Formally Verified Roundoff Errors using SMT-based Certificates and Subdivisions

Joachim Bard, Heiko Becker, and Eva Darulova

MPI-SWS, Saarland Informatics Campus {jbard,hbecker,eva}@mpi-sws.org

Abstract. When compared to idealized, real-valued arithmetic, finite precision arithmetic introduces unavoidable errors, for which numerous tools compute sound upper bounds. To ensure soundness, providing formal guarantees on these complex tools is highly valuable.
In this paper we extend one such formally verified tool, FloVer. First, we extend FloVer with an SMT-based domain using results from an external SMT solver as an oracle. Second, we implement interval subdivision on top of the existing analyses. Our evaluation shows that these extensions allow FloVer to efficiently certify more precise bounds for nonlinear expressions.

Keywords: Coq · roundoff error · finite-precision · SMT · subdivision

1 Introduction

Floating-point or fixed-point arithmetic are commonly used representations of the reals in today's computers. They necessarily only provide a discrete approximation of infinite-precision reals, resulting in roundoff errors. These errors are introduced by arithmetic operations and are individually small, but can accumulate during the course of a computation. For safety-critical systems, it is thus imperative to soundly bound the overall roundoff error of a program.

A number of automated static analysis tools have been developed in the past for computing roundoff error bounds [14,13,10,9,4,6]. However, their analyses and implementations are complex, raising questions of correctness. Most of these tools thus generate certificates which can be independently and formally verified by a theorem prover such as Coq [1], PVS [12] or HOL4 [2].

One tool to check certificates is FloVer [3], an open source certificate checker for roundoff errors computed using a dataflow static analysis. FloVer's *checker functions* are formally verified in Coq and HOL4 and check roundoff error bounds computed by external tools for floating-point as well as fixed-point arithmetic. The current version of FloVer uses the interval [11] and affine arithmetic (AA) [8] abstract domains, which are efficient and accurate for linear expressions, but which suffer from over-approximations for nonlinear arithmetic programs.

In this paper, we describe two new approaches to certify tighter error bounds and implement them in FloVer's Coq formalization. First, we implement an SMT-based range estimation [5] (Section 2) which computes tighter enclosures

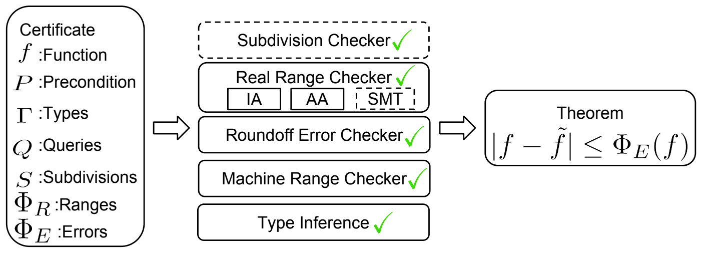

Fig. 1. Overview of FloVer's infrastructure

for expressions using a trusted SMT solver as an oracle. Second, interval subdivision [9,4] (Section 2) further increases analysis precision by splitting input ranges into disjoint subintervals and by analyzing them separately. These techniques are employed by unverified state-of-the-art tools [9,4] but were beyond the scope of formally verified checkers. Our extensions presented in this paper thus close the gap between the errors computed from state-of-the-art tools and what can be certified. Our experimental evaluation (Section 3) shows that our extensions increase FloVer's accuracy on a standard benchmark set of nonlinear expressions, while maintaining a reasonable certificate checking time. Our implementation is available online at `https://gitlab.mpi-sws.org/AVA/FloVer/tree/SMT_Subdiv`.

Related Work PRECiSA [14], FPTaylor [13], and real2float [10] provide certificate checkers like FloVer, verifying roundoff error bounds encoded by an untrusted static analysis. Certificates of PRECiSA are written in PVS, FPTaylor's in HOL-Light, and real2float's and FloVer's in Coq. Unlike FloVer, their roundoff error verification is based on global optimization. This approach can often verify tighter error bounds than a dataflow analysis, but is currently only applicable to floating-point arithmetic computations and not fixed-point arithmetic. PRECiSA in addition handles loops by widening, and conditional branches by path-by-path error analysis, which are orthogonal to the error estimation of straight-line code which we focus on in this paper.

Gappa [6] is a general purpose finite-precision analysis tool inferring roundoff error bounds, but is not limited to only those. It bounds roundoff errors with a dataflow analysis like FloVer's using intervals as the abstract domain. A certificate in Gappa is encoded as a chain of lemmas proven at checking time, whereas FloVer encodes certificates as a call to a function proven sound once and forall. Gappa already supports subdivisions and, as it emits Coq proofs, we believe that the SMT extension in this paper can also improve its computed error bounds.

2 Extensions to FloVer

Figure 1 illustrates FloVer's *modular* checker structure. Each checker function is first proven correct individually and the separate proofs are then combined into an overall soundness theorem which states that if all checker functions are successful, the roundoff error bound encoded in the certificate is sound. This design facilitates relatively easy extensions, and allows for efficient certificate checking; verifying a certificate does not require any formal proofs at certificate checking time, or formal proof expertise by the user.

FloVer supports arithmetic expressions $(+, -, *, /)$, a fused-multiply-add operation and let-bindings. As other dataflow analysis based tools, FloVer splits checking of roundoff error bounds into checking of real-valued range bounds (`Real Range Checker` in Figure 1) and checking of error bounds (`Roundoff Error Checker`). Roundoff error bounds are checked for mixed-precision programs with 16, 32 and 64 bit floating-points, or arbitrary fixed-point precisions. The type checker (`Type Inference`) verifies that all mixed-precision type assignments are valid. Component `Machine Range Checker` checks that evaluation results can be represented in their inferred type, i.e. no overflow occurs.

A certificate checked by FloVer encodes only the minimum necessary information: the analyzed expression f, range (Φ_R) and roundoff error bounds (Φ_E) inferred by a static analysis tool, the precondition constraining input variables (P), a type assignment Γ, the queries to the SMT solver (Q) and the interval subdivisions (S). Our extensions are marked in Figure 1 by dashed lines. We implement SMT-based range estimation as a real-valued range analysis (`Real Range Checker`). Interval subdivision is implemented on top of the existing components (`Subdivision Checker`) and reuses FloVer's existing checker functions internally.

Extension 1: Tighter Ranges using SMT Oracles. Our first extension to FloVer introduces an abstract domain for computing tighter range bounds based on the existing analysis implemented in the static analyzer Daisy [5]. This analysis tracks ranges as plain intervals and achieves better accuracy by using a nonlinear decision procedure provided by an SMT solver to track nonlinear correlations, which cannot be captured by the existing interval and AA-based domains.

Given an expression e, a range bound $[e_{lo}, e_{hi}]$ is first computed using interval arithmetic. Next, the analysis attempts to tighten e_{lo} and e_{hi} separately. For the lower bound, it queries an SMT solver whether e, constrained by the precondition, can take a value which is smaller than some e'_{lo} with $e_{lo} < e'_{lo}$ If the query is unsatisfiable, the tighter bound $[e'_{lo}, e_{hi}]$ is sound, and tightenting repeats a predetermined number of times using a binary search. Tightening of the upper bound is analogous. If the solver times out, the bound is not tightened.

The SMT-based analysis in Daisy makes multiple queries to the SMT solver for tightening a single range. Of these queries, only the last unsatisfiable one for each lower and upper bound is relevant for correctness. We thus instrument Daisy such that these last queries are saved and encoded in a certificate. We do not otherwise modify Daisy.

During certificate checking, we treat the results of SMT queries as oracles. Verifying the query results themselves would require proof reconstruction which current SMT solvers do not support due to the complexity of nonlinear arithmetic. Instead, we trust the SMT solver, but keep the amount of queries that must be trusted to a minimum by storing only the last queries.

We implemented the `SMT` component of the `Real Range Checker` from Figure 1 in the checker function `validSMTBounds` (`f`, `P`, Φ_R, `Q`) by structural recursion on the AST of the analyzed expression `f`. For each subexpression of `f`, a sound interval enclosure is computed first using existing FloVer infrastructure. If `Q` contains SMT queries which were used to improve the lower or upper bound, `validSMTBounds` checks first that the queries were correctly encoded by Daisy (we check that the expression and precondition encoded in the query match the currently analyzed expression and the precondition given in the certificate). If this check succeeds, the function checks that the range bound can be tightened to the new bound encoded in the query. Finally, FloVer checks that the inferred range bound is contained in the interval enclosure encoded in the analysis result Φ_R. The soundness proof of `validSMTBounds` shows that if the checker succeeds, the range bound encoded in Φ_R is valid.

Extension 2: Interval Subdivision. The second analysis we implement in FloVer is interval subdivision, which splits the input domain into equally-sized subdomains. The range and roundoff error analyses are then run on each subdomain separately and joined together into a global analysis result. The over-approximations on each subdomain tend to be smaller, which increases the overall tightness of range and error bounds.

Checker function `validSubdivs` (`f`, `P`, Φ_R, Φ_E, `S`) implements checking of interval subdivisions, where `S` is a list of subdomains, represented as quadruples $(P^S, \Phi_R^S, \Phi_E^S, Q^S)$. The checker function checks correctness for each subdomain in `S` by calling the existing certificate checker on `f`, P^S, Φ_R^S, Φ_E^S, and Q^S. `validSubdivs` checks that the global analysis results Φ_R and Φ_E are upper bounds for the current subdivision results Φ_R^S and Φ_E^S for each subexpression of f. Performing this check on every element of `S` proves correctness of the global analysis results Φ_R and Φ_E.

Finally, `validSubdivs` checks that the subdomains (P^S) cover the overall input domain (encoded in `P`), to ensure that Daisy did not forget a subdomain in the roundoff error computation. The check iterates over the free variables of `f`. For each free variable x and subinterval $[x_{lo}, x_{hi}]$ we check that there exist subdomains where P^S maps x to $[x_{lo}, x_{hi}]$ and the union of these subdomains covers the full global range constraint for all other free variables. This essentially checks for each free variable that Daisy computed the correct cartesian product.

The soundness theorem for both our extensions is:

Theorem 1. *Let* `f`, `P`, Φ_R, Φ_E, *and* `S` *be as before. If for all* $(P^S, \Phi_R^S, \Phi_E^S, Q^S)$ *in* `S` *the queries encoded in* Q^S *are unsatisfiable, and* `validSubdivs`$(f, P, \Phi_R, \Phi_E, S)$ *succeeds, there exists an idealized real-value* $v_{\mathbb{R}}$*, a finite-precision value* $v_{\mathbb{F}}$ *and a precision* m*, such that* f *evaluates to* $v_{\mathbb{R}}$ *under an idealized real-valued semantics,* $v_{\mathbb{F}}$ *has precision* m*, and* f *evaluates to* $v_{\mathbb{F}}$ *under finite-precision semantics. Furthermore,* $\Phi_E(f)$ *is an upper bound to the roundoff error* $|v_{\mathbb{R}} - v_{\mathbb{F}}|$.

3 Experiments

We have evaluated our extension of FloVer to check whether it can verify more precise error bounds with reasonable certificate checking times. As neither SMT-

Benchmark	Interval	Affine	SMT	Subdiv	SMT & Subdiv	Cmp.	FPTaylor
Bspline0	2.41e-16	2.41e-16	2.41e-16	2.41e-16	2.41e-16	1.00	**1.39e-16**
Bspline1	1.52e-15	1.60e-15	1.35e-15	1.28e-15	1.19e-15	0.79	**5.15e-16**
Bspline2	1.41e-15	1.45e-15	1.19e-15	1.26e-15	1.16e-15	0.83	**5.43e-16**
Bspline3	1.30e-16	1.30e-16	1.30e-16	1.30e-16	1.30e-16	1.00	**8.33e-17**
Doppler	6.53e-13	5.61e-12	6.12e-13	3.03e-13	3.03e-13	0.46	**1.22e-13**
DopplerFMA	6.41e-13	5.51e-12	6.00e-13	2.99e-13	2.99e-13	0.47	**1.21e-13**
Floudas26	1.05e-12	1.07e-12	8.13e-13	1.04e-12	$\perp^{\%}$	0.77	**7.74e-13**
Floudas33	7.29e-13	7.29e-13	**4.93e-13**	7.29e-13	$\perp^{\%}$	0.68	6.20e-13
Floudas34	3.11e-15	3.11e-15	3.11e-15	3.11e-15	$\perp^{\%}$	1.00	**2.22e-15**
Floudas46	1.55e-15	1.55e-15	1.55e-15	1.55e-15	$\perp^{\%}$	1.00	1.55e-15
Floudas47	2.80e-14	2.85e-14	2.30e-14	2.73e-14	$\perp^{\%}$	0.82	**1.67e-14**
Floudas1	7.29e-13	7.29e-13	**4.93e-13**	7.29e-13	$\perp^{\%}$	0.68	5.76e-13
Himmilbeau	3.42e-12	3.42e-12	1.50e-12	1.50e-12	1.50e-12	0.44	**1.00e-12**
InvPendulum	5.37e-14	5.37e-14	5.37e-14	5.37e-14	5.37e-14	1.00	**3.84e-14**
JetEngine	$\perp^{0}$	$\perp^{0}$	1.67e-08	$\perp^{0}$	1.87e-10	—	**1.72e-11**
Kepler0	1.85e-13	1.77e-13	1.77e-13	1.70e-13	1.65e-13	0.93	**7.71e-14**
Kepler1	8.97e-13	8.21e-13	8.47e-13	7.07e-13	6.63e-13	0.81	**3.04e-13**
Kepler2	4.13e-12	3.81e-12	3.77e-12	3.75e-12	3.52e-12	0.93	**1.60e-12**
RigidBody1	5.58e-13	5.58e-13	5.58e-13	5.58e-13	5.58e-13	1.00	**2.95e-13**
RigidBody2	6.57e-11	6.57e-11	6.57e-11	6.57e-11	6.57e-11	1.00	**3.61e-11**
Verhulst	8.34e-16	8.34e-16	8.34e-16	7.01e-16	7.01e-16	0.84	**3.24e-16**
PredatorPrey	3.40e-16	3.47e-16	3.40e-16	3.20e-16	3.20e-16	0.94	**1.84e-16**
CarbonGas	5.69e-08	5.67e-08	5.49e-08	2.07e-08	2.03e-08	0.36	**9.13e-09**
Turbine1	1.59e-13	1.59e-13	1.50e-13	6.49e-14	6.32e-14	0.40	**1.67e-14**
Turbine2	2.21e-13	2.23e-13	2.09e-13	5.89e-14	5.64e-14	0.26	**2.00e-14**
Turbine3	1.11e-13	1.11e-13	1.04e-13	2.47e-14	2.43e-14	0.22	**8.69e-15**

Table 1. Roundoff errors verified by FloVer and FPTaylor

based techniques nor interval subdivisions improve precision for linear benchmarks our evaluation focuses on nonlinear ones. For our experiments we used a Debian 9 machine with a 3.3GHz four-core Intel i5-6600 processor and 16 GB of main memory. Daisy uses Z3 [7] for the SMT-based analysis. When using interval subdivision we split at most 3 input ranges into 5 subintervals each, resulting in at most 125 subdomains.

Precision Improvements. Table 1 compares the roundoff errors verified by the existing version of FloVer [3] (columns 'Interval' and 'Affine') with those verified by our extensions (columns 'SMT' and 'Subdiv') and those computed by FPTaylor, a state-of-the-art optimization-based analyzer. Column 'SMT & Subdiv' shows roundoff errors computed using both interval subdivision and SMT-based range estimation. All errors are computed for uniform 64-bit floating-point precision.

Column 'Cmp.' shows the ratio by which our new analyses improve over the roundoff error that could be verified by FloVer before (best new analysis / best

Benchmark	Interval		Affine		SMT		Subdiv		SMT & Subdiv	
	Daisy	Coq	Daisy	Coq	Daisy	Coq	Daisy	Coq	Daisy	Coq
Bsplines	3.00	3.51	2.95	3.55	7.77	3.83	3.71	12.57	10.63	12.82
Doppler	6.04	6.70	2.71	7.36	5.22	4.95	6.1	237.25	86.54	237.86
DopplerFMA	2.57	4.20	2.72	6.99	5.46	4.05	5.29	167.01	86.08	172.64
Floudas	3.93	5.58	4.24	5.68	58.1	11.26	13.04	672.86	$\perp^{\%}$	$\perp^{\%}$
Himmilbeau	2.83	3.21	2.91	3.63	5.96	3.46	3.85	23.77	31.3	25.33
InvPendulum	2.68	3.14	2.71	3.48	4.89	3.46	4.28	50.82	67.65	50.86
JetEngine	$\perp^0$	$\perp^0$	$\perp^0$	$\perp^0$	34.28	47.07	$\perp^0$	$\perp^0$	120.99	1158.97
Kepler	3.21	12.53	3.2	13.44	55.99	13.15	12.38	1326.32	1840.17	1427.25
RigidBody	2.68	3.92	2.74	3.57	11.25	4.07	5.79	138.86	275.19	155.98
Science	2.89	6.87	2.89	420.50	7.73	6.88	3.8	25.99	12.86	26.69
Turbine	3.56	12.98	3.79	19.77	12.89	13.25	14.04	1476.35	331.98	1507.11

Table 2. Running times for Daisy and FloVer in seconds

previous analysis), values < 1.0 mean that a tighter roundoff error bound can be proven. We highlight the smallest roundoff error among all verifiers in bold.

While FPTaylor usually computes the best roundoff error, the errors verified by our extension bring FloVer closer to the state-of-the-art. FloVer further supports fixed-point arithmetic which FPTaylor does not (which is why we perform the comparison in floating-points). For none of our benchmarks the roundoff error has become worse and we further achieve significant improvements where the new roundoff error is up to 4.5 times smaller than the old roundoff error (Turbine3). Verifying SMT-based results also allowed us to compute and verify a roundoff error for the JetEngine benchmark, for which interval and affine arithmetic report a spurious division by zero error (denoted by $\perp^0$). For the Floudas benchmarks, Daisy does not compute any roundoff error when using both SMT and subdivision due to a missing check for empty subdomains (denoted by $\perp^{\%}$).

Running Times. We give the overall certificate checking times of FloVer for each benchmark in Table 2. For each of the analyses supported by FloVer, we give the end-to-end running times for both Daisy and FloVer's Coq implementation on the full benchmark file (one file may include multiple functions and thus multiple calls to the certificate checker). The certificate checking times for our extension are higher than those of the baseline as expected, but remain reasonable (below 2 hours for the most complex benchmark). FPTaylor's checking times are in the same order of magnitude as those for SMT with subdivisions.

Summary. Our evaluation has shown that checking certificates with our extension of FloVer is feasible and improves its accuracy. Given the implemented analyses, FloVer now supports the same analyses as the state-of-the-art dataflow-analysis based tool Daisy. FloVers Coq formalization makes it reusable for other tools like Gappa to increase their precision using SMT-based range estimation and interval subdivision.

References

1. The Coq Proof Assistant, `https://coq.inria.fr`
2. The HOL4 Theorem Prover, `https://hol-theorem-prover.org/`
3. Becker, H., Zyuzin, N., Monat, R., Darulova, E., Myreen, M.O., Fox, A.: A Verified Certificate Checker for Finite-Precision Error Bounds in Coq and HOL4. In: 2018 Formal Methods in Computer Aided Design (FMCAD). pp. 1–10. IEEE (2018)
4. Darulova, E., Izycheva, A., Nasir, F., Ritter, F., Becker, H., Bastian, R.: Daisy-Framework for Analysis and Optimization of Numerical Programs (Tool Paper). In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 270–287. Springer (2018)
5. Darulova, E., Kuncak, V.: Towards a compiler for reals. ACM Transactions on Programming Languages and Systems (TOPLAS) **39**(2), 8 (2017)
6. De Dinechin, F., Lauter, C.Q., Melquiond, G.: Assisted Verification of Elementary Functions using Gappa. In: ACM Symposium on Applied Computing (SAC) (2006)
7. De Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
8. de Figueiredo, L.H., Stolfi, J.: Affine Arithmetic: Concepts and Applications. Numerical Algorithms **37**(1-4) (2004)
9. Goubault, E., Putot, S.: Static Analysis of Finite Precision Computations. In: Verification, Model Checking, and Abstract Interpretation (VMCAI) (2011)
10. Magron, V., Constantinides, G., Donaldson, A.: Certified Roundoff Error Bounds Using Semidefinite Programming. ACM Transactions on Mathematical Software (TOMS) **43**(4), 34 (2017)
11. Moore, R.: Interval Analysis. Prentice-Hall (1966)
12. Owre, S., Rushby, J.M., Shankar, N.: Pvs: A prototype verification system. In: International Conference on Automated Deduction. pp. 748–752. Springer (1992)
13. Solovyev, A., Jacobsen, C., Rakamaric, Z., Gopalakrishnan, G.: Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions. In: International Symposium on Formal Methods (FM) (2015)
14. Titolo, L., Feliú, M.A., Moscato, M., Munoz, C.A.: An Abstract Interpretation Framework for the Round-off Error Analysis of Floating-Point Programs. In: International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). pp. 516–537. Springer (2018)