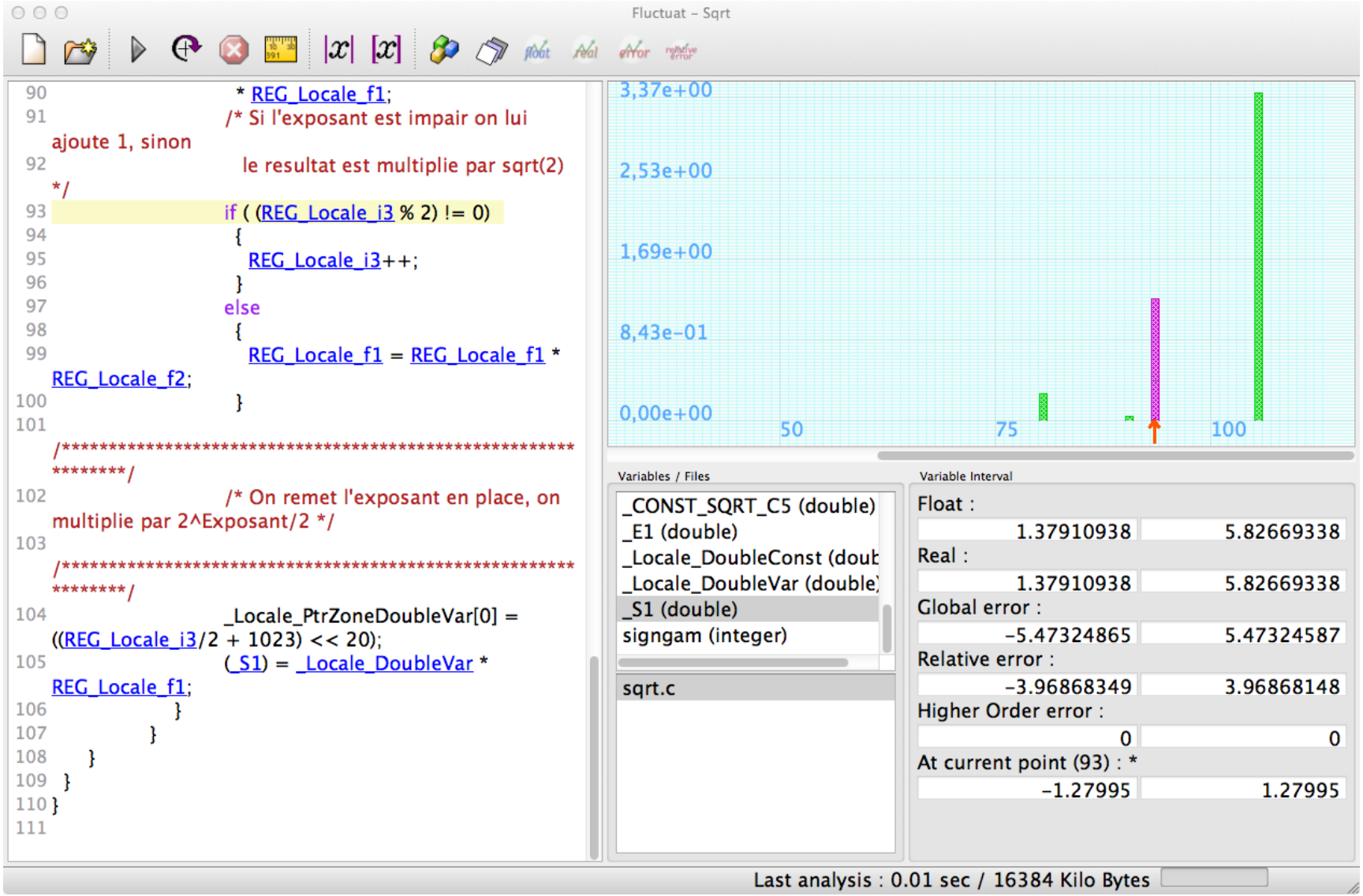
- Control flow of the program is same for the finite precision and real values of the program
- If found not to be the case: unstable test warning
- Joining branches:
  - join values and errors coming from the two branches
  - in case of unstable test, possibly unsound error bounds

Sound unstable test analysis (new version ... November 2012!)

- Tests interpreted over real and float values: two sets of constraints on noise symbols
- Joining branches
  - join fp and real values from the branches
  - errors: join error computed in the two branches with, when it exists (unstable test), the difference between real value in one branch and float value in the other branch for the same execution (ie for same values of the ε_i = intersections of the constraints for these two branches)

In the line of robustness/continuity analysis of Chaudhuri, Gulwani and al.

Test interpretation: example of unstable test



The static analyzer FLUCTUAT

The Fluctuat team

- E. Goubault, S. Putot (2001-2012), M. Martel (2001-2005, user interface and Fluctuat Assembler), F. Védrine (2008-2012), K. Tekkal (2008-2011, Digiteo OMTE then start-up incubation), T. Le Gall (2012)
- Continuous support by Airbus and IRSN, more occasional by other users
  - Is/has been used for a wide variety of codes (automotive, nuclear industry, aeronautics, aerospace) of size up to about 50000 LOCs

Assertions in the program analyzed

- Hypotheses on the environment of the program
- Local partitioning and collecting strategies

Exploitation of results

- Warnings: unstable tests, run-time errors
- Identifying problems and refining results: worst case scenarios, symbolic execution, subdivisions
- Library and new interactive version (F. Védrine)

Second order filters

Back to the Householder scheme

Around FLUCTUAT

- First extension to the analysis of hybrid systems by interaction with the ODE guaranteed solver GrkLib (HybridFluctuat, CAV 2009, with O. Bouissou, E. Goubault, K. Tekkal, F. Védrine)
- First interface with Esterel's SCADE (with library version of Fluctuat)
- (Present/Future) Interaction with constraint solvers
  - first demonstration of refinement of Fluctuat's result (Ponsini, Michel, Rueher 2011-2012)
  - natural for us because increasing use of constraints on noise symbols (and we could provide more for non linear operations)
- (Present/Future) Floating-point to fixed point automatic conversion (ANR DEFIS project 2012-2015, first steps in 2005 with intern J. Mascunan)
- (Future) Interaction with provers, Frama-C platform
  - ACSL language to exchange information on real, float and errors
  - provide provers with loop invariants
  - use locally refined results and properties
  - towards formally proved abstract domain implementations ? (cf D. Pichardie)

Conclusion and Perpectives

Some highlights in the near future

- Under-approximations (O. Mullier's PhD thesis)
- Fixpoint computations and zonotopes (if possible in Fluctuat): policy iteration, mix with ellipsoids ?

Keep the line of numerically-oriented static analysis

- Natural application is industrial safety-critical control-command software
  - needs driven by the users (modularity, interaction with other tools etc)
- Extension to safety-critical control systems (medical instrumentation?)
- Long term goal: programs from scientific computing
  - as a complement to existing sensitivity and uncertainty propagation analyses (Chaos polynomial, Cestac method)
  - specific problems (large arrays)
  - parallelism and reordering of arithmetic operations
  - link with mathematical studies of schemes in finite precision (Wilkinson, Meurant, Demmel etc), view as perturbed schemes