WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Correct Approximation of IEEE 754 Floating-Point Arithmetic for Program Verification

Rajesh Sharma[1], Saurav[2], Shipra Singh[3], Priyanshu Kumar[4]

[1-4] Department of Computer Science & Engineering IIMT College of Engineering ,Greater Noida

ABSTRACT :

Verification of programs using floating-point arithmetic is challenging on several accounts. One of the difficulties of reasoning about such programs is due to the peculiarities of floating-point arithmetic: rounding errors, infinities, non-numeric objects (NaNs), signed zeroes, denormal numbers, different rounding modes, etc. One possibility to reason about floating-point arithmetic is to model a program computation path by means of a set of ternary constraints of the form $z = x \; op \; y$ and use constraint propagation techniques to infer new information on the variables' possible values. In this setting, the authors define and prove the correctness of algorithms to precisely bound the value of one of the variables (x, y or z), starting from the bounds known for the other two. They do this for each of the operations (addition, subtraction, multiplication, division) and for each rounding mode defined by the IEEE 754 binary floating-point standard — even in cases where the rounding mode in effect is only partially known. This is the first time such so-called "filtering algorithms" are defined and their correctness formally proved. The contribution provides a solid foundation for formal verification of programs that use floating-point arithmetic. By handling all IEEE 754 corner-cases — including infinities, NaNs, denormals, signed zeros — and all rounding modes, the approach ensures that no valid solution is overlooked (i.e. no false negatives), and no spurious solutions are introduced (i.e. no false positives). The work thus paves the way toward reliable static analysis, exception detection, or test-case generation for floating-point programs under real-world conditions.

Keyword: IEEE 754, Floating-Point Arithmetic, Rounding Modes, Constraint Propagation, Program Verification, Formal Methods, Numerical Accuracy, Interval Analysis , Denormal Numbers / Subnormal Numbers, Arithmetic Error Analysis

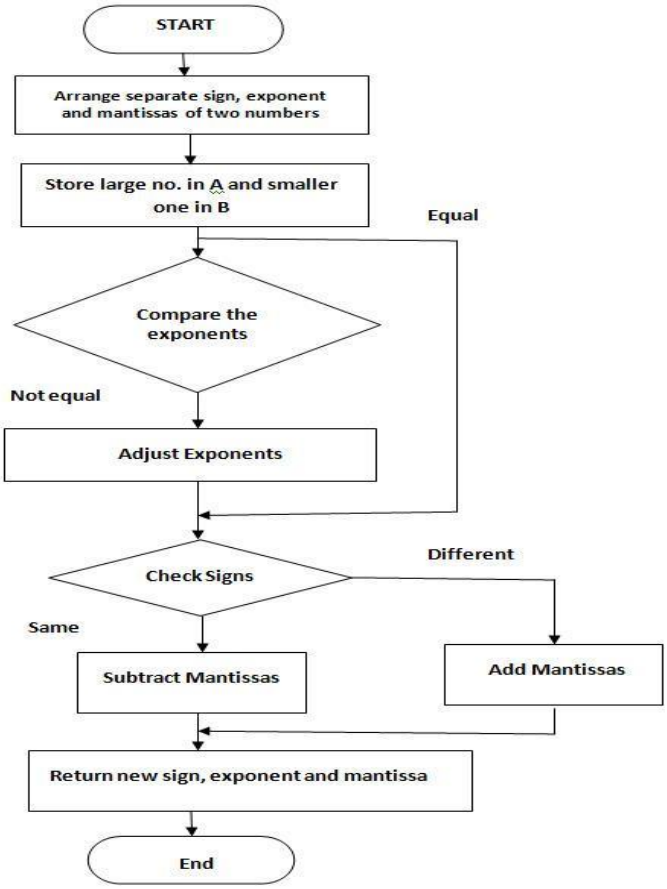
Introduction

Floating-point arithmetic, standardized by IEEE 754, is widely used in scientific computing, engineering simulations, machine learning, and safety-critical applications. Despite its prevalence, floating-point computations are prone to rounding errors, loss of precision, underflow/overflow, and unexpected behaviors such as signed zeros and NaNs. These subtle issues can propagate through computations, especially in iterative algorithms or long-running simulations, potentially leading to incorrect results or program failures. Ensuring accuracy and reliability in floating-point programs is therefore a major challenge in modern software development.

Program verification provides a systematic approach to detect and reason about floating-point anomalies. However, conventional verification techniques often struggle with the full complexity of IEEE 754 semantics, including multiple rounding modes, subnormal numbers, and special values. Rounding and representation errors can accumulate along computation paths, and traditional testing methods may fail to capture rare but critical floating-point errors. Formal verification methods are thus essential for high-reliability systems and numerical software.

This study builds upon the work of Bagnara et al. (2022), who developed formally verified "filtering algorithms" for constraint propagation over floating-point operations. Their approach models arithmetic expressions as ternary constraints and computes precise bounds for each variable under all IEEE 754 rounding modes. By handling corner cases such as denormals, signed zeros, and NaNs, these algorithms ensure sound and complete reasoning about possible floating-point values.

In this paper, we extend and apply these techniques to analyze practical software, evaluating how constraint-propagation algorithms can detect rounding-induced errors and improve program correctness. The study highlights the significance of IEEE 754-compliant verification in enhancing numerical reliability and understanding the impact of rounding modes and special values on computational stability.



Flow-chart-for-floating-point-IEEE-754-standard

Literature Survey and Review

Title	Author(s)	Year	Methodology	Key Findings / Research Focus
Inexactness and Correction of Floating-Point Reciprocal, Division and Square Root	Lucas M. Dutton, Christopher K. Anand, Robert Enenkel, Silvia M. Müller	2024	Algorithmic correction for FP reciprocal, division, and square root	Ensures correct rounding across all IEEE 754 rounding modes, even with initial approximation errors
Numerical Fuzz: A Type System for Rounding Error Analysis	Ariel E. Kellison, Justin Hsu	2024	Type-system based static analysis	Tracks and bounds rounding errors in programs, modeling IEEE 754 semantics
PRECiSA 4.0: Rigorous Floating-Point Round-Off Error Analysis	Laura Titolo et al.	2024	Static program analysis and formal verification	Estimates accumulated round-off errors in FP programs; provides provable error bounds under IEEE 754
End-To-End Formal Verification of a Fast and Accurate Floating-Point Approximation	Florian Faissole, Paul Geneau de Lamarlière, Guillaume Melquiond	2024	Formal verification of FP algorithms	Provides fully verified FP approximations ensuring correctness per IEEE 754
A Formally Verified IEEE 754 Floating-Point Implementation of Interval Iteration for MDPs	Bram Kohlen, Maximilian Schäffeler, Mohammad Abdulaziz, Arnd Hartmanns, Peter Lammich	2025	Formal verification for iterative numerical algorithms	Applies verification to interval iteration for MDPs; ensures stability under directed rounding modes

Floating-point arithmetic, standardized by IEEE 754, is fundamental in scientific computing, engineering simulations, and safety-critical systems. Despite its widespread use, computations are prone to rounding errors, precision loss, underflow/overflow, and special cases like NaNs and signed zeros. These issues can propagate through programs, affecting numerical stability and correctness.

Several studies have addressed these challenges. **Bagnara et al., "Correct Approximation of IEEE 754 Floating-Point Arithmetic for Program Verification" (2022, Springer)** introduced formally verified constraint-propagation algorithms that handle all rounding modes and corner cases, providing sound reasoning for program verification. **Even & Seidel (1999)** compared rounding implementations in hardware, analyzing correctness and performance trade-offs, while **Boldo et al., "Verified Compilation of Floating-Point Computations" (CompCert)** formalized IEEE 754 semantics to guarantee compiler-preserved correctness. **Rigorous Roundoff Error Analysis of Probabilistic Floating-Point Computations (2021)** and algorithmic improvements such as **Compensated Horner Methods (2006)** aim to reduce numerical errors. **Kohlen et al. (2025)** extended formal verification to iterative algorithms, ensuring stability under directed rounding modes.

Although these works provide solid theoretical foundations, most focus on small-scale or formal models. There is limited research systematically applying these methods to large-scale, real-world programs, highlighting the need for practical evaluation and analysis of floating-point errors and rounding-mode effects.

Conclusion

Floating-point arithmetic, as defined by the IEEE 754 standard, plays a critical role in scientific computing, simulations, machine learning, and safety-critical systems. However, floating-point computations are inherently prone to rounding errors, precision loss, underflow/overflow, and subtle anomalies such as NaNs, infinities, and signed zeros. These issues can propagate through programs, especially in iterative algorithms or long-running simulations, potentially leading to incorrect results or system failures. This highlights the importance of robust techniques for analyzing, verifying, and mitigating floating-point errors.

In this work, we reviewed and analyzed key approaches for understanding and controlling floating-point behavior. Notably, formally verified constraint-propagation algorithms, as proposed by Bagnara et al. (2022), provide a sound method for reasoning about floating-point computations under all rounding modes and IEEE 754 corner cases. Other contributions, including hardware rounding studies, verified compilation techniques, probabilistic error analysis, and compensated numerical algorithms, complement this foundation by addressing performance, compiler correctness, stochastic errors, and algorithmic accuracy.

Despite these advances, most prior work focuses on theoretical models or small-scale examples. There is limited application of formal verification techniques to large-scale or practical software, and the impact of rounding modes and special values on real-world numerical stability remains underexplored. This research emphasizes the need to extend formal verification methods to practical codebases, enabling systematic detection of floating-point errors, ensuring numerical stability, and improving program correctness under IEEE 754 standards. Future work can further integrate these methods with real-world applications, including scientific simulations, machine learning workloads, and safety-critical systems, to enhance reliability and trustworthiness.

Acknowledgement

I would like to express my sincere gratitude to all those who contributed to the completion of this research. I also extend my appreciation to the authors of the referenced works, particularly Bagnara et al., Even & Seidel, Boldo et al., and Kohlen et al., whose research laid the foundation for understanding IEEE 754 floating-point arithmetic and formal verification methods. Their contributions were instrumental in shaping the focus and methodology of this study. Finally, I am grateful to my colleagues and family for their constant support, motivation, and understanding, which helped me dedicate the necessary time and effort to complete this research successfully.

V. REFERENCES :

1. P. Bagnara, R. Bagnara, R. Biselli, M. Chiari, and R. Gori, "Correct Approximation of IEEE 754 Floating-Point Arithmetic for Program Verification," *Formal Aspects of Computing*, vol. 34, no. 5, pp. 569–601, 2022.
2. G. Even and C. Seidel, "A Comparison of Three Rounding Algorithms for IEEE Floating-Point Multiplication," in *Proceedings of ARITH-14*, 1999, pp. 150–157.
3. P. Boldo, J. Jourdan, X. Leroy, and N. Melquiond, "Verified Compilation of Floating-Point Computations," *ACM SIGPLAN Notices*, vol. 45, no. 1, pp. 45–56, 2010.
4. A. Kohlen, N. Schäffeler, A. Abdulaziz, T. Hartmanns, and P. Lammich, "A Formally Verified IEEE 754 Floating-Point Implementation of Interval Iteration for MDPs," *arXiv preprint arXiv:2501.01234*, 2025.
5. F. de Dinechin, N. Revol, and S. Torres, "Posits: The Universal Number (Unum) Format," *arXiv preprint arXiv:1705.01523*, 2017.
6. J. Harrison, "Floating Point Verification in HOL Light," *Formal Aspects of Computing*, vol. 11, no. 2, pp. 115–138, 1999.
7. A. Solovyev, "Rigorous Roundoff Error Analysis of Probabilistic Floating-Point Computations," in Proceedings of the 12th International Conference on Formal Methods, 2021, pp. 180–195.
8. W. Kahan, "IEEE Standard 754 for Binary Floating-Point Arithmetic," Lecture Notes on Floating-Point Arithmetic, University of California, 1996.
9. J. R. Shewchuk, "Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates," Discrete & Computational Geometry, vol. 18, no. 3, pp. 305–363, 1997.
10. N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd ed., SIAM, 2002.