Scheduling data structures with term rewriting techniques

Laure Gonnord, Université Lyon1/LIP
Carsten Fuhs, Birkbeck, University of London

2020

Keywords

Abstract interpretation, compilation, GPU, scratchpad memory, numerical abstract domains.

Location and supervision

The internship will take place in LIP, Lyon, coadvised by Laure Gonnord and remotely with Carsten Fuhs.

General Context : CODAS project

The long term objective of the ANR CODAS project is to give a general way to reason about and manipulate programs with general control flow and complex data structures.

A PhD is ongoing on the definition of a polyhedral model that include trees, but it has not explored the relationship between inductive data structures and term rewriting. In this internship we want to explore this idea.

Indeed, the problem of scheduling is closely linked to the well-studied topic of termination. This motivates the investigation of the benefits we may encounter from the rewriting community :

— Automated termination analysis for *term rewrite systems (TRSs)* [2] is a well-studied topic and has given rise to several powerful, fully automatic, tools over the last years (e.g., AProVE, Mu-Term, TTT2). TRSs are particularly suitable to represent complex data structures, such as lists or trees.

— Two-stage approaches to termination analysis, harnessing the power of termination tools for TRSs also for programming languages, have been proposed [5, 7, 6]. In the first stage, symbolic execution and abstraction on programming language level are used to over-approximate all possible program executions. From this program analysis, one then extracts a TRS whose termination implies the termination of the original program. In the second stage, termination of this TRS is analyzed with existing tools.
This approach is currently limited mainly by scalability of the program analysis front-ends.

— More recently Fuhs et al. [4] proposed an extension of term rewriting by *built-in predefined integers*, based on the observation that termination, in particular for imperative programs, often depends on integer variables. This allows existing techniques for proving termination on integers to be combined with techniques for term rewriting. Tools for analysis of term rewriting are now successfully applied for termination analysis (e.g., for Java [7, 3] and C programs [9]).

The preliminary paper [1] describes this relationship between schedules and termination proofs, and derives an estimation on the *parallel complexity* of the given programs. This paper can be viewed as a first step of the current proposal.

Internship content

Steps of the internship :

— In the first part of the internship, a high level langage with inductive data structures, à la ML, will be defined, and some benchmarks will be designed in order to further validate our approach.

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / Inria – 69007 Lyon
Birkbeck, University of London
E-mail : Laure.Gonnord@ens-lyon.fr and Carsten@dcs.bbk.ac.uk

— The second step is to define what the notion of a scheduling means for this semantics. Formally, we aim to express the notion of dependence, like the “happens-before” relation of the polyhedral model. This notion of scheduling will be used to construct a dependence graph of a given program under analysis. The intern will explore how the notion of position (in classical term rewriting) [] could be used to define the notion of dependance. (S)He might also get some inspiration from the parallelization of functional programs proposed in [8].
— A first prototype will be implemented.

Références

[1] Christophe Alias, Carsten Fuhs, and Laure Gonnord. Estimation of Parallel Complexity with Rewriting Techniques. In *Proceedings of the 15th Workshop on Termination*, WST ’16, pages 2 :1–2 :5, Obergurgl, Austria, September 2016.

[2] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[3] Marc Brockschmidt, Richard Musiol, Carsten Otto, and Jürgen Giesl. Automated termination proofs for Java programs with cyclic data. In *Proceedings of the 24th International Conference on Computer Aided Verification*, CAV ’12, pages 105–122, 2012.

[4] Carsten Fuhs, Jürgen Giesl, Martin Plücker, Peter Schneider-Kamp, and Stephan Falke. Proving termination of integer term rewriting. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, RTA ’09, pages 32–47, 2009.

[5] Jürgen Giesl, Matthias Raffelsieper, Peter Schneider-Kamp, Stephan Swiderski, and René Thiemann. Automated termination proofs for Haskell by term rewriting. *ACM Transactions on Programming Languages and Systems*, 33(2) :7 :1–7 :39, 2011.

[6] Jürgen Giesl, Thomas Ströder, Peter Schneider-Kamp, Fabian Emmes, and Carsten Fuhs. Symbolic evaluation graphs and term rewriting : A general methodology for analyzing logic programs. In *Proceedings of the 14th International Symposium on Principles and Practice of Declarative Programming*, PPDP ’12, pages 1–12, 2012.

[7] Carsten Otto, Marc Brockschmidt, Christian von Essen, and Jürgen Giesl. Automated termination analysis of Java Bytecode by term rewriting. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, RTA ’10, pages 259–276, 2010.

[8] Rodrigo C. O. Rocha, Luís Fabrício Wanderley Góes, and Fernando Magno Quintão Pereira. An algebraic framework for parallelizing recurrence in functional programming. In *Proceedings of the 20th Brazilian Symposium on Programming Languages*, SBLP ’16, pages 140–155, 2016.

[9] Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, Peter Schneider-Kamp, and Cornelius Aschermann. Automatically proving termination and memory safety for programs with pointer arithmetic. *Journal of Automated Reasoning*, 58(1) :33–65, 2017.