Constructive Methods in Automatic Analysis of Correctness Proofs ⋆

Alessandro Avellone[1] Marco Benini[1] Dirk Nowotka[2]

[1] Dipartimento di Scienze dell'Informazione
Universitá degli Studi di Milano, via Comelico 39/41 — 20135, Milano, ITALY
[2] Turku Centre for Computer Science, Åbo Akademi University
Lemminkäisenkatu 14 A — 20520 Turku, FINLAND
{avellone,beninim}@dsi.unimi.it, dnowotka@abo.fi

The goal of this contribution is to show that an instrument from Constructive Proof Theory, the *Collection Method* [MO79,MO81], can be used in practice to extract information from a correctness proof. In particular, we use such a method to automatically label the source code of a program with assertions that state the minimal pre- and postconditions (wrt the correctness proof) which ensure the validity of the specification.

Our application is in the analysis of correctness proofs, so we have a fixed program and we want to *maximize* the amount of significant information we can get about it from its correctness argument. In this respect, as shown in [MO79], the Collection Method is more powerful than normalization-based techniques, since it extracts much more information. Of course, we want to filter the extracted information, since every correctness proof contains parts with are relevant for the program, while other parts are accessory, providing just logical or arithmetical results.

The Collection Method takes a set I of proofs as input, and generates a set, $\mathrm{Coll}^*(I)$, as output, which contains the information content of I. In our application I will be the singleton set composed by the correctness proof, and $\mathrm{Coll}^*(I)$ will be the set the labelling algorithm operates on.

Our verification environment is for object code programs, thus we gave a specialized version of the labelling algorithm. It is also possible to define a general version.

Hence, our process to analyze an object program through its correctness proof, is as follows

1. To generate via the Collection Method the information content $\mathrm{Coll}^*(I)$ where I is the singleton set containing the correctness proof;
2. To filter $\mathrm{Coll}^*(I)$ via the labelling algorithm, and to construct a set of assertions, one for every instruction;
3. To remap these sets onto the original program as intermediate specifications.

We should remark that these steps can be performed lazily, and interleaved, so that we are able to generate specifications step by step.

⋆ **Keywords:** Verification, Analysis

References

[MMO88] Pierangelo Miglioli, Ugo Moscato, and Mario Ornaghi. Constructive theories with abstract data types for program synthesis. In D. Skordev, editor, *Mathematical Logic and its Applications*, pages 293–302. Plenum Press, New York, 1988.

[MO79] Pierangelo Miglioli and Mario Ornaghi. A purely logical computing model: the open proofs as programs. Technical Report MIG-7, Istituto di Cibernetica – University of Milano, 1979.

[MO81] Pierangelo Miglioli and Mario Ornaghi. A logically justified model of computation. *Fundamenta Informaticae*, IV((1,2)):151–172, 277–341, 1981.