Analyse en présence de tests instables dans Fluctuat
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

Abstract interpretation based static analysis
Relying on zonotopic abstract domains
Normalized floating-point numbers

\[ (-1)^s 1.x_1 x_2 \ldots x_n \times 2^e \] (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),

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

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 in Fluctuat

- 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)

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

\[
\begin{align*}
    f^x &= 0.1 + 1.49e^{-9} \quad [1] \\
    f^y &= 0.5 \\
    f^z &= 0.6 + 1.49e^{-9} \quad [1] + 2.23e^{-8} \quad [3] \\
    f^t &= 0.06 + 1.04e^{-9} \quad [1] + 2.23e^{-9} \quad [3] - 8.94e^{-10} \quad [4] - 3.55e^{-17} \quad [ho]
\end{align*}
\]
float x, y;
x = FREAL_WITH_ERROR(2,2,—1.0e−5,—1.0e−5);
if (x <= 2)
    y = x + 2;
else
    y = x;

Unstable test: \( r^x = 2 \) and \( f^x = 2 + 1.0e^{-5} \)

- execution in reals takes the then branch: \( r^y = r^x + 2 = 4 \),
- execution in floats takes the else branch: \( f^y = f^x = 2 + 1.0e^{-5} \).

The test introduces a discontinuity \( f^y - r^y = -2 + 1.0e^{-5} \) around the test condition \( x == 2 \)

- discontinuity in the sense of robustness/continuity analysis of Chaudhuri, Gulwani and al.
- want to consider this discontinuity as a new error term
- accurate abstraction?
Outline

- Introduction
- Affine arithmetic and zonotopic abstraction
  - of real values of program variables
  - extension to floating-point and error
- Abstraction in the case of unstable tests: discontinuity error analysis
Affine forms

- Affine form for variable $x$:

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

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

- Sharing $\varepsilon_i$ 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 $\mathbb{R}^p$
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)
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

```plaintext
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
For each variable:

- Interval $f^x = [\underline{f}^x, \overline{f}^x]$ bounds the finite precision value, $(\underline{f}^x, \overline{f}^x) \in F \times F$.
- Affine forms for real value and error:

$$
egin{align*}
  f^x &= (\alpha_0^x + \bigoplus_{i} \alpha_i^x \varepsilon_i^x) + \left( \begin{array}{c} e_0^x \\ \text{center of the error} \end{array} \right) + \bigoplus_{l} e_l^x \varepsilon_l^e \\
  &\quad + \bigoplus_{i} m_i^x \varepsilon_i^f \\
  &\quad \text{propag of uncertainty on value at pt } i
\end{align*}
$$
Test interpretation: stable test assumption (old version)

- Same control flow for finite precision and real executions
- If found not to be the case: unstable test warning
- When joining branches:
  - join values and errors coming from the two branches
  - in case of unstable test, possibly unsound error bounds

```plaintext
x = DREAL_WITH_ERROR(1, 3, -1.0 e - 5, -1.0 e - 5);
if (x <= 2)
  y = x + 2;
else
  y = x;
```

- Before the test: \( f^x = (2 + \varepsilon_1) - 10^{-5} \)
- Test \( x \leq 2 \):
  - then branch: \( \varepsilon_1 \leq 0 \) and \( f^y = \begin{cases} 4 + \varepsilon_1 - 10^{-5} + u\varepsilon_2^e \end{cases} \) (where \( u = \text{ulp}(1) \))
    \[ r^y \in [3, 4] \]
  - else branch: \( \varepsilon_1 > 0 \) and \( f^y = (2 + \varepsilon_1) - 10^{-5} \)
    \[ r^y \in [2, 3] \]
- Joining branches: \( f^y = \begin{cases} (3 + \eta_1) - 10^{-5} + u\eta_2^e \end{cases} \)
  \[ r^x \in [2, 4] \]
Tests interpreted independently over real and float values: two sets of constraints on (shared) noise symbols

Join on float and real values from two branches: same as previously

Join on error terms: join the error computed in the two branches with the difference between the real value in one branch and float value in the other branch when it is possible that for a same execution (same values of the noise symbols $\varepsilon_i$) the control flow is different

- this unstable test condition computed as an intersection of constraints on the $\varepsilon_i$: allows us to bound accurately the discontinuity error
- constraints on the $\varepsilon_i$ interpreted for the time being in intervals using additional slack variables (see example 2)
Abstract value

- For each variable:
  - Interval $f^x = [\underline{f^x}, \overline{f^x}]$ 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 $\eta$ symbols

$$f^x = \left( \alpha_0^x + \bigoplus_i \alpha_i^x \varepsilon_i^r \right) + \left( \varepsilon_0^x \right) + \left( \bigoplus_l e_l^x \varepsilon_l^e \right)$$

- real value
- center of the error
- uncertainty on error due to point $l$

$$+ \left( \bigoplus_i m_i^x \varepsilon_i^r \right)$$

- propag of uncertainty on value at pt $i$

- A couple of constraints on noise symbols (interval + equality constraints) coming from test interpretation:
  - on affine forms $f^x = r^x + e^x$ for finite precision control flow
  - on affine forms $r^x$ for real control flow
Back to the example: error with sound unstable test analysis
is now in \([-2, 10^{-5}]\)
Example: sound unstable test analysis

```c
int main(void) {
    double x, y;
    x = DREAL_WITH_ERROR(1, 3, -1.0e-5, -1.0e-5);
    if (x <= 2)
        y = x + 2;  [1]
    else
        y = x;  [2]
}
```

- Before the test: \( f^x = (2 + \varepsilon_1) - 10^{-5} \)
- Test \( x \leq 2 \):
  - in reals: \( \varepsilon_1 \leq 0 \)
  - in floats: \( \varepsilon_1 + 1.0e^{-5} \leq 0 \), ie \( \varepsilon_1 \leq -1.0e^{-5} \).
- First unstable test possibility:
  - real takes then branch: \( \varepsilon_1 \leq 0 \)
  - float takes else branch: \( \varepsilon_1 > -1.0e^{-5} \)
  - unstable test = intersection of constraints: \( -1.0e^{-5} < \varepsilon_1 \leq 0 \)
    \[
    f^y_{[2]} - r^y_{[1]} = (2 + \varepsilon_1 + 1.0e^{-5}) - (4 + \varepsilon_1) = -2 + 1.0e^{-5}.
    \]
- Second unstable test possibility: conditions \( \varepsilon_1 \leq -1.0e^{-5} \) and \( \varepsilon_1 > 0 \) are non compatible (no unstable test)
Second example

```c
#include "daed_buildins.h"
int main(void) {
  double x,y,t;
  x = DBETWEEN(-0.5,0.5)*2;
  y = DBETWEEN(-0.5,0.5)*2;
  if (x < y)
    t = y - x;
  else
    t = x - y;
}
```

The code snippet above is intended to test for discontinuity, but it appears to work correctly as there is no actual discontinuity. The variable `t` should be a difference between `x` and `y`, and the test seems to pass as there is no actual discontinuity in the data points presented.
First discontinuity error:

- real takes then branch: $\varepsilon_1^r < \varepsilon_2^r$
- float takes if branch: $\varepsilon_1^r + u\varepsilon_1^e \geq \varepsilon_2^r + u\varepsilon_2^e$ (where $u = ulp(1)$)
- intersection (unstable test), computed in intervals
  - using additional slack variable $\eta_1^t = \varepsilon_1^r - \varepsilon_2^r$, which will appear both in the constraints on real and floats (more generally, if we are interested in the test $exp1 \ op \ exp2$, we will associate a slack variable to $exp1-exp2$).
  - we then get $-2u < \varepsilon_1^r - \varepsilon_2^r < 0$ and $\varepsilon_2^e \leq \varepsilon_1^e$

$$f_{[5]}^t - r_{[4]}^t = (\varepsilon_3^r + \varepsilon_1^r - \varepsilon_2^r + u\varepsilon_1^e - u\varepsilon_2^e + 5u\varepsilon_5^e) - (\varepsilon_3^r + \varepsilon_2^r - \varepsilon_1^r)$$

and thus

$$f_{[5]}^t - r_{[4]}^t = 2(\varepsilon_1^r - \varepsilon_2^r) + u(\varepsilon_1^e - \varepsilon_2^e) + 5u\varepsilon_5^e \in [-7u, 7u]$$
Conclusion

- A sound and accurate abstraction of errors in presence of unstable tests
- Still on-going work
  - a first implementation in Fluctuat
  - using additional slack variables to improve (in intervals) the abstraction of intersections of constraints on noise symbols
  - towards integrated use of constraint solving?