Internship proposals

Certification of program termination

Frédéric Blanqui (INRIA)

http://www-rocq.inria.fr/~blanqui

Place:

The internship will take place at Tsinghua University[1] (Beijing, China) within the INRIA[2] project-team FORMES[3] which is part of the LIAMA[4] Consortium, the Sino-French Laboratory for Computer Science, Automation and Applied Mathematics. The members of the FORMES project-team also include Pr Ming Gu, Dr Fei He, Dr Claude Helmstetter, Pr Vania Joloboff, Pr Jean-Pierre Jouannaud, Pr Jean-François Monin, Dr Pierre-Yves Strub and Dr Bow-Yaw Wang.

Introduction:

Termination is an important property of programs. For a given program, it may even be necessary to establish its termination in order to prove its correctness. For this undecidable problem, many criteria and tools have been developed over the last years. But these become more and more complex and difficult to verify. To recover the confidence one can expect from such tools, it is necessary to certify their results. For stimulating the research in this area, the steering committee of the international competition on termination[5] organizes a competition for certified termination provers since 2007.

The CoLoR project[6] aims at providing tools for certifying the results of automated provers. The tools developed in this project, CoLoR and Rainbow, are currently used by four different termination provers: AProVE (best 2007 termination tool for TRSs), Matchbox (best 2007 termination tool for SRSs), TPA and TTT2. And CoLoR+Rainbow was the best certification back-end for the last two years in the international competition on termination[7].

The approach taken in CoLoR is as follows. It is based on two important elements. First, a grammar for termination certificates (TCG) (currently implemented as an XML Schema). Second, a Coq library of the termination techniques used in the grammar. Coq[8] is a highly secure proof checker and proof development tool which allows one to reach the highest security standards

[1] http://www.thss.tsinghua.edu.cn/index_en.asp
[2] http://www.inria.fr
[3] http://formes.asia
[4] http://liama.ia.ac.cn
[5] http://termination-portal.org
[6] http://color.inria.fr
[7] http://termination-portal.org/wiki/Termination_Competition
[8] http://coq.inria.fr

(Common Criteria EAL7 level).

The certification chain is then as follows. Termination tools provide a certificate in TCG (XML file). The Rainbow program generates then a Coq file from this certificate using the Coq library CoLoR. Finally, the Coq compiler is called to check the correctness of the termination certificate.

Many powerful termination criteria have been developed for rewrite rules based programs [4]. Rewriting is a fundamental notion that is also of practical interest since programs in other programming paradigms can be encoded by using rewrite systems. Currently, CoLoR and Rainbow can handle rewrite systems only. The following subjects propose to extend CoLoR and Rainbow with other important termination techniques and to other programming paradigms.

Subjects:

1. Rainbow certification and efficient proof checking.

   One way to improve proof checking efficiency and reduce the risk of errors that the program converting TCG to Coq could itself introduce, is to formalize this program in Coq and use the Coq extraction mechanism to get an OCaml program that can in turn be compiled and run independently of Coq.

2. Modular and parallel proof checking.

   Another way to improve proof checking efficiency, but inside Coq, is to prove each TCG step as a separate lemma (the verification of each lemma can then be done in parallel).

3. Usable rules.

   Among the various techniques used by the current state-of-the-art provers, an important one is the usable rules in the dependency pair framework [6]. We propose to extend CoLoR and Rainbow with this notion.

4. Rewriting under strategy.

   General rewriting imposes no specific stragegy for applying rules. Some particular strategies (innermost, outermost, context sensitive) are often used in practice. For instance, innermost rewriting corresponds to the usual evaluation strategy in most common programming languages, where the arguments of a function are computed before calling it, and innermost termination implies termination for an important class of rewrite systems [7]. We propose to formalize in Coq termination certificates for innermost rewriting.

5. Haskell programs.

   AProVE[9] can prove the termination of Haskell[10] programs [8] by adapting the technique of dependency pairs to Haskell programs [5]. We propose

[9] `http://aprove.informatik.rwth-aachen.de`
[10] `http://www.haskell.org`

to define and prove in Coq a notion of termination certificate for Haskell programs based on this work.

6. Prolog programs.

   AProVE[11] can prove the termination of Prolog[12] programs[13] by adapting the technique of dependency pairs to logic programs [10, 11]. We propose to define and prove in Coq a notion of termination certificate for Prolog programs based on this work.

Prerequisites:
Some knowledge on rewriting theory [4], dependency pairs [1, 12], functional programming (Rainbow is written in OCaml [9]) and Coq [2] is recommended. CoLoR, Rainbow and the papers related to them can be downloaded on `http://color.inria.fr`. See in particular [3].

References

[1] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[2] Y. Bertot and P. Castéran. *Coq'Art: The Calculus of Inductive Constructions*. EATCS Texts in Theoretical Computer Science. Springer, 2004.

[3] F. Blanqui and A. Koprowski. Automated verification of termination certificates. Technical Report 6949, INRIA Rocquencourt, France, 2009.

[4] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6. North-Holland, 1990.

[5] J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated termination analysis for haskell: From term rewriting to programming languages. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 4098, 2006.

[6] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[7] B. Gramlich. On proving termination by innermost termination. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 1103, 1996.

[11] `http://aprove.informatik.rwth-aachen.de`
[12] `http://en.wikipedia.org/wiki/Prolog`
[13] `http://en.wikipedia.org/wiki/Logic_programming`

[8] S. P. Jones and all. *Haskell 98 Language and Libraries, The revised report.* Cambridge University Press, 2003.

[9] X. Leroy and P. Weis. *Le langage Caml (Seconde édition).* Dunod, 1999.

[10] M. T. Nguyen, J. Giesl, P. Schneider-Kamp, and D. De Schreye. Termination analysis of logic programs based on dependency graphs. In *Proceedings of the 17th International Symposium on Logic-Based Program Synthesis and Transformation*, Lecture Notes in Computer Science 4915, 2007.

[11] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated termination proofs for logic programs by term rewriting. *ACM Transactions on Computational Logic*, ?(?):?–?, 2008. To appear.

[12] R. Thiemann. *The DP Framework for Proving Termination of Term Rewriting.* PhD thesis, RWTH Aachen University, Germany, 2007.