Automatically Analysing Graph Programs for Termination and Complexity

PhD Project Proposal

Detlef Plump

This project will contribute to the departmental research theme Critical Systems in that it develops automatic analyses for the termination and time complexity of safety-critical programs working on graph-like structures. This class of programs includes, in particular, pointer manipulating programs in languages such as C which are notoriously error-prone. Pointer programs can be modelled by graph transformation rules and then analysed for their properties [3, 2].

This project will model pointer programs in the rule-based graph programming language GP 2 [6]. The language has a simple syntax and semantics which facilitates formal reasoning (see [9] for a Hoare logic approach to verifying graph programs). Deciding termination of arbitrary graph programs is impossible as GP 2 is computationally complete [7], hence the goal are static analysers that can automatically check termination and infer complexity bounds for large classes of programs.

Such analysers will be developed by pursuing two lines of research. Firstly, there exist powerful techniques for proving termination of term rewriting systems [1] which are ultimately based on Kruskal's Tree Theorem. To obtain similar methods for graph programs, the Tree Theorem has to be replaced with the seminal Robertson-Seymour Theorem, which states that graphs are well-quasi ordered by the minor ordering [10]. The goal is then, in analogy to term rewriting, to derive syntactic termination orders that can be automated. In addition, this line of research will adapt recent techniques for proving termination of graph transformation systems to graph programs. This includes the method of [4] and the modularity criterion of [8].

The second line of research will aim to adapt to graph programs recently developed automatic termination analysers for languages such as Java, C and Haskell [5]. These tools transform programs into term rewriting systems and then apply a range of techniques for proving termination of rewriting. Moreover, this approach allows to automatically derive bounds for time complexity (usually polynomials) from successful termination proofs.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] A. Bakewell, D. Plump, and C. Runciman. Checking the shape safety of pointer manipulations. In *Int. Seminar on Relational Methods in Computer Science (RelMiCS 7), Revised Selected Papers*, volume 3051 of *Lecture Notes in Computer Science*, pages 48–61. Springer-Verlag, 2004.

[3] A. Bakewell, D. Plump, and C. Runciman. Specifying pointer structures by graph reduction. In *Applications of Graph Transformations With Industrial Relevance (AGTIVE 2003), Revised Selected and Invited Papers*, volume 3062 of *Lecture Notes in Computer Science*, pages 30–44. Springer-Verlag, 2004.

[4] H. S. Bruggink, B. König, and H. Zantema. Termination analysis for graph transformation systems. In *Proc. Theoretical Computer Science (TCS 2014)*, volume 8705 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2014.

[5] J. Giesl, C. Aschermann, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, J. Hensel, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, and R. Thiemann. Analyzing program termination and complexity automatically with AProVE. *Journal of Automated Reasoning*, 58(1):3–31, 2017.

[6] D. Plump. The design of GP 2. In *Proc. Workshop on Reduction Strategies in Rewriting and Programming (WRS 2011)*, volume 82 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–16, 2012.

[7] D. Plump. From imperative to rule-based graph programs. *Journal of Logical and Algebraic Methods in Programming*, 88:154–173, 2017.

[8] D. Plump. Modular termination of graph transformation. In R. Heckel and G. Taentzer, editors, *Festschrift in Memory of Hartmut Ehrig*, Lecture Notes in Computer Science. Springer, 2017. To appear.

[9] C. M. Poskitt. *Verification of Graph Programs*. PhD thesis, University of York, 2013.

[10] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.