PhD/Postdoc: First Class Compilers with Partial Evaluation and Analytic Macros

Gabriel Radanne – Inria/LIP

2024

1 Context

Domain-Specific Languages (DSL) are instrumental in exploring new complex use cases through novel programming techniques. DSLs have been at a forefront of recent computer science innovations, from blockchains (contract languages [4]) to quantum computing (Quipper [2], Q#, ...), including machine learning (TensorFlow [1], ...). For High Performance Computing (HPC), DSLs have been largely used to describe complete programs (High Level Synthesis, synchronous programming, dataflows, ...), computational kernels (CUDA, ...), or hardware circuits.

Furthermore, HPC programs are highly domain-specific, from hardware accelerators for audio/video processing to computational kernels for machine learning and financial applications. These domains require different abstractions and might benefits from custom optimisations that could be provided directly by tailor-made HPC DSLs.

Unfortunately, designing such custom optimisations is a difficult task. It requires extending an existing compiler, or even more difficult: creating a new language from scratch. This has lead most HPC tools to use dialects of C as source-language equipped with compiler annotations, such as CUDA, openMP or most High Level Synthesis tools. While the common use of C has eased adoption, it offers very poor understandability, as such DSLs re-use the syntax of C with different semantics, and are still very difficult to use, prototype, compose and distribute, as they involve modification of existing compilers. Approaches based on meta-programming in C++, while popular, suffers from very poor usability and safety.

This project aims to propose *first class optimisations*, which allow to define optimisations and program transformations as first class objects of the language, similarly to function. Such first class optimisations can be defined by the user for a specific purpose, composed with other transformations, and then packaged as libraries and distributed to other users of the language.

2 Objective

AnyDSL [3] aims to ease the creation of embedded domain specific language by exploiting *partial-evaluation*. This offers a lightweight, easy-to-use interface for programmers, where the only special constructs is an annotation informing the compiler to unfold a definition. For instance, the `pow` function is declared as follows: `fn @(?n) pow (x: i32 , n: i32 ) -> i32`. It takes two integers x and n and returns an integer (i.e. x^n). In addition, the annotation `@(?n)` indicate that if n is known statically, then the definition of `pow` is unfolded during compilation.

Turns out, this seemingly simple construct is enough to implement extremely efficient domain specific libraries, ranging from sequence alignment [5] to ray-tracers [7], which outperform existing hand-tuned implementation.

Nevertheless, developing new DSL which relies on code transformation is difficult endeavor. Indeed, code transformation relies on inspecting pieces of program (in an appropriate intermediate representation) to emit new pieces of code. Partial evaluation simply annotates what should and shouldn't be evaluated during compilation. The latter is easier to use, but the former is more powerful, making it the method of choice for compilation algorithms.

In this project, we aim to bring the two approaches together by combining *analytic macros* [6] with partial evaluation. Analytic macros would allow to define rich code rewriting, which would be leveraged by AnyDSL's compile-time code evaluation, allowing to easily define complex code transformation packaged as easy-to-use domain specific libraries.

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / INRIA – 69007 Lyon
E-mail : Gabriel.Radanne@inria.fr

3 Program

The candidate may follow the following steps (some of the steps are independent and could be given relative priority based on subsequent findings and the candidate's preference):

- Develop a formal system with both partial evaluation and code rewriting. This system aims to develop a good understanding of the interaction between the two features on paper
- Implement code rewriting in AnyDSL, in collaboration with the AnyDSL team at DFKI-Saarbrücken
- Experiment on concrete domains, such as:
  - Auto-Scheduling
  - Data Layout Optimisations
  - High Level Synthesis and Stencils
  - Others, following the candidate's interest

Location and Supervising This project will take place in LIP, Lyon, France. It will be supervised by Gabriel Radanne, Inria researcher, specialist in compilation of high level languages. It will be made in collaboration with Richard Membarth, Professor at DFKI-Saarbrücken, Germany.

Candidate profile The candidate should ideally be familiar with formal approaches in programming languages and compilation. The candidate should also ideally have a taste for creating high performance programs.

From the practical point of view, a basic experience in software programming and usage of collaborative tools such that `git`. Knowledge of C++ would be preferable.

References

[1] Jeff Dean. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 2015.

[2] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In Hans-Juergen Boehm and Cormac Flanagan, editors, *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, pages 333–342. ACM, 2013.

[3] Roland Leißa, Klaas Boesche, Sebastian Hack, Arsène Pérard-Gayot, Richard Membarth, Philipp Slusallek, André Müller, and Bertil Schmidt. Anydsl: a partial evaluation framework for programming high-performance libraries. *Proc. ACM Program. Lang.*, 2(OOPSLA):119:1–119:30, 2018.

[4] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 254–269. ACM, 2016.

[5] André Müller, Bertil Schmidt, Richard Membarth, Roland Leißa, and Sebastian Hack. Anyseq/gpu: a novel approach for faster sequence alignment on gpus. In Lawrence Rauchwerger, Kirk W. Cameron, Dimitrios S. Nikolopoulos, and Dionisios N. Pnevmatikatos, editors, *ICS '22: 2022 International Conference on Supercomputing, Virtual Event, June 28 - 30, 2022*, pages 20:1–20:11. ACM, 2022.

[6] Lionel Parreaux, Antoine Voizard, Amir Shaikhha, and Christoph E. Koch. Unifying analytic and statically-typed quasiquotes. *PACMPL*, 2(POPL):13:1–13:33, 2018.

[7] Arsène Pérard-Gayot, Richard Membarth, Roland Leißa, Sebastian Hack, and Philipp Slusallek. Rodent: generating renderers without writing a generator. *ACM Trans. Graph.*, 38(4):40:1–40:12, 2019.