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


