Testing interval arithmetic libraries

Nathalie Revol[1], Luis Benet Fernández[2], Luca Ferranti[3] and Sergei Zhilin[4]

[1] INRIA - LIP UMR 5668, ENS Lyon, University Lyon 1, Inria, CNRS - France
Nathalie.Revol@inria.fr (presenting author)
[2] Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México
benet@icf.unam.mx
[3] University of Vaasa, Vaasa, Finland
luca.ferranti@uwasa.fi
[4] CSort LLC, Barnaul, Russia
szhilin@gmail.com

Keywords: Interval Arithmetic, Library, Test, Unit Test

Testing interval arithmetic libraries: why

Interval arithmetic is used to get guarantees on numerical results. Indeed, it provides an anclosure of the sought result. However, the user must trust the library implementing interval arithmetic that is employed to solve the given problem. What guarantees that this library is correct? Formal proof is a desirable approach, and in particular it is available within the CoqInterval library [1]. Another, complementary, approach consists in testing the library: it covers aspects that are usually not covered by formal proof, such as the specifics of the language, compiler (with the notable exception of CompCert [2]) or hardware.

Unit tests

In what follows, only unit tests will be considered, and tests that are complete applications will not be discussed, see [3] for a first step in this direction. Unit tests target only one function of the library at a time; typically, they consist of a list of test cases, that is, of input values along with the expected output values: one checks whether the

function returns the expected output values for each input arguments. If this is the case, the function passes the test.

Testing interval arithmetic libraries: how

The goal of this talk is to discuss the different aspects that unit tests should cover, and how to devise corresponding test cases, with a specific focus on compliance with the IEEE 1788-2015 standard for interval arithmetic [4]. The ultimate goal would be to create a collection of test cases for each function required or recommended by this standard, and to share them. We emphasize that this collection should be easy to use for libraries written in different programming languages, such as MPFI [5] written in C, libieee1788 [6] written in C++, JInterval [7] written in Java, Intlab [8] available in MatLab, Octave/interval [9] written in Octave or JuliaIntervals/IntervalArithmetic.jl [10] written in Julia. We will survey two approaches in this direction, namely JInterval [11] and ITF-1788 [12], and discuss their limitations. An even more desirable goal would be to design a generator of test cases, we will discuss this point as well.

References

[**1**] E. Martin-Dorel and G. Melquiond, *Proving tight bounds on univariate expressions with elementary functions in Coq*, Journal of Automated Reasoning 57(3), pp. 187–217 (2016).
`https://coqinterval.gitlabpages.inria.fr`

[**2**] X. Leroy et al., *CompCert-a formally verified optimizing compiler*, in ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress.
`https://compcert.org`

[**3**] X. Tang, Z. Ferguson, T. Schneider, D. Zorin, S. Kamil, D. Panozzo, *A Cross-Platform Benchmark for Interval Computation Libraries*, arXiv 2021. `https://arxiv.org/abs/2110.06215`

[4] IEEE: Institute of Electrical and Electronic Engineers, *1788-2015 - IEEE Standard for Interval Arithmetic.*

[5] N. Revol and F. Rouillier, *Motivations for an Arbitrary Precision Interval Arithmetic and the MPFI Library*, Reliable Computing **11**(4), pp. 275–290 (2005).
https://gitlab.inria.fr/mpfi/mpfi

[6] M. Nehmeier, *libieeep1788: A C++ Implementation of the IEEE interval standard P1788*, in 2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW), pp. 1–6.
https://github.com/nehmeier/libieeep1788

[7] D.Y. Nadezhin and S.I. Zhilin, *JInterval Library: Principles, Development, and Perspectives*, Reliable Computing **19**(3), pp. 229–247 (2013).
https://github.com/jinterval/jinterval/

[8] S.M. Rump, *INTLAB - INTerval LABoratory*, in Developments in Reliable Computing, Tibor Csendes (ed), pp. 77–104. Kluwer Academic Publishers (1999).
https://www.tuhh.de/ti3/rump/intlab/

[9] O. Heimlich, *Interval arithmetic in GNU Octave*, in SWIM 2016, 9th Summer Workshop on Interval Methods, Lyon, France.
https://octave.sourceforge.io/interval/index.html

[10] D.P. Sanders and L. Benet Fernández, *JuliaIntervals/-IntervalArithmetic.jl: v0.20.5*, Zenodo, DOI 10.5281/zenodo.6337817
https://github.com/JuliaIntervals/ValidatedNumerics.jl

[11] *P1788 Test Launcher (based on JInterval Library).*
https://github.com/jinterval/jinterval/tree/master/p1788-launcher-java

[12] M. Kiesner, M. Nehmeier, and J. Wolff von Gudenberg, *ITF1788: An Interval Testframework for IEEE 1788*, Report no 495, Dpt Computer Science, University of Würzburg (2015).
https://github.com/oheim/ITF1788