FLUCTUAT: VERIFICATION OF ACCURACY PROPERTIES OF NUMERICAL COMPONENTS AND SYNCHRONOUS EMBEDDED SOFTWARE

Journée précision numérique | Védrine Franck
Joint work with Eric Goubault & Sylvie Putot & Karim Tekkal + Maxime Jacquemin
VERIFICATION OF POLYNOMIAL APPROXIMATIONS
VERIFICATION OF INTERPOLATION TABLE

double y[10] = { 0.0, 2.0, 3.0, 2.0, 0.0, -2.0, -3.0, -1.9, 0.1, 2.1 };
double in = interval_with_error(0, 10, -1e-6, 1e-6), out;
SPLIT
int index = (int) in;
MERGE
if (index < 0) out = y[0];
else if (index >= 9) out = y[9];
else out = y[index] + (in - index) * (y[index+1] - y[index]);
MERGE

• 11 regular paths
• 110 possible unstable branches
• 20 effective unstable branches \Rightarrow 40 unstable paths
• 51 abstract interpretation of the code between SPLIT/MERGE
• out: float \in [-3.0, 3.0], ideal \in [-3.0 - 1.11\times10^{-16}, 3.0],
  \quad error \in [-4.0\times10^{-6} - 4.85\times10^{-15}, 4.0\times10^{-6} + 4.44\times10^{-15}]
VERIFICATION OF CONVERGENT LINEAR FILTERS

- **Why it works**
  - all computations are linear
  - almost no approximation in Fluctuat domain and error = same type of transfer function
  - \( 1 + x + x^2 + \ldots + x^n \) bounded by \( 1/(1-x) \)

- **Transfer function**
  \[
  (M) = \begin{pmatrix} 0 & 1 \\ 1.4 & -0.7 \end{pmatrix}
  \]

\[
S_n = \sum_{k=0}^{n} (M)^k \cdot (V) + (V_0) - (V)
\]
CONVERGENCE PROOF FOR CONVERGENT LINEAR FILTERS

```c
int main(int N) {
    double S, SO, E, EO, EI, S1;
    int Ej;
    S = 0.0;
    SO = 0.0;
    S1 = 0.0;
    E = DBETWEEN(-1.0,1.0);
    EO = DBETWEEN(-1.0,1.0);
    N = TOP(int);
    for (i=1;i<N;++i) {
        EI = EO;
        EO = E;
        E = DBETWEEN(-1.0,1.0);
        S1 = SO;
        SO = S;
        S = 0.7 * E - 1.3 * SO + 1.1 * EI + 1.4 * SO - 0.7 * S1;
        DPRINT(S);
        DPRINT(SO);
        DPRINT(S1);
    }
    DSENSITIVITY(S);
}
```
**EMBEDDED SYNCHRONOUS SYSTEM**

- **Infinite loop**
  - Take inputs from sensors
  - Do some computation
    - in constrained time
    - without any allocation
    - read/write internal state variables (ex filters)
  - Produce outputs for actuators

- **Questions**
  - Is the accuracy of outputs bounded over time? within acceptable bounds?
  - Is the system robust?
    - small perturbation of inputs produces a small perturbation of outputs

- **Objective**
  - Ensure that the implementation is conform to the model
  - Avoid to take irreversible decisions on wrong computations
FLUCTUAT CAN BOUND THE ACCURACY

- if state variables only concern
  - convergent linear filters of the inputs
  - and/or convergent linear filters of the outputs

- if no discontinuous unstable branches can occur

- if all atomic operations respect the operation hypotheses
  - no possible division by zero,
  - no negative square root

- Then Fluctuat can bound the accuracy of the output
  - for input values in full range
  - in theory for every simulation cycle,
    - unroll the loops on first cycles and see the evolution of error on the filters’ state variables
    - depends on time, find adequate parameters (cyclic unfolding, widening and narrowing threshold) to achieve the proof for all cycles
- Fluctuat has many reasons to fail at bounding domains and accuracy for a full range of inputs
  - acceptable bounds may not exist
  - accumulation of over-approximations

- Apply the methodology from some test-cases
  - compare with observed accuracy with MPFR
  - compare with stochastic intended accuracy with Verrou, Verificarlo, Cadna
  - guaranteed intended accuracy around the test-case
    - investigation and annotations for the unstable branches
    - verification of the results on the first iteration cycles

- More and more complex system – ex geolocalization with solvers inside
• To find bugs when different tools return unexpectedly different results
  • ex: if you forget to std::: when calling the abs function, a discontinuous unstable branch will reveal an implicit int conversion

• In a certification context (DO178C – avionics, FDA – medical, …)

• To progressively add annotations and go to proofs
CONCLUSION

- Fluctuat has shown that Abstract Interpretation can effectively deal with accuracy properties to infer and prove accuracy bounds

- Fluctuat has features that needs to be developed to target HPC codes
  - ability to analyze parts of code on large input ranges
    - create a summary that computes an approximation of the precision loss of a function
    - challenge: more symbolic computations for the error
  - ability to keep a trace of the origins of the numerical errors (data flow)
    - to couple with debugging capabilities to explain abnormal behavior
    - challenge: avoid explosion of the number of noise symbols
  - ability to generate and simulate a worst-case
    - to couple with debugging capabilities
    - challenge: define a semantics different but close to IEEE-754 to run this worst-case
  - ability to generates dependencies from a specific parameter/input
    - challenge: adequate user-interface to quickly investigate these dependencies
Thanks for your attention!