Formal Result Checking for Unverified Theorem Provers

Nicole Rauch
Technische Universität Kaiserslautern

The importance of formal software verification is growing steadily. Software companies that develop safety-critical applications start to introduce formal verification into their software production processes. Usually, special-purpose theorem provers are used for this task. The vast majority of these provers has not been formally verified for various reasons. This is a huge drawback because these tools are not formally trustworthy. The work presented in this paper shows how this drawback can be circumvented. It was inspired by the methodology called "Proof-carrying code" [Nec97], which ensures certain properties for translated programs. The approach presented here ensures certain properties (to be precise: the existence of a formally correct proof) for programs proven by unverified proof tools. The work is applied to the JIVE tool to demonstrate the usability of the approach.

JIVE is a verification system developed at the Universities of Hagen and Kaiserslautern. It is an interactive special-purpose theorem prover for the verification of object-oriented programs on the basis of a Hoare-style programming logic. Jive operates on a subset of sequential Java which is called Java-KE. The language Java-KE is presented in [PHR04b], together with its Hoare logic. One of JIVE's main strengths is its dedicated user interface. At all times, the user has full visual control over the whole proof process. Proof operations can be selected in menus. A description of JIVE's architecture is given in [MPH00].

A disadvantage of Jive that is often criticized is that it is implemented in Java. The system is equipped with an encapsulated proof container [Sch04], and only the Hoare rules are allowed to modify the proof state stored in this container. But still there are many opportunities to invalidate the resulting proof if there are bugs in the implementation of the Hoare rules in Jive. This disadvantage is wedded out by using the following approach: A proof that has been performed in Jive is written to an Isabelle/HOL theory. This theory is then checked with Isabelle/HOL. If Isabelle/HOL succeeds, then it can be assumed that the goal of the original proof has been proven. This way, we gain much more confidence in the Jive proofs.

For this task, we need to formalize the programming language and Hoare logic in Isabelle/HOL. The idea behind the formalization is of Java-KE's abstract syntax is described in detail in [PHR04a]. The formalization of Java-KE's Hoare Logic rules in Isabelle/HOL has been performed according to [Nip02].

What do we gain by implementing and using the proof-carrying proofs technique? If we generate an Isabelle/HOL theory that contains a Hoare logic proof, and if this proof is accepted by Isabelle/HOL, then we know that the program contained in this theory matches its specification contained in this theory – provided that the formalization of the Hoare Logic rules was correct (which can be verified by a correctness proof), and provided that Isabelle/HOL is implemented correctly (which we do not verify, but since it is an open source system with a very small kernel, and since quite a number of persons looked at it in so far, we believe that it is quite reliable). It is worth noting that we do *not* know that the proof that was created in Jive is valid, because Jive might be implemented incorrectly, and the check theory might be generated incorrectly, in such a way that the incorrect strategy produces a correct proof in Isabelle/HOL from the incorrect Jive proof. But we do know that there exists a valid proof, which is finally what we want to know.

References

[MPH00] Jörg Meyer and Arnd Poetzsch-Heffter. An architecture for interactive program provers. In S. Graf and M. Schwartzbach, editors, *TACAS00, Tools and Algorithms for the Construction and Analysis of Systems*, volume 276 of *Lecture Notes in Computer Science*, pages 63–77. Springer Verlag, 2000.

[Nec97] George C. Necula. Proof-carrying code. In *ACM Symposium on Principles of Programming Languages and Systems*, pages 106–119, Paris, France, January 1997.

[Nip02] Tobias Nipkow. Hoare logics in Isabelle/HOL. In H. Schwichtenberg and R. Steinbrüggen, editors, *Proof and System-Reliability*, pages 341–367. Kluwer, 2002.

[PHR04a] Arnd Poetzsch-Heffter and Nicole Rauch. Application and formal specification of sorted term-position algebras. In José L. Fiadeiro, editor, *17th International Workshop on Recent Trends in Algebraic Development Techniques, WADT 2004, Barcelona, Spain*, volume 3423 of *Lecture Notes in Computer Science*, pages 201–217. Springer Verlag, March 2004.

[PHR04b] Arnd Poetzsch-Heffter and Nicole Rauch. A Hoare Logic for a Java Subset and its Proof of Soundness and Completeness. Internal report, Technische Universität Kaiserslautern, Germany, 2004. To appear.

[Sch04] Jan Schäfer. Encapsulation and specification of object-oriented runtime components. Master's thesis, University of Kaiserslautern, September 2004.