DE LA RECHERCHE À L'INDUSTRIE

Robustness analysis of finite precision implementations

Eric Goubault and Sylvie Putot

LMeASI, CEA LIST | APLAS Conference, Melbourne, December 9, 2013

www.cea.fr

INSTITUT CARNOT CEA LIST

digiteo

list

Motivations

Context: automatic validation of numerical programs

- Infer invariant properties both in floating-point and real number semantics
  - Abstract interpretation based static analysis (affine arithmetic/zonotopes)
- Validate finite precision implementations: prove the program computes something close to expected (in real numbers)
  - Bound and propagate rounding errors: accuracy of computations
  - Behaviour of the program (control flow, number of iterations)
- Implemented in our abstract interpreter FLUCTUAT

A difficulty in error analysis: unstable tests

- When finite precision and real control flows are potentially different
- If discontinuity of treatment between branches, error analysis is unsound
- When considering sets of executions, most tests are potentially unstable

We propose here to compute discontinuity errors in unstable tests

- Makes our error analysis sound in presence of unstable tests
- Provides a robustness analysis

Linked to program robustness & continuity analysis

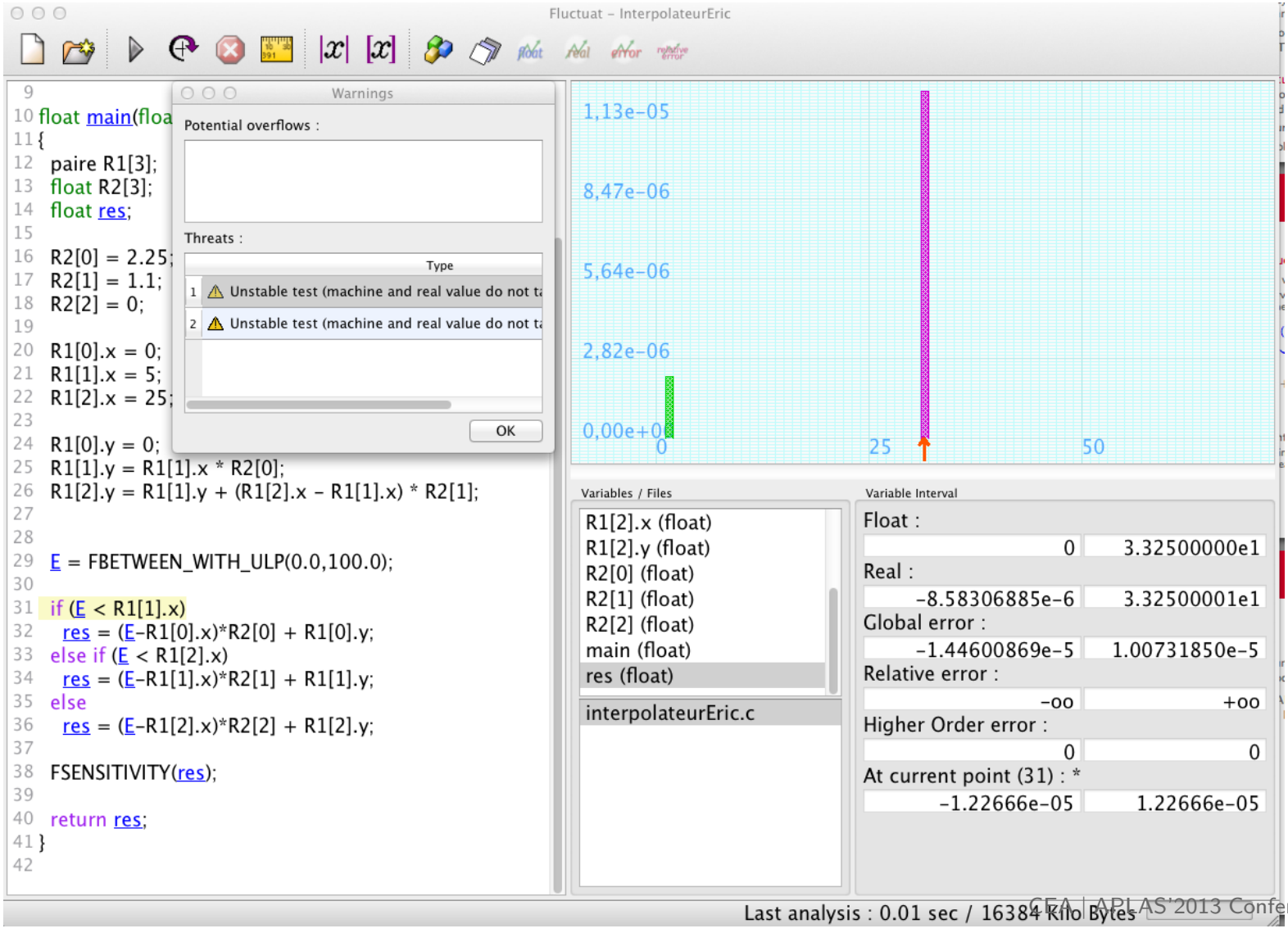
Can small uncertainty on inputs cause only small perturbations on the outputs (with different execution paths): notions classical for control systems but not so much for software implementations.

Some recent work in critical embedded sofware:

- Some real cases (cf NASA engineer Bushnell's pres. at NSV 2011 on the F22 raptor crossing int. date line in 2007)
- Continuity in Software Systems [Hamlet 2002]
- Continuity analysis and robustness of programs [Chaudhuri, Gulwani, Lublineramn 2010-2012]
- Robust software synthesis, Symbolic robustness analysis, etc [Majumdar, Render, Tabuada, Saha, 2009-2012]

A typical example of unstable tests: affine interpolators

All tests are unstable, but the implementation is robust, the conditional block does not introduce a discontinuity



But discontinuities also actually occur (sqrt approx.)

```
#define sqrt2 1.41421353816986083984375
double x, y;  x = DREAL_WITH_ERROR(1,2,0,0.001);
if (x>2)
  y = sqrt2*(1+(x/2-1)*(.5-0.125*(x/2-1)));
else
  y = 1+(x-1)*(.5+(x-1)*(-.125+(x-1)*.0625));
```

Without unstable test analysis, unsound results in Fluctuat:

- An unstable test is signalled at the `if` statement
- `y` has real value in [1,1.4531] with an error in [-0.0005312,0.00008592]

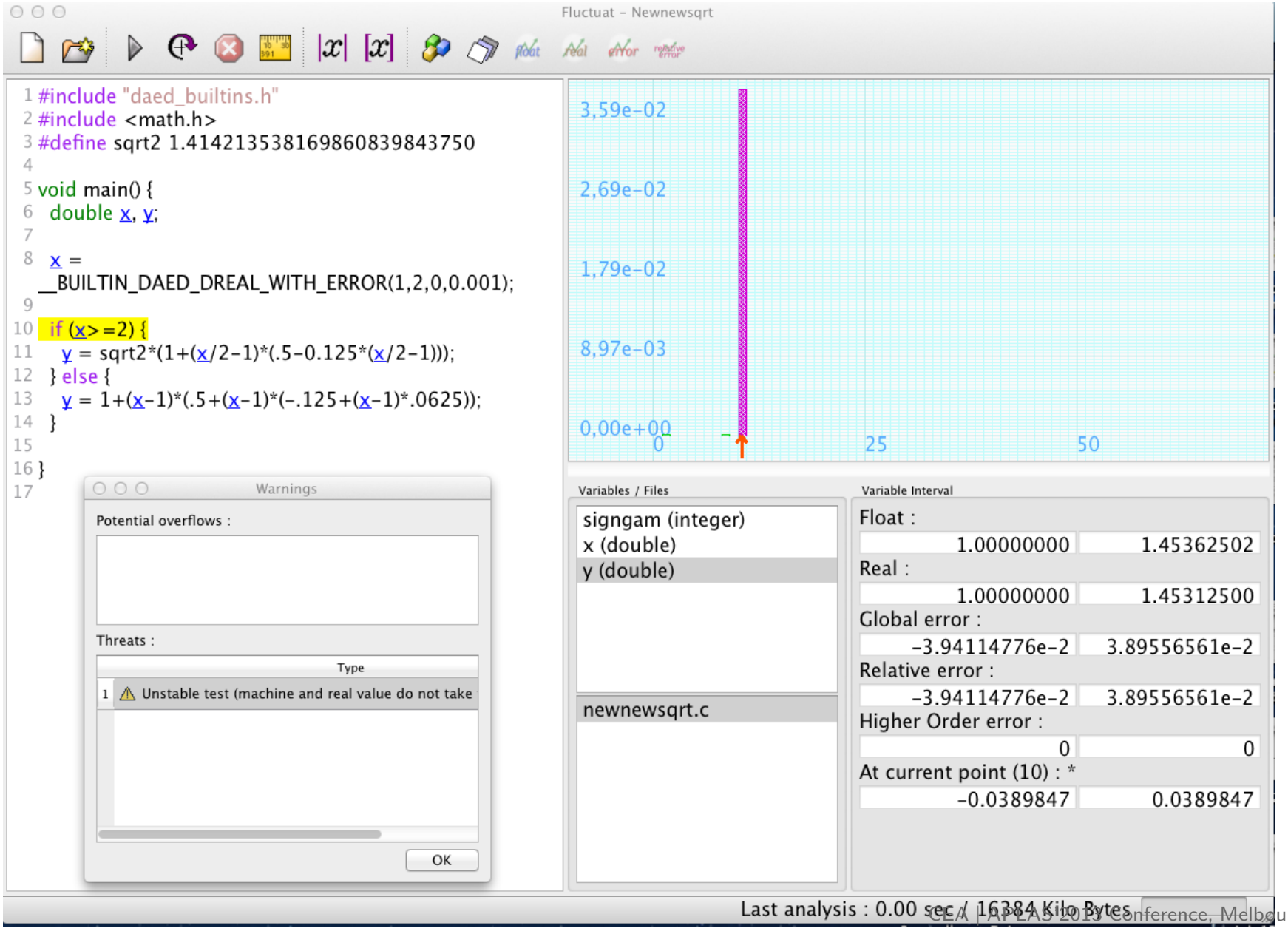
Unstable test: consider for instance $r^x = 2$ and $f^x = 2 + 0.001$

- execution in reals ($r^x = 2$) takes the `else` branch: $r^y = 1.4375$,
- execution in floats ($f^x = 2 + 0.001$) takes the `then` branch: $f^y = 1.4145...$

The test introduces a discontinuity $f^y - r^y = -0.023$ around the test condition ($x == 2$): larger than the error bounds

- want to consider discontinuity as a new error term; accurate abstraction ?

With unstable test analysis



Outline of the talk

- Abstraction of real and finite precision computations in FLUCTUAT
  - Affine arithmetic and zonotopes
  - Extension to the analysis of finite precision implementations
- Test interpretation, unstable test / robustness analysis

Affine Arithmetic (Comba & Stolfi 93)

Affine forms and affine arithmetic

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

where the ε_i are symbolic variables (*noise symbols*), with value in $[-1, 1]$.

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

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

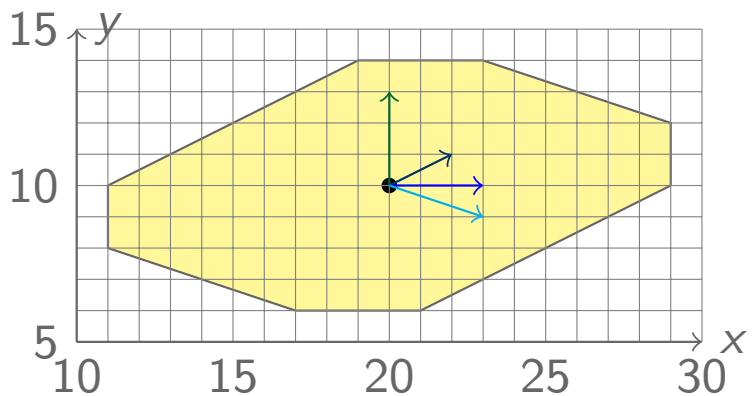
- *Addition/subtraction* are exact:

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

- *Non linear operations* : approximate linear form, new noise term bounding the approximation error (close to Taylor models of low degree)

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

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



Test interpretation

Main idea to interpret test, informally

- Affine forms are unchanged, translate the condition on noise symbols
- Equality tests are interpreted by the substitution of one noise symbol of the constraint

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

Functional interpretation

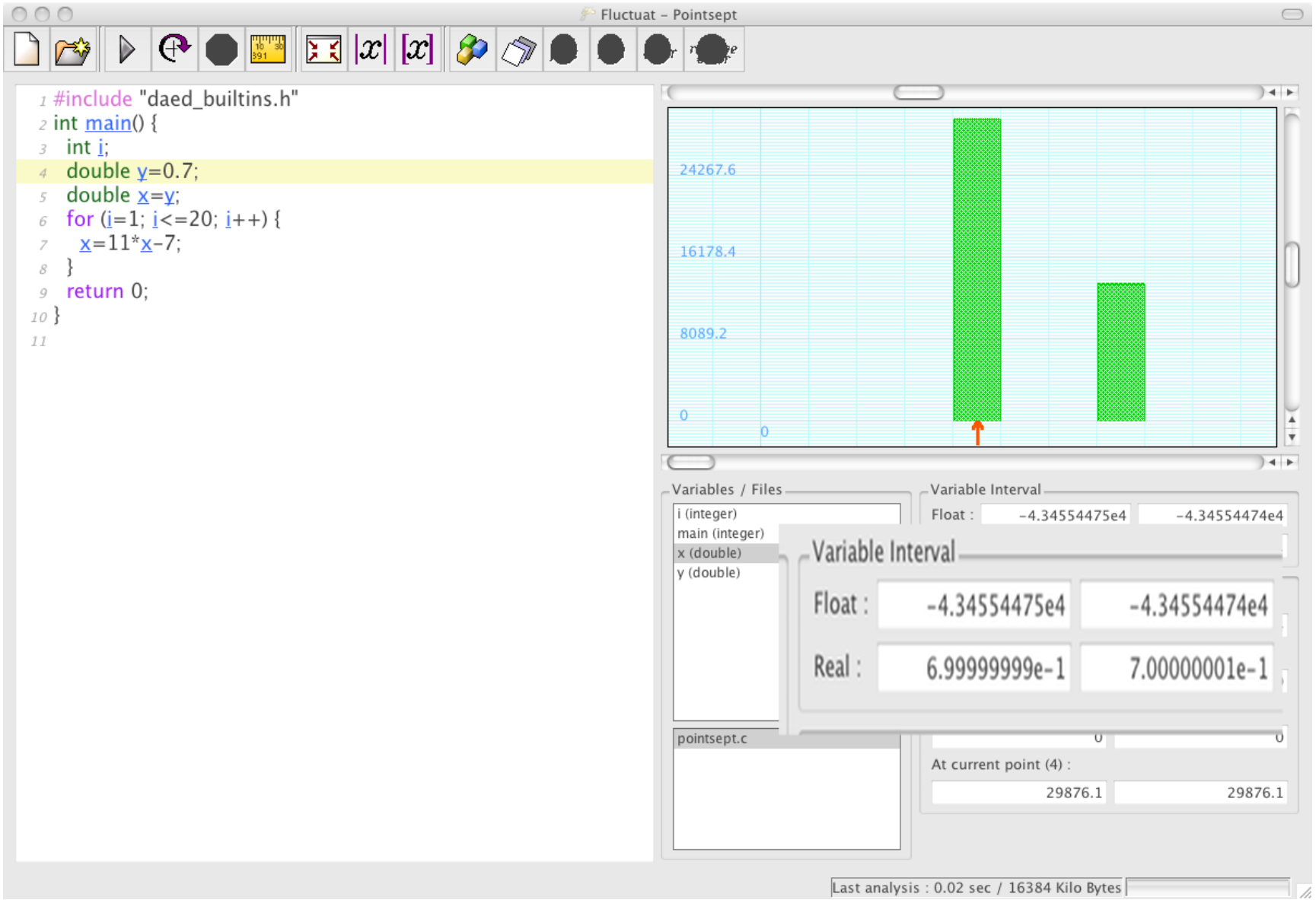
The test condition leads to a condition on the noise symbols ε_i, that can be interpreted as a restriction to the set of inputs that can lead to an execution satisfying the test condition.

- Aim: compute rounding errors and their propagation
  - we need the real and floating-point 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]
```

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

Example (Fluctuat)



Abstract domain in Fluctuat

Abstract value at each control point c

- For each variable, affine forms for real value and error:

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

- Constraints on noise symbols coming from interpretation of test condition
  - $\varepsilon^r \in \Phi_r^X$ for real control flow (test on the r^x: constraints on the ε_i^r)
  - $(\varepsilon^r, \varepsilon^e) \in \Phi_f^X$ for finite precision control flow (test on the $f^x = r^x + e^x$: constraints on the ε_i^r and ε_l^e)

Unstable test condition = intersection of constraints $\varepsilon^r \in \Phi_r^X \sqcap \Phi_f^Y$:

- unstable test: *for a same execution (same values of the noise symbols ε_i)* the control flow is different
- restricts the range of the ε_i: allows us to bound accurately the discontinuity error

Formally, sound abstraction (with discontinuity errors)

Abstract value

An abstract value X, for a program with p variables $x_1, \ldots, x_p$, is a tuple $X = (R^X, E^X, D^X, \Phi_r^X, \Phi_f^X)$ composed of the following affine sets and constraints, for all $k = 1, \ldots, p$:

$$\left\{ \begin{array}{lcll} R^X : \hat{r}_k^X & = & r_{0,k}^X + \sum_{i=1}^n r_{i,k}^X \, \varepsilon_i^r & \text{where } \varepsilon^r \in \Phi_r^X \\ E^X : \hat{e}_k^X & = & e_{0,k}^X + \sum_{i=1}^n e_{i,k}^X \, \varepsilon_i^r + \sum_{j=1}^m e_{n+j,k}^X \, \varepsilon_j^e & \text{where } (\varepsilon^r, \varepsilon^e) \in \Phi_f^X \\ D^X : \hat{d}_k^X & = & d_{0,k}^X + \sum_{i=1}^o d_{i,k}^X \, \varepsilon_i^d & \\ \hat{f}_k^X & = & \hat{r}_k^X + \hat{e}_k^X & \text{where } (\varepsilon^r, \varepsilon^e) \in \Phi_f^X \end{array} \right.$$

E^X is the propagated rounding error, D^X the propagated discontinuity error

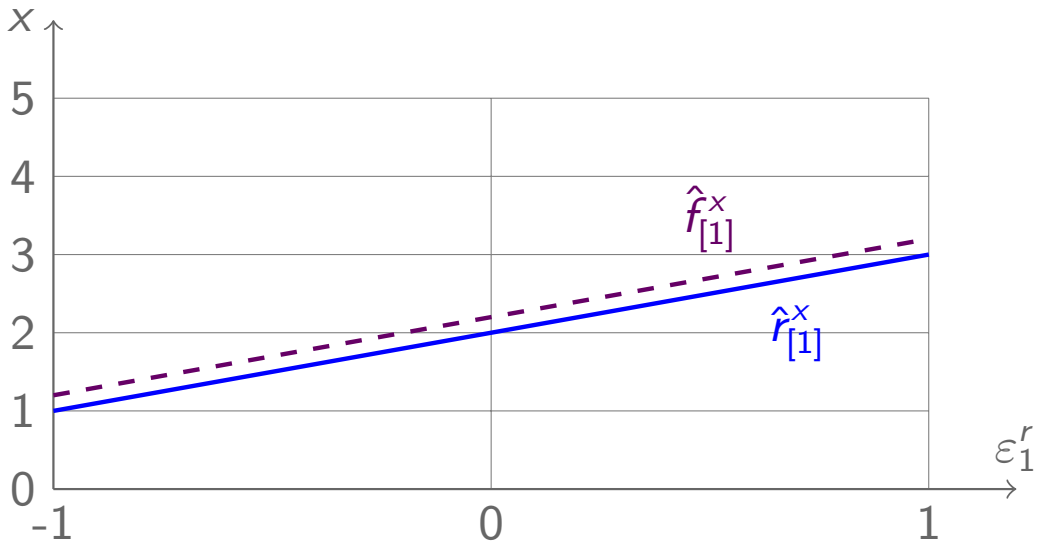
New discontinuity errors computed when joining branches of a possibly unstable test

$Z = X \sqcup Y$ is $Z = (R^Z, E^Z, D^Z, \Phi_r^X \cup \Phi_r^Y, \Phi_f^X \cup \Phi_f^Y)$ such that

$$\left\{ \begin{array}{l} (R^Z, \Phi_r^Z \cup \Phi_f^Z) = (R^X, \Phi_r^X \cup \Phi_f^X) \sqcup (R^Y, \Phi_r^Y \cup \Phi_f^Y) \\ (E^Z, \Phi_f^Z) = (E^X, \Phi_f^X) \sqcup (E^Y, \Phi_f^Y) \\ D^Z = D^X \sqcup D^Y \sqcup (R^X - R^Y, \Phi_f^X \sqcap \Phi_r^Y) \sqcup (R^Y - R^X, \Phi_f^Y \sqcap \Phi_r^X) \end{array} \right.$$

Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```



At cpt 1: $\hat{r}^x_{[1]} = 2 + \varepsilon^r_1$; $\hat{e}^x_{[1]} = u$

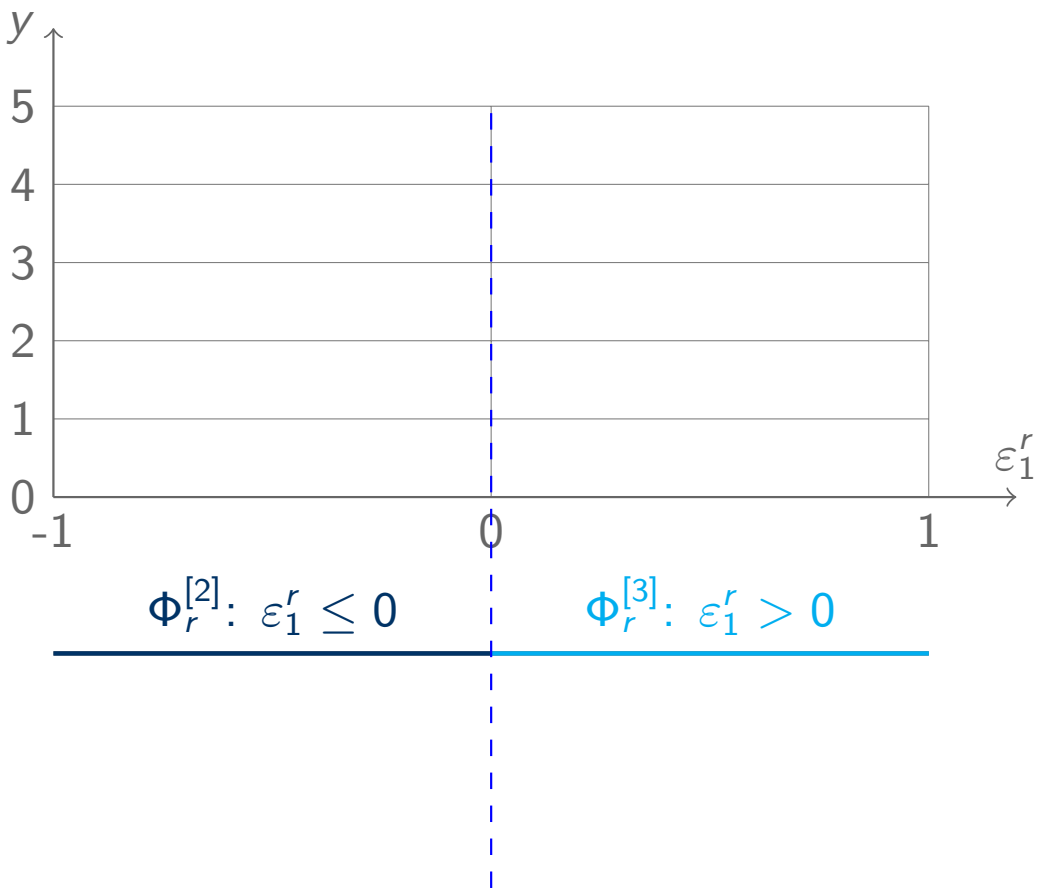
Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

$\hat{r}^x_{[1]} = 2 + \varepsilon^r_1;\ \hat{e}^x_{[1]} = u$

Test $x \leq 2$, real flow:

$\Phi^{[2]}_r : \hat{r}^x_{[1]} = 2 + \varepsilon^r_1 \leq 2$



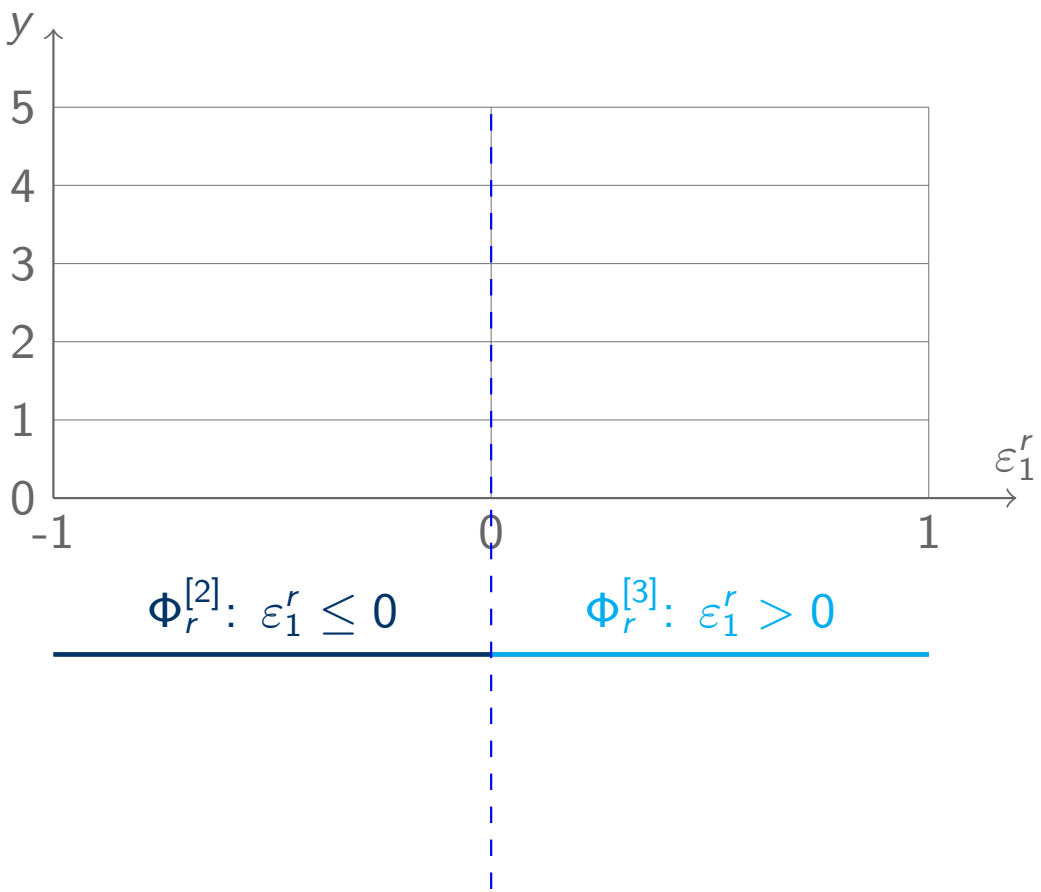
Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

$\hat{r}^x_{[1]} = 2 + \varepsilon^r_1;\ \hat{e}^x_{[1]} = u$

Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$

Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$



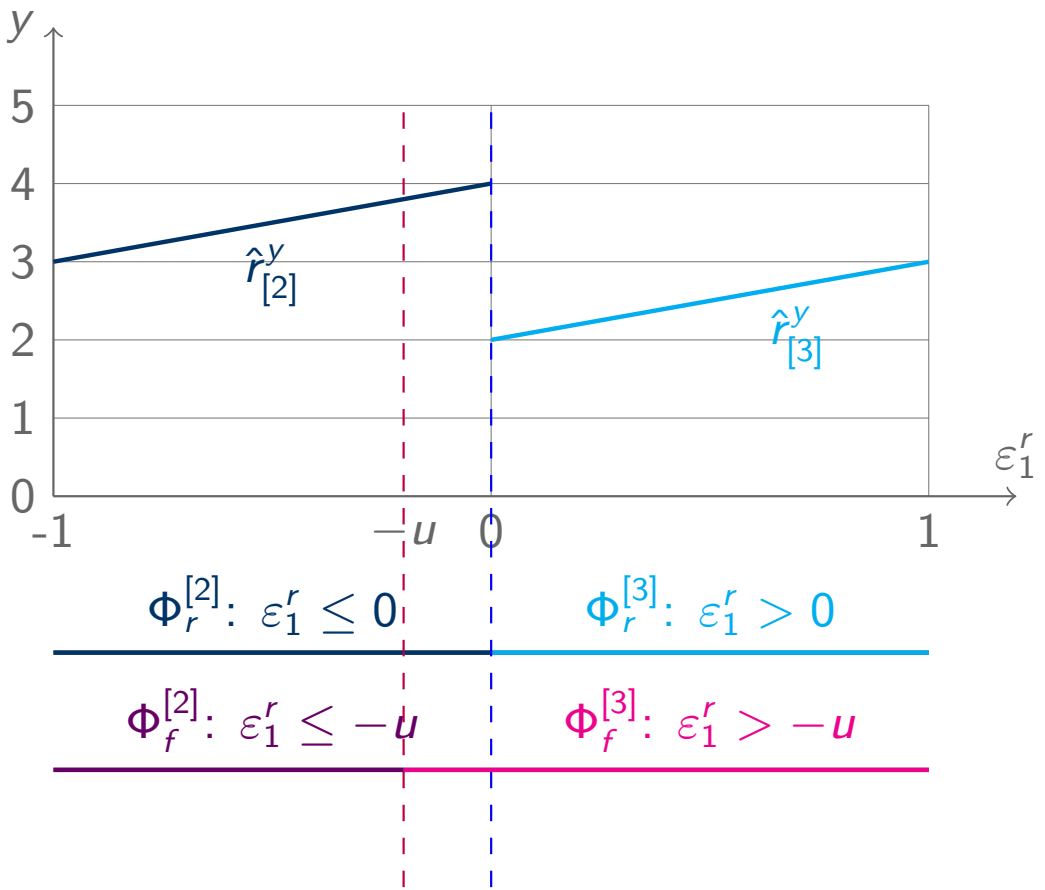
Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

$\hat{r}^x_{[1]} = 2 + \varepsilon^r_1;\ \hat{e}^x_{[1]} = u$

Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

Test $x \leq 2$, float flow:
$\Phi^{[2]}_f : \hat{r}^x_{[1]} + \hat{e}^x_{[1]} = 2 + \varepsilon^r_1 + u \leq 2$



Example: sound unstable test analysis

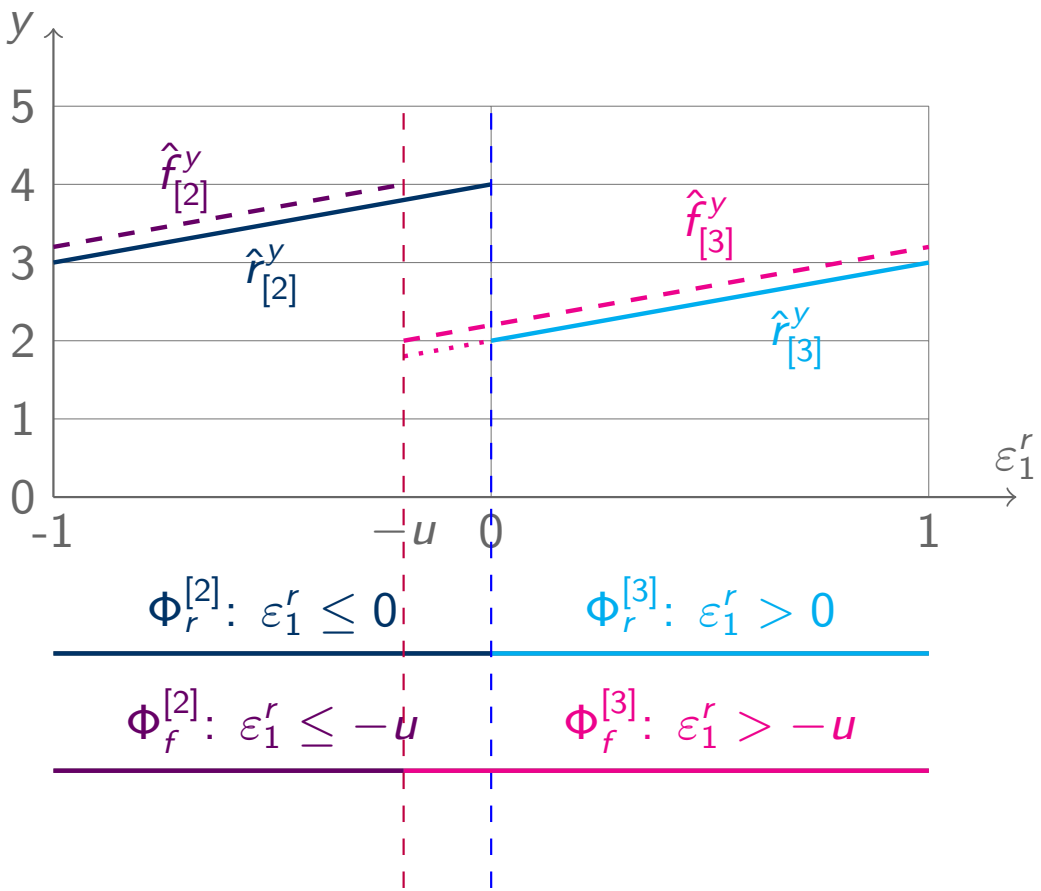
```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$

Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

Error at [2]: $\hat{e}^y_{[2]} = \hat{e}^x_{[1]} + \delta\varepsilon^e_2$

Error at [3]: $\hat{e}^y_{[3]} = \hat{e}^x_{[1]}$



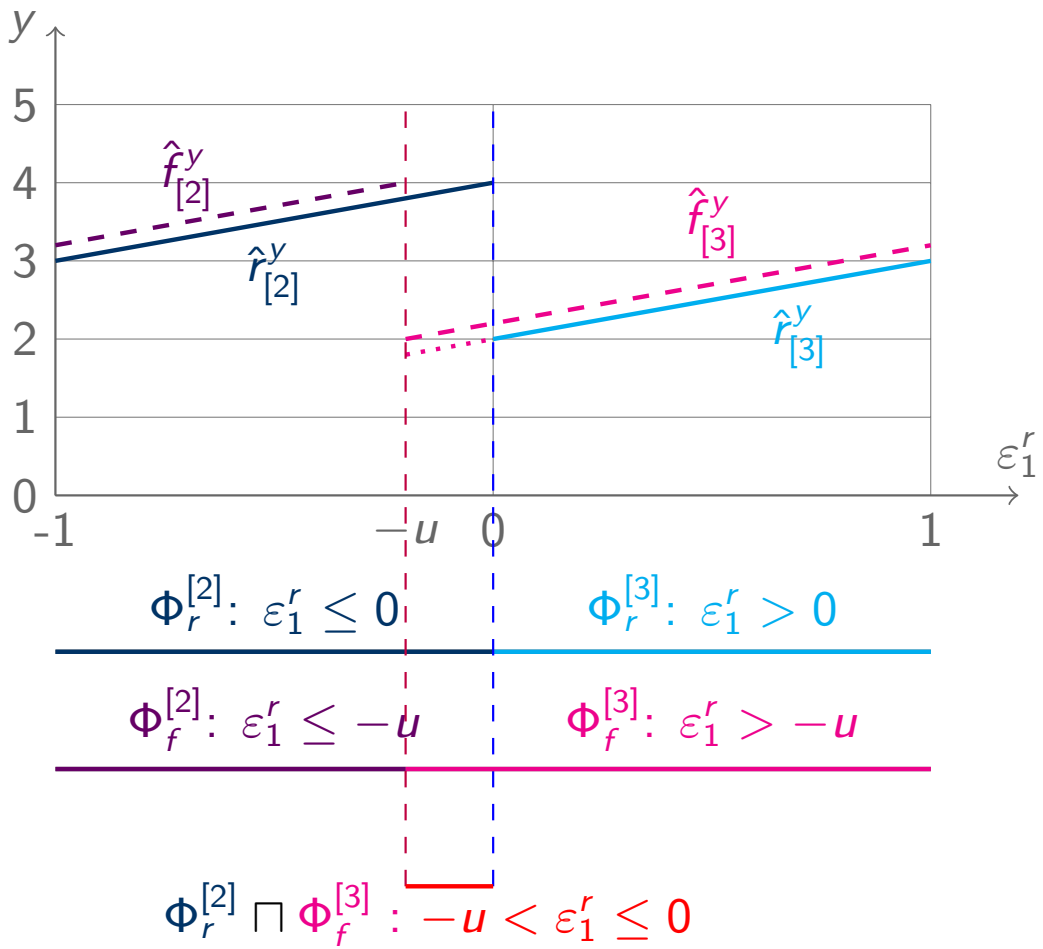
Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

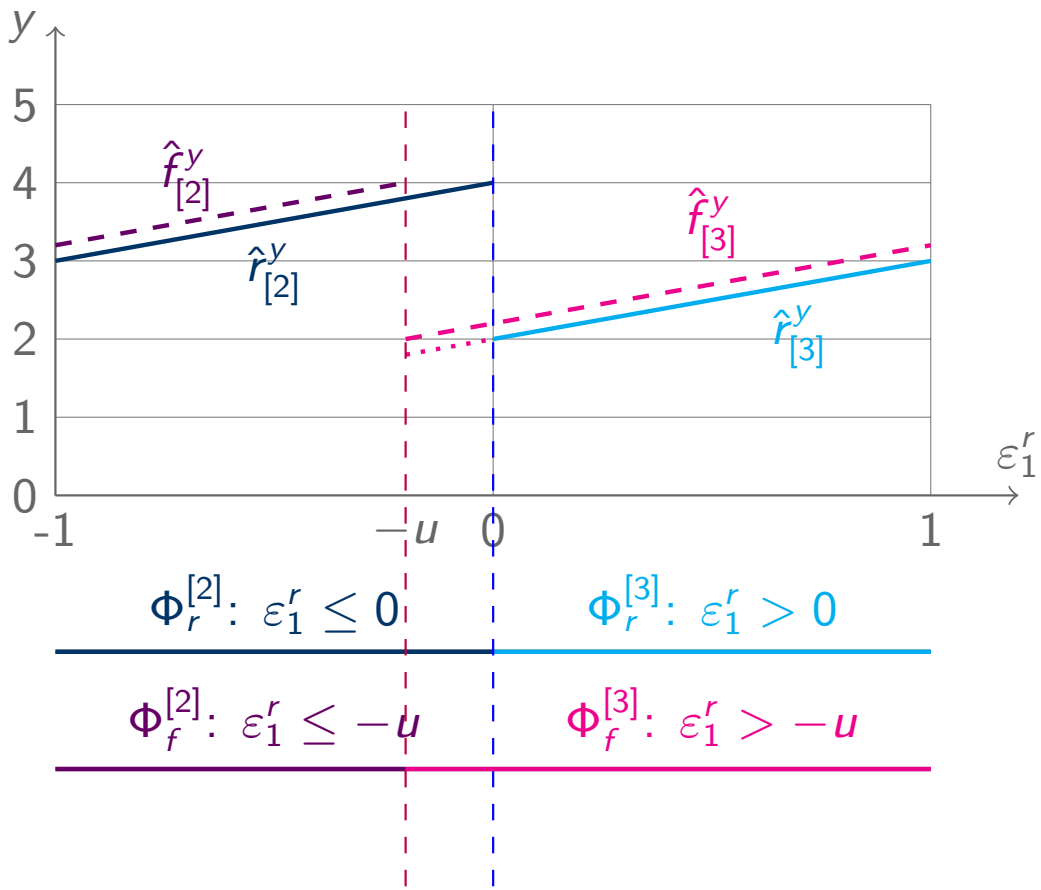
Error at [2]: $\hat{e}^y_{[2]} = u + \delta\varepsilon^e_2$
Error at [3]: $\hat{e}^y_{[3]} = u$

Unstable test, first possibility:
$\Phi^{[2]}_r \sqcap \Phi^{[3]}_f : -u < \varepsilon^r_1 \leq 0$



Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```



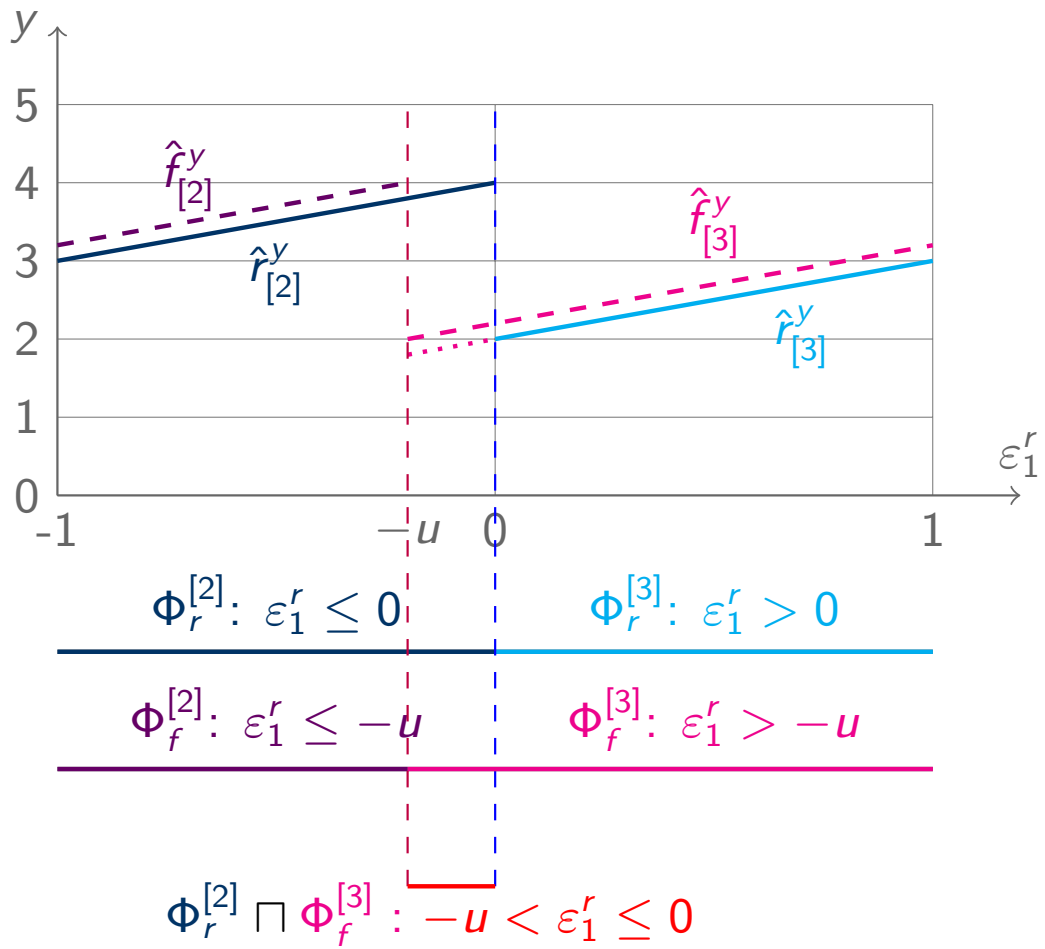
Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon_1^r, \Phi_r^{[2]})$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon_1^r, \Phi_r^{[3]})$

Error at [2]: $\hat{e}^y_{[2]} = u + \delta\varepsilon_2^e$
Error at [3]: $\hat{e}^y_{[3]} = u$

Unstable test, second possibility:
$\Phi_r^{[3]} \sqcap \Phi_f^{[2]} = \emptyset$

Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```



Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon_1^r, \Phi_r^{[2]})$

Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon_1^r, \Phi_r^{[3]})$

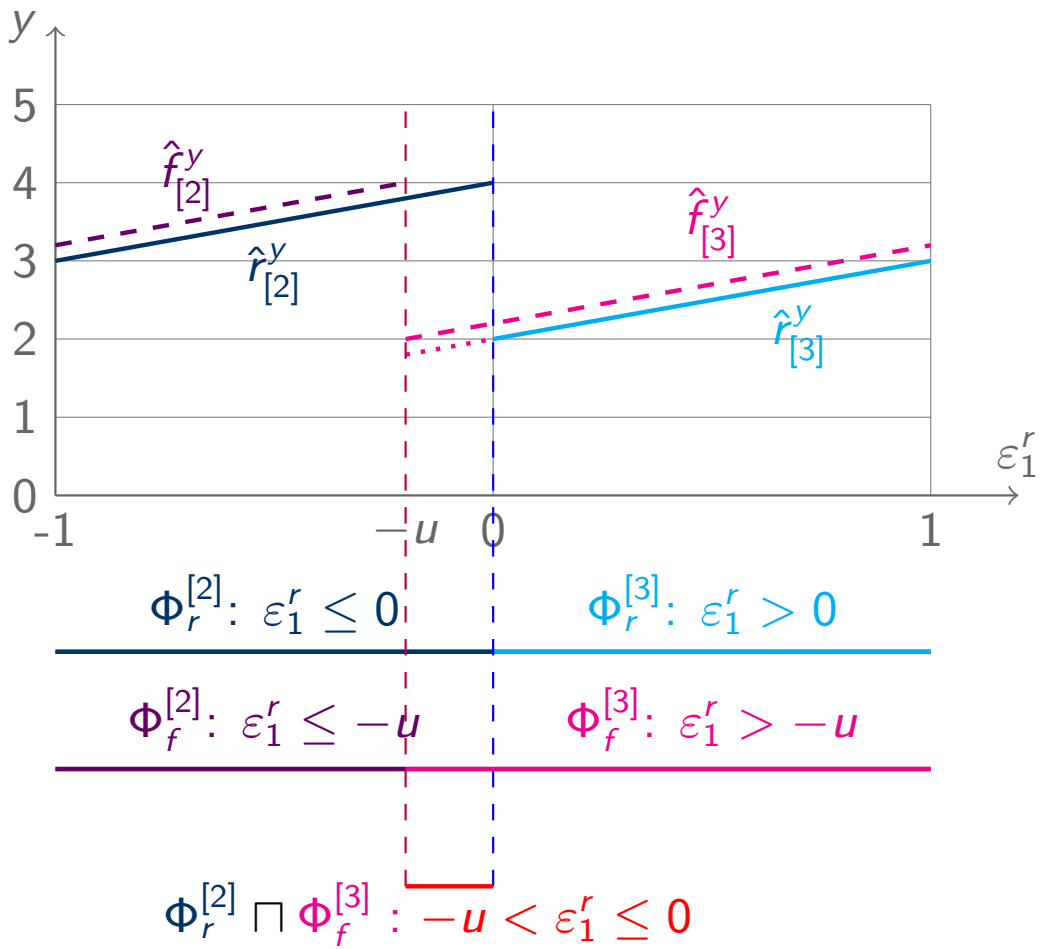
Error at [2]: $\hat{e}^y_{[2]} = u + \delta\varepsilon_2^e$

Error at [3]: $\hat{e}^y_{[3]} = u$

Error at [4] : $\hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]} \sqcup (\underbrace{\hat{f}^y_{[3]} - \hat{r}^y_{[2]}}_{\hat{r}^y_{[3]} + \hat{e}^y_{[3]} - \hat{r}^y_{[2]}}, \Phi_f^{[3]} \sqcap \Phi_r^{[2]}) = \hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]} + (\hat{r}^y_{[3]} - \hat{r}^y_{[2]}, \Phi_f^{[3]} \sqcap \Phi_r^{[2]})$

Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```



Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon_1^r, \Phi_r^{[2]})$

Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon_1^r, \Phi_r^{[3]})$

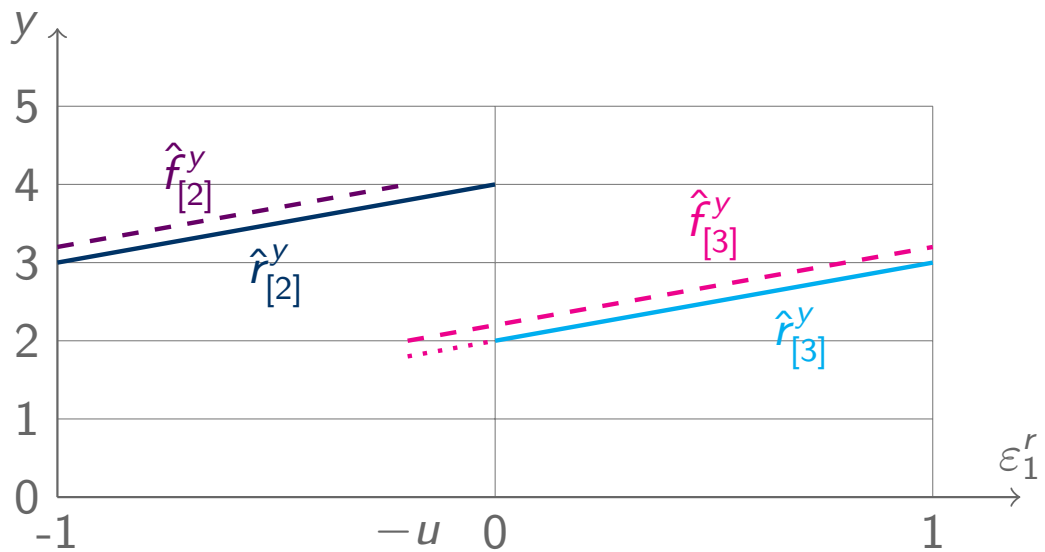
Error at [2]: $\hat{e}^y_{[2]} = u + \delta\varepsilon_2^e$

Error at [3]: $\hat{e}^y_{[3]} = u$

Error at [4] : $\underbrace{\hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]}}_{\hat{e}^y_{[4]}} + \underbrace{(\hat{r}^y_{[3]} - \hat{r}^y_{[2]}, \Phi_f^{[3]} \sqcap \Phi_r^{[2]})}_{\hat{d}^y_{[4]}} = u + \delta\varepsilon_2^e - 2\chi_{[-u,0]}(\varepsilon_1^r)$

Example: sound unstable test analysis

```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```



Error at [4] : $\underbrace{\hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]}}_{\hat{e}^y_{[4]}} + \underbrace{(\hat{r}^y_{[3]} - \hat{r}^y_{[2]}, \Phi_f^{[3]} \sqcap \Phi_r^{[2]})}_{\hat{d}^y_{[4]}} = \underbrace{u + \delta\varepsilon_2^e}_{\hat{e}^y_{[4]}} + \underbrace{-2\chi_{[-u,0]}(\varepsilon_1^r)}_{\hat{d}^y_{[4]}}$

Real value $\hat{r}^y_{[4]} = 3 + \varepsilon_4^r \in [2, 4]$

Float value $\hat{f}^y_{[4]} = \hat{r}^y_{[4]} + \hat{e}^y_{[4]} = 3 + \varepsilon_4^r + u + \delta\varepsilon_2^e \in [2 + u - \delta, 4 + u + \delta]$

Back to the definitions

Abstract value

An abstract value X, for a program with p variables $x_1, \ldots, x_p$, is a tuple $X = (R^X, E^X, D^X, \Phi_r^X, \Phi_f^X)$ composed of the following affine sets and constraints, for all $k = 1, \ldots, p$:

$$\left\{ \begin{array}{lcll} R^X : \hat{r}_k^X & = & r_{0,k}^X + \sum_{i=1}^{n} r_{i,k}^X \, \varepsilon_i^r & \text{where } \varepsilon^r \in \Phi_r^X \\ E^X : \hat{e}_k^X & = & e_{0,k}^X + \sum_{i=1}^{n} e_{i,k}^X \, \varepsilon_i^r + \sum_{j=1}^{m} e_{n+j,k}^X \, \varepsilon_j^e & \text{where } (\varepsilon^r, \varepsilon^e) \in \Phi_f^X \\ D^X : \hat{d}_k^X & = & d_{0,k}^X + \sum_{i=1}^{o} d_{i,k}^X \, \varepsilon_i^d & \\ \hat{f}_k^X & = & \hat{r}_k^X + \hat{e}_k^X & \text{where } (\varepsilon^r, \varepsilon^e) \in \Phi_f^X \end{array} \right.$$

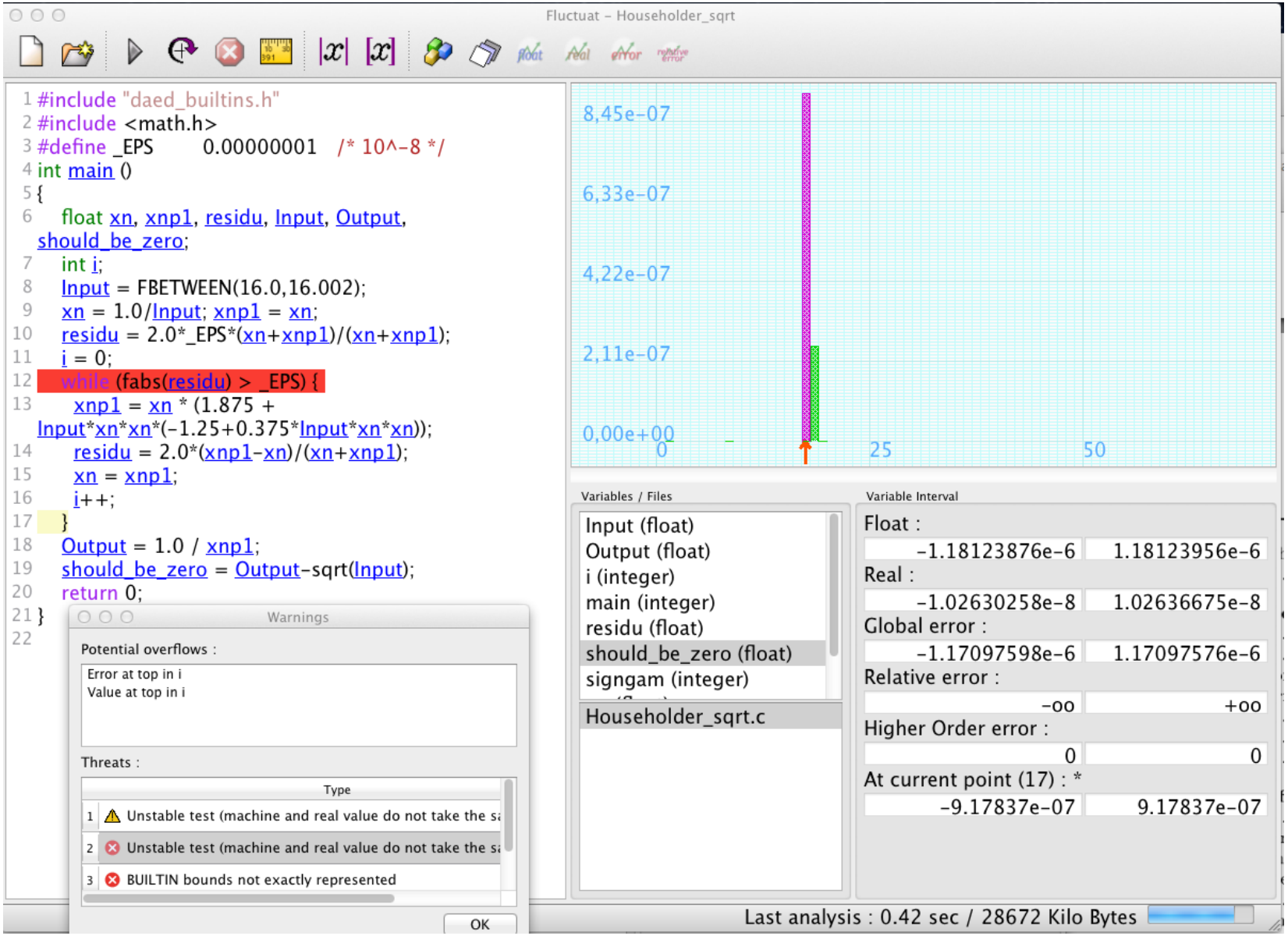
E^X is the propagated rounding error, D^X the propagated discontinuity error

New discontinuity errors computed when joining branches of a possibly unstable test

$Z = X \sqcup Y$ is $Z = (R^Z, E^Z, D^Z, \Phi_r^X \cup \Phi_r^Y, \Phi_f^X \cup \Phi_f^Y)$ such that

$$\left\{ \begin{array}{l} (R^Z, \Phi_r^Z \cup \Phi_f^Z) = (R^X, \Phi_r^X \cup \Phi_f^X) \sqcup (R^Y, \Phi_r^Y \cup \Phi_f^Y) \\ (E^Z, \Phi_f^Z) = (E^X, \Phi_f^X) \sqcup (E^Y, \Phi_f^Y) \\ D^Z = D^X \sqcup D^Y \sqcup (R^X - R^Y, \Phi_f^X \sqcap \Phi_r^Y) \sqcup (R^Y - R^X, \Phi_f^Y \sqcap \Phi_r^X) \end{array} \right.$$

Householder algorithm for square root



Conclusion

- Tractable discontinuity analysis implemented in FLUCTUAT (making it sound even when tests are unstable)
- More applications, in particular to robustness analysis in control
- Potential links with constraint-based approaches to the verification of finite-precision implementations such as, e. g.
  O. Ponsini, C. Michel, M. Rueher: Refining Abstract Interpretation Based Value Analysis with Constraint Programming Techniques. CP 2012

Come and see more about FLUCTUAT in the poster and demo session !