Master Internship Proposal

Numerical Analysis and Optimization of Functional Array Programs

Location	Department of Information Technology, Uppsala, Sweden or Team Inria CAMUS, ICube Laboratory, Strasbourg (Illkirch campus), France
Advisors	Eva Darulova, Uppsala University Thomas Kœhler, CNRS

1 Context

Computing frequently involves approximating exact arithmetic using finite precision number representations: e.g. floating-point numbers and fixed-point arithmetic. However, achieving the desired numerical precision in finite precision programs while maximizing performance is challenging due to overflow and roundoff errors.

In the optimizing compiler community, floating points are often treated as reals (e.g. gcc's `-ffast-math`), enabling useful transformations ($x + (y + z) = (x + y) + z$, $x/10.0 = 0.1 * x$, etc), but ignoring accuracy problems.

In the program analysis community, tools were developed to analyze and improve numerical software [1, 2]. Many tools tune the precision of arithmetic variables and operations in a given program (*mixed-precision tuning*), potentially rewriting arithmetic expressions [3, 4, 5]. Beyond straight-line code, Rosa [6] and Precisa [7] can analyse rounding errors over certain kinds of conditionals and loops. Analysing rounding errors of programs written over arrays has only been made possible recently in the Daisy DS2L [8]. The analysis of Daisy DS2L scales by avoiding to unroll array operations and leveraging high-level information from its functional array input language that features high-level patterns over arrays (e.g. map, reduce). However, the functional array code of Daisy DS2L is currently executed without applying state-of-the-art compiler optimizations.

The goal of this internship is to demonstrate for the first time that it is possible to perform both state-of-the-art compiler optimizations and numerical analyses on array code, in order to produce fast programs with guaranteed error bounds. To achieve this, the Daisy DS2L rounding error analysis will be combined with recent work on the Shine compiler [9, 10]. The Shine compiler takes as input programs defined in the functional array language called Rise, and outputs imperative code (e.g. C, OpenCL, or CUDA). Optimizations are not hard-coded, but expressed as Rise rewrite rules that can be applied more or less automatically (via *rewriting strategies* [11] or *equality saturation* [12]).

2 Objectives

We envision the following steps for the internship:

1. Implement translations between the Daisy DS2L and Rise languages. While the Daisy DS2L and Rise languages are different, they are both functional array languages based on similar patterns, and the translation appears feasible in many cases.

2. Use these translations to first analyse input programs using Daisy DS2L before optimizing them using Rise rewrite rules that preserve floating-point behaviour and generating C code using Shine. Evaluate the performance of the generated C code on the Daisy DS2L benchmarks, and experimentally validate the computed error bounds.

3. Ideally, explore the use of Rise rewrite rules that do not preserve floating-point behaviour. For example, by applying them before the Daisy DS2L analysis in the previously process. Floating-point sums [13], reductions [14] and scans [15] would be interesting benchmarks as their parallelisation requires associativity properties found in reals but not in floating-points.

3 Required and Acquired Skills

The intern should come with:

- solid programming skills, familiarity with imperative and functional paradigms
- interest in (or experience with) analysis, compilation and optimization tools
- interest in (or experience with) functional array languages, map/reduce patterns
- willingness to learn (or knowledge of) Scala and C

The intern is expected to acquire:

- experience in basic academic skills: reviewing literature, conducting novel research, collaborating with other researchers, presenting research and communicating ideas
- knowledge in floating-point analysis, term rewriting and code generation
- the ability to combine theory (numerical analysis) with practice (fast imperative code)

4 Practical Aspects

The internship can either take place in Uppsala (Sweden), or in Strasbourg (France), and a brief visit of the other location could be organized during the internship. If you are interested, send us an e-mail with a short statement of interest, questions you might have, the desired start date and duration of your internship (4-6 months), as well as a 1 page CV and your Master transcripts. If the internship goes well, there is potential to pursue a PhD in similar topics.

References

[1] Stefano Cherubin and Giovanni Agosta.
"Tools for reduced precision computation: a survey".
In: *ACM Computing Surveys (CSUR)* 53.2 (2020), pp. 1–35.

[2] URL: https://fpbench.org/community.html.

[3] Pavel Panchekha, Alex Sanchez-Stern, James R Wilcox, and Zachary Tatlock.
"Automatically improving accuracy for floating point expressions".
In: *ACM SIGPLAN Notices* 50.6 (2015), pp. 1–11.

[4] Eva Darulova, Anastasiia Izycheva, Fariha Nasir, Fabian Ritter, Heiko Becker, and Robert Bastian.
"Daisy - Framework for Analysis and Optimization of Numerical Programs (Tool Paper)".
In: *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I.*
Ed. by Dirk Beyer and Marieke Huisman. Vol. 10805. Lecture Notes in Computer Science.
Springer, 2018, pp. 270–287. URL: https://doi.org/10.1007/978-3-319-89960-2%5C_15.

[5] Nasrine Damouche.
"Improving the Numerical Accuracy of Floating-Point Programs with Automatic Code Transformation Methods. (Amélioration de la précision numérique de programmes basés sur l'arithmétique flottante par les méthodes de transformation automatique)".
PhD thesis. University of Perpignan, France, 2016.
URL: https://tel.archives-ouvertes.fr/tel-01455727.

[6] Eva Darulova and Viktor Kuncak. "Towards a Compiler for Reals".
In: *ACM Trans. Program. Lang. Syst.* 39.2 (2017), 8:1–8:28.
URL: https://doi.org/10.1145/3014426.

[7] Laura Titolo, Marco A. Feliú, Mariano M. Moscato, and César A. Muñoz. "An Abstract Interpretation Framework for the Round-Off Error Analysis of Floating-Point Programs". In: *Verification, Model Checking, and Abstract Interpretation - 19th International Conference, VMCAI 2018, Los Angeles, CA, USA, January 7-9, 2018, Proceedings.* Ed. by Isil Dillig and Jens Palsberg. Vol. 10747. Lecture Notes in Computer Science. Springer, 2018, pp. 516–537. URL: https://doi.org/10.1007/978-3-319-73721-8%5C_24.

[8] Anastasia Isychev and Eva Darulova. "Scaling up Roundoff Analysis of Functional Data Structure Programs". In: *International Static Analysis Symposium*. Springer. 2023, pp. 371–402.

[9] Michel Steuwer, Thomas Kœhler, Bastian Köpcke, and Federico Pizzuti. *RISE & Shine: Language-Oriented Compiler Design*. 2022. arXiv: 2201.03611 [cs.PL].

[10] Thomas Kœhler. "A domain-extensible compiler with controllable automation of optimisations". In: *arXiv preprint arXiv:2212.12035* (2022).

[11] Bastian Hagedorn, Johannes Lenfers, Thomas Kœhler, Xueying Qin, Sergei Gorlatch, and Michel Steuwer. "Achieving High Performance the Functional Way: Expressing High-Performance Optimizations as Rewrite Strategies". In: *Communications of the ACM* 66.3 (2023), pp. 89–97.

[12] Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. "Egg: Fast and extensible equality saturation". In: *Proceedings of the ACM on Programming Languages* 5.POPL (2021), pp. 1–29.

[13] URL: https://orlp.net/blog/taming-float-sums/.

[14] Nikolay M. Evstigneev, Oleg I. Ryabkov, A. N. Bocharov, V. P. Petrovskiy, and I. O. Teplyakov. "Compensated summation and dot product algorithms for floating-point vectors on parallel architectures: Error bounds, implementation and application in the Krylov subspace methods". In: *J. Comput. Appl. Math.* 414 (2022), p. 114434. URL: https://doi.org/10.1016/j.cam.2022.114434.

[15] Ivo Gabe de Wolff, David P. van Balen, Gabriele K. Keller, and Trevor L. McDonell. "Zero-Overhead Parallel Scans for Multi-Core CPUs". In: *Proceedings of the 15th International Workshop on Programming Models and Applications for Multicores and Manycores, PMAM 2024, Edinburgh, United Kingdom, 3 March 2024*. ACM, 2024, pp. 52–61. URL: https://doi.org/10.1145/3649169.3649248.