Verification of Mutual Recursive Functional Programs

Nikolaj Popov, Tudor Jebelean*

Research Institute for Symbolic Computation,
Johannes Kepler University, Linz, Austria
{popov,jebelean}@risc.uni-linz.ac.at

Abstract. We present an environment for proving total correctness of mutual recursive functional programs. As usual, correctness is transformed into a set of first-order predicate logic formulae—verification conditions. As a distinctive feature of our method, these formulae are not only sufficient, but also necessary for the correctness. A specialized strategy for proving termination is developed. The detailed termination proofs may in many cases be avoided due to their reusability.

1 Introduction

We develop a method for the generation of verification conditions for proving total correctness of mutual recursive functional programs. Our focus is on the generation of the conditions, and we do not treat here the problem of discharging them. We assume that the specification and the program are provided and our task is to generate sound and complete verification conditions. (In fact, we believe that specifications should be developed before the program is written.)

Mutual recursion is a special form of recursion where two programs are defined in terms of each other. Moreover, the two programs form a system, and its solution is the computable function which is defined by the programs.

We study the class of simple mutual recursive programs and we extract the purely logical conditions which are sufficient for the program correctness. They are inferred using Scott induction and induction on natural numbers in the fixpoint theory of functions and constitute a meta-theorem.

As usual, correctness is transformed into a set of first-order predicate logic formulae – verification conditions. As a distinctive feature of our method, these formulae are not only sufficient, but also necessary for the correctness [2]. We demonstrate our method on several examples and show how correctness of those may be proven fully automatically. In fact, even if a small part of the specification is missing – in the literature this is often a case – the correctness cannot be

* This research was partly supported by BMBWK (Austrian Ministry of Education, Science, and Culture) and Upper Austrian Government Project "Technologietransferaktivitäten".

proven. Furthermore, a relevant counterexample may be constructed automatically [3].

A specialized strategy for proving termination of recursive functional programs is developed [5]. The detailed termination proofs may in many cases be skipped, because the termination conditions are reusable and thus collected in specialized libraries. Enlargement of the libraries is possible by proving termination of each candidate, but also by taking new elements directly from existing libraries.

2 Theorema System

Our work is performed in the frame of the *Theorema* system [1], which is a mathematical computer assistant aiming at supporting all the phases of mathematical activity: construction and exploration of mathematical theories, definition of algorithms for problem solving, as well as experimentation and rigorous verification of them. Moreover, the logical verification conditions can be passed to the automatic provers of the system. *Theorema* includes a collection of general as well as specific provers for various interesting domains (e. g., integers, sets, reals, tuples, etc.).

3 New Contributions

Based on earlier development on various program types (e.g. recursive programs with multiple recursive calls [2], and programs containing nested recursion [4]), we present here the most recent achievements we have made, and in particular verification of functions defined by mutual recursion.

References

1. B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 2005.
2. T. Jebelean, L. Kovacs, and N. Popov. Experimental Program Verification in the Theorema System. *Int. Journal on Software Tools for Technology Transfer (STTT)*, 2008. To appear.
3. N. Popov and T. Jebelean. A Prototype Environment for Verification of Recursive Programs. In Z. Istenes, editor, *Proceedings of FORMED'08*, pages 121–130, March 2008. to appear as ENTCS volume, Elsevier.
4. N. Popov and T. Jebelean. Verification of Functional Programs Containing Nested Recursion. In B. Buchberger, T. Ida and T. Kutsia, editors, *Proceedings of SCSS'08*, pages 163–175, Hagenberg, Austria, July 2008.
5. N. Popov and T. Jebelean. Proving Termination of Recursive Programs by Matching Against Simplified Program Versions and Construction of Specialized Libraries in Theorema. In D. Hofbauer and A. Serebrenik, editors, *Proceedings of 9-th International Workshop on Termination (WST'07)*, pages 48–52, Paris, France, June 2007.