Robustness analysis of finite precision implementations

Eric Goubault and Sylvie Putot

Cosynus, LIX, Ecole Polytechnique

Journée MEA, calcul ensembliste et interprétation abstraite, 19/03/2015

Motivations (see Eric's talk)

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)
- Bound the implementation errors
- Implemented in the abstract interpreter FLUCTUAT

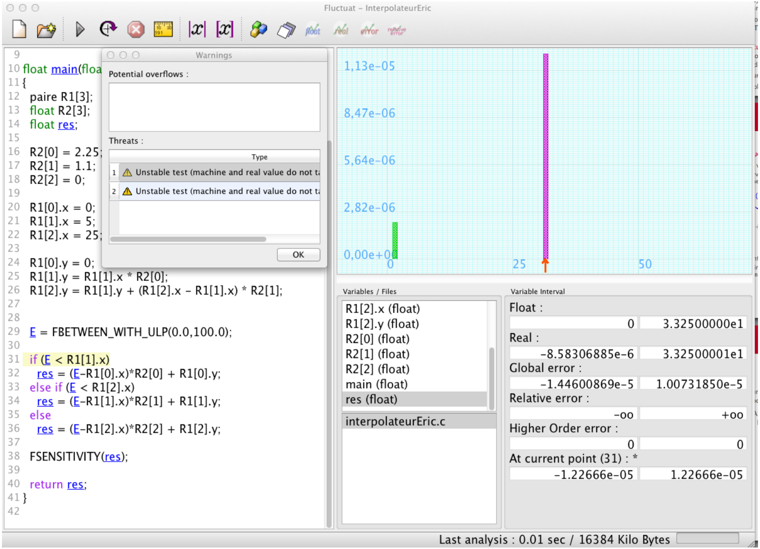
A difficulty in error analysis: unstable tests

- When finite precision and real control flows are potentially different
- If discontinuity of treatment between branches, joining error analyses from each branch is unsound
- When considering sets of executions, most tests are potentially unstable (just issuing a warning is not practical)

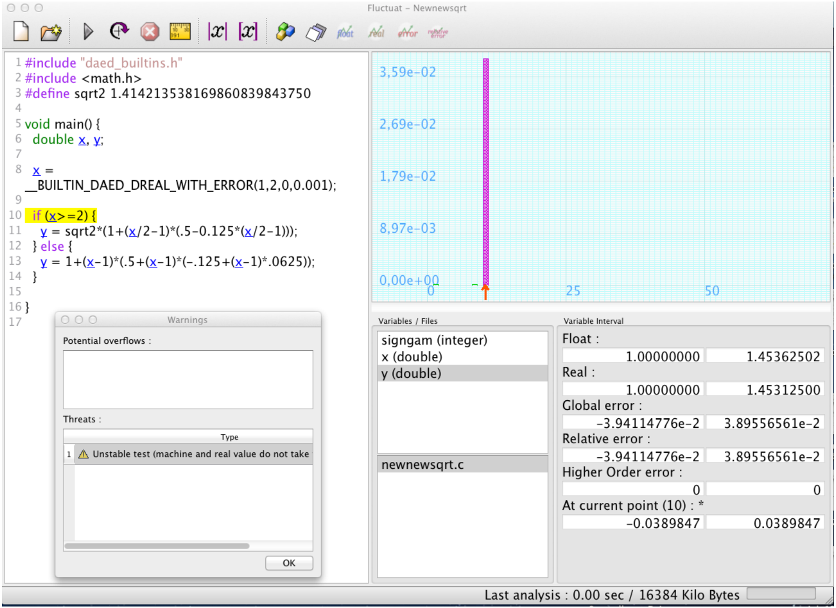
We propose here to compute discontinuity errors in unstable tests

- Bound the difference between the computation in two branches under conditions of unstable tests (mix constraint solving / affine arithmetic)
- Makes our error analysis sound in presence of unstable tests
- Provides a robustness analysis of the implementation

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



But discontinuities also actually occur (sqrt approx.)



Test interpretation

Fluctuat: computes rounding errors and propagates errors and uncertainties

- Relying on affine forms both for real value and error terms;
- With two sets of constraints on the noise symbols, resp. corresponding to real and finite control flows

Main idea to interpret test, informally

- Affine forms are unchanged, translate the test condition as a condition on the noise symbols ε_i
- The test condition is interpreted as a restriction of the set of inputs, that lead to an execution satisfying the condition: functional interpretation

Example

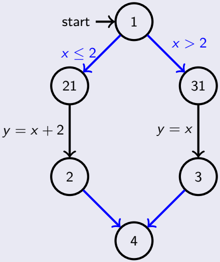
```
real x = [0,10];
real y = 2*x;
if (y >= 10)
  y = x;
```

- 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

Test interpretation and set operations - informally

Example

```
x=[1,3] + u;  //  [1]
if (x <= 2)   //  [2-1]
  y = x+2;    //  [2]
else          //  [3-1]
  y = x;      //  [3]
              //  [4]
}
```



$$
\begin{array}{rcll}
(x_1, y_1) & = & (2 + \varepsilon_1 + u, \top); & \varepsilon_1 \in \Phi_1^X = [-1, 1] \\
(x_{21}, y_{21}) & = & (x_1, y_1) \cap (x \leq 2); & \varepsilon_1 \in \Phi_{21}^X = [0, 1] \\
(x_2, y_2) & = & (x_{21}, x_{21} + 2); & \varepsilon_1 \in \Phi_{21}^X \\
(x_{31}, y_{31}) & = & (x_1, y_1) \cap (x > 2); & \varepsilon_1 \in \Phi_{31}^X = [-1, 0] \\
(x_3, y_3) & = & (x_{31}, x_{31}); & \varepsilon_1 \in \Phi_{31}^X \\
(x_4, y_4) & = & (x_2, y_2) \cup (x_3, y_3); & \varepsilon_1 \in \Phi_4^X = [-1, 1]
\end{array}
$$

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: <u>for a same execution (same values of the noise symbols ε_i)</u> 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 $k = 1, \ldots, p$:

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

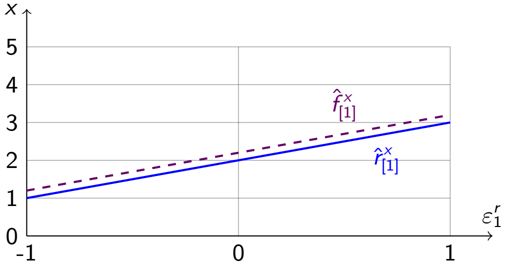
E^X is the propagated rounding error, D^X the propagated discontinuity error

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

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

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

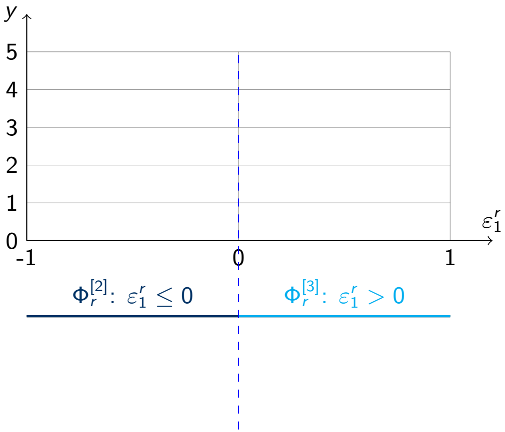


At cpt 1: $\hat{r}_{[1]}^x = 2 + \varepsilon_1^r$; $\hat{e}_{[1]}^x = u$

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

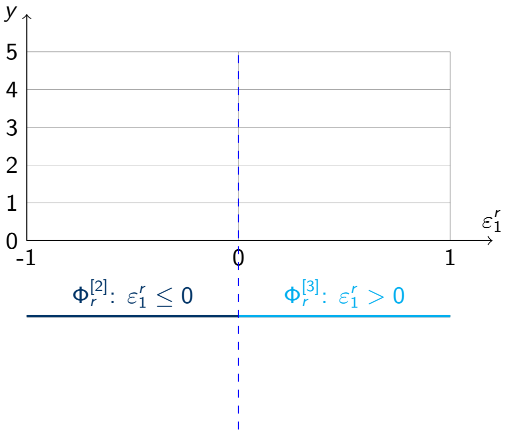


Example: sound unstable test analysis

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

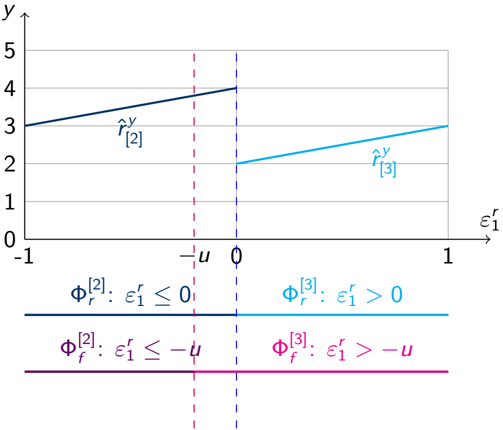
$\hat{r}^x_{[1]} = 2 + \varepsilon^r_1$; $\hat{e}^x_{[1]} = u$

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]
```



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

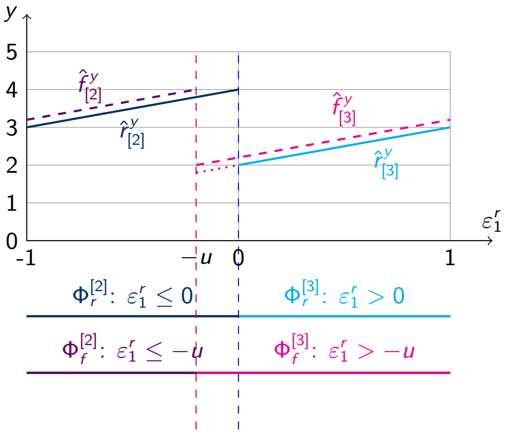
Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

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

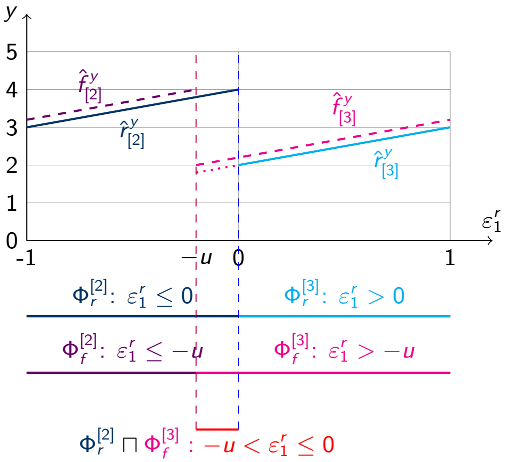
```
x := [1,3] + u; // [1]
if (x ≤ 2)
y = x+2; // [2]
else
y = x; // [3]
// [4]
```

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



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

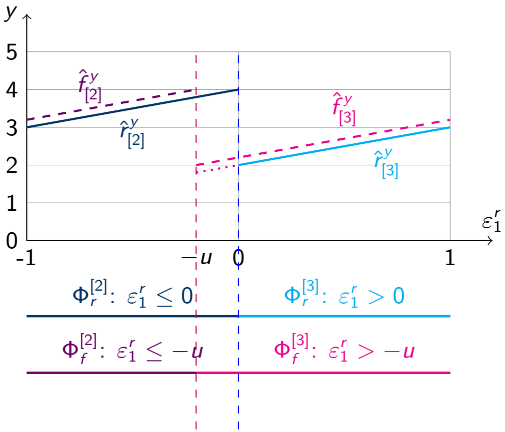


Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

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

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

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

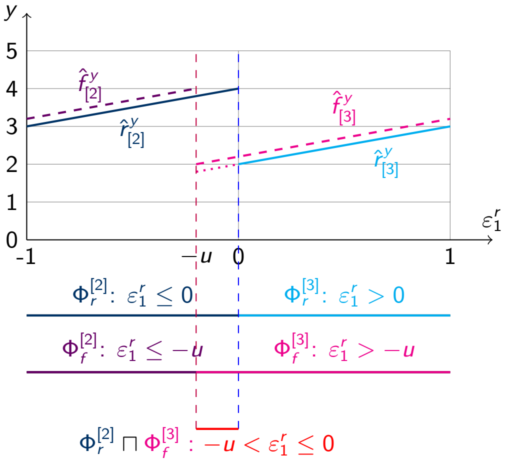


Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

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

Unstable test, second possibility:
$\Phi^{[3]}_r \sqcap \Phi^{[2]}_f = \emptyset$

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



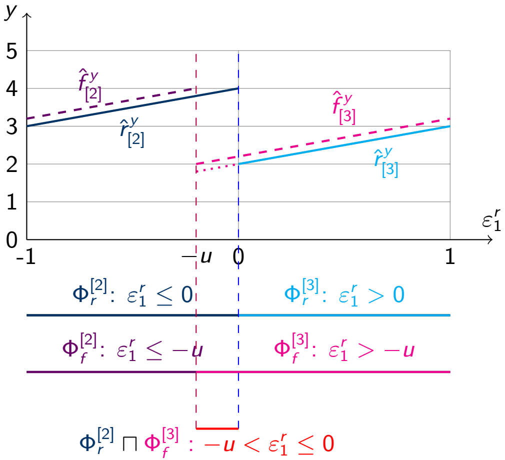
Real at [2]: $(\hat{r}^y_{[2]} = 4 + \varepsilon^r_1, \Phi^{[2]}_r)$
Real at [3]: $(\hat{r}^y_{[3]} = 2 + \varepsilon^r_1, \Phi^{[3]}_r)$

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

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

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

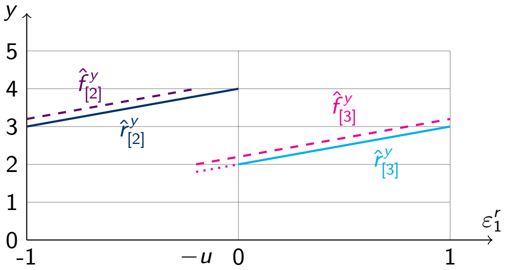
Error at [2]: $\hat{e}^y_{[2]} = u + \delta\varepsilon^e_2$

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

Error at [4] : $\underbrace{\hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]}}_{\hat{e}^y_{[4]}} + \underbrace{(\hat{r}^y_{[3]} - \hat{r}^y_{[2]}, \Phi^{[3]}_f \sqcap \Phi^{[2]}_r)}_{\hat{d}^y_{[4]}} = u + \delta\varepsilon^e_2 - 2\chi_{[-u,0]}(\varepsilon^r_1)$

Example: sound unstable test analysis

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

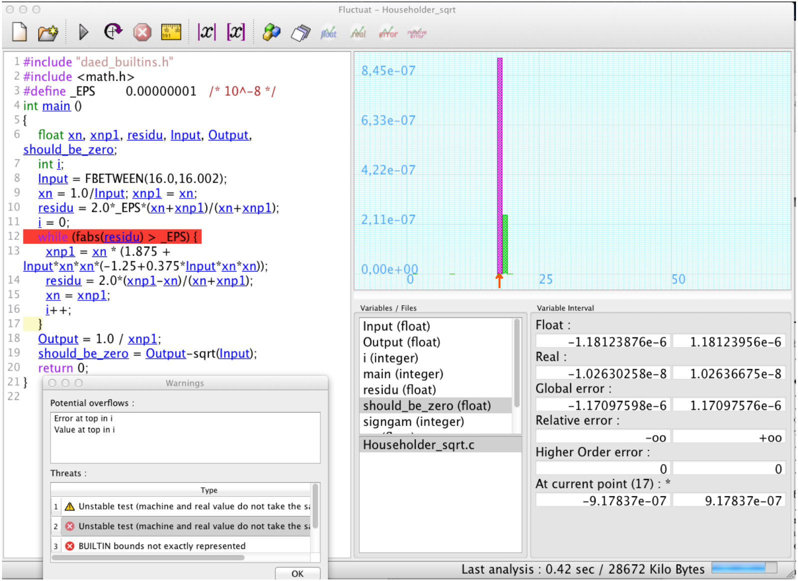


Error at [4] : $\underbrace{\hat{e}^y_{[2]} \sqcup \hat{e}^y_{[3]}}_{\hat{e}^y_{[4]}} + \underbrace{(\hat{r}^y_{[3]} - \hat{r}^y_{[2]}, \Phi_f^{[3]} \sqcap \Phi_r^{[2]})}_{\hat{d}^y_{[4]}} = \underbrace{u + \delta\varepsilon_2^e}_{\hat{e}^y_{[4]}} + \underbrace{-2\chi_{[-u,0]}(\varepsilon_1^r)}_{\hat{d}^y_{[4]}}$

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

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

Householder algorithm for square root



Conclusion

- Error bounds now sound even with unstable tests
- Candidate input values for unstable tests given by $\Phi_f^X \sqcap \Phi_r^Y$
- Robustness analysis and discontinuity error bounds applicable to general uncertainties (not only finite precision)
- A.I. techniques allow to extend error analyses compared to more classical interval-like techniques
- Natural potential on these robustness aspects for more interaction 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