VOCAL – A Verified OCAml Library

Arthur Charguéraud[3,4], Jean-Christophe Filliâtre[1,2],
Mário Pereira[1,2], and François Pottier[3]

[1] Lab. de Recherche en Informatique, Univ. Paris-Sud, CNRS, Orsay, F-91405
[2] Inria Saclay–Île-de-France, Orsay, F-91893
[3] Inria Paris
[4] ICube – CNRS, Université de Strasbourg, France

Libraries are the basic building blocks of any realistic programming project. It is thus of utmost interest for a programmer to build her software on top of bug-free libraries. Even massively used and tested libraries can contain bugs: in 2006 a bug was found in Java's standard library, after 9 years of undetected presence [3]. One approach to verifying the behavior of such libraries is to employ deductive software verification [11], that is, to reduce the correctness of a program down to a mathematical statement, and to prove that this statement is true. Projects such as CompCert [14] and seL4 [13] show how proof assistants can handle large program verification efforts. In addition, the remarkable progress of SMT solvers makes it possible to apply these tools to the verification of realistic program artifacts. For instance, the Verisoft XT project [2] was verified using VCC [16], which builds on the SMT solver Z3 [10]. Although this may seem surprising, program verification has seldom been applied to libraries of significant size. A remarkable exception is the verification of the EiffelBase2 containers library [15], performed with the AutoProof system [18].

This work presents the first steps towards VOCAL, a mechanically verified library of efficient general-purpose data structures and algorithms, written in the OCaml language. OCaml is the implementation language of systems used worldwide where stability, safety, and correctness are of utmost importance. Examples include the Coq proof assistant [17], the Astrée [8] and Frama-C [9] static analyzers, the Cubicle model-checker [7], and the Alt-Ergo theorem prover [4].

One of the key ingredients of the VOCAL project is the design of a specification language for OCaml, independently of any verification tool. This is similar to what JML is for Java [5], or ACSL for C [1]. Another ingredient of the VOCAL project is the development of the verified library itself, using a combination of three tools: CFML [6], Coq, and Why3 [12]. These tools nicely complement each other: CFML implements a separation logic and targets pointer-based data structures; Coq is a tool of choice for purely applicative programs; and Why3 provides a high degree of automation using off-the-shelf SMT solvers. A consistent collaboration between these tools, keeping the benefits of each one, is one of the challenges we intend to address in this project.

In this talk we will present the current state of the specification language, which is still under development, and of the library itself. Using examples of already verified OCaml modules, we will illustrate several verification challenges (modular proofs, absence of arithmetic overflows, proof of complexity bounds, etc.) and how we successfully cope with them.

This research was partly supported by the Portuguese Foundation for Sciences and Technology (grant FCT-SFRH/BD/99432/2014) and by the French National Research Organization (project VOCAL ANR-15-CE25-008).

References

1. Patrick Baudin, Jean-Christophe Filliâtre, Claude Marché, Benjamin Monate, Yannick Moy, and Virgile Prevosto. *ACSL: ANSI/ISO C Specification Language, version 1.4*, 2009. `http://frama-c.cea.fr/acsl.html`.
2. Bernhard Beckert and Michał Moskal. Deductive verification of system software in the Verisoft XT project. *Künstliche Intelligenz*, 24(1):57–61, 2010.
3. Joshua Bloch. Nearly all binary searches and mergesorts are broken, 2006. `http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html`.
4. François Bobot, Sylvain Conchon, Évelyne Contejean, Mohamed Iguernelala, Stéphane Lescuyer, and Alain Mebsout. The Alt-Ergo automated theorem prover, 2008. `http://alt-ergo.lri.fr/`.
5. Lilian Burdy, Yoonsik Cheon, David R. Cok, Michael D. Ernst, Joeseph R. Kiniry, Gary T. Leavens, K. Rustan M. Leino, and Erik Poll. An overview of JML tools and applications. *International Journal on Software Tools for Technology Transfer (STTT)*, 7(3):212–232, June 2005.
6. Arthur Charguéraud. Characteristic formulae for the verification of imperative programs. In Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy, editors, *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming (ICFP)*, pages 418–430, Tokyo, Japan, September 2011. ACM.
7. Sylvain Conchon, Amit Goel, Sava Krstić, Alain Mebsout, and Fatiha Zaïdi. Cubicle: A parallel SMT-based model checker for parameterized systems. In Madhusudan Parthasarathy and Sanjit A. Seshia, editors, *CAV 2012: Proceedings of the 24th International Conference on Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, Berkeley, California, USA, July 2012. Springer.
8. Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. The ASTRÉE analyzer. In *ESOP*, number 3444 in Lecture Notes in Computer Science, pages 21–30, 2005.
9. Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C: A software analysis perspective. In *Proceedings of the 10th International Conference on Software Engineering and Formal Methods*, number 7504 in Lecture Notes in Computer Science, pages 233–247. Springer, 2012.
10. Leonardo de Moura and Nikolaj Bjørner. Z3, an efficient SMT solver. In *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
11. Jean-Christophe Filliâtre. Deductive software verification. *International Journal on Software Tools for Technology Transfer (STTT)*, 13(5):397–403, August 2011.
12. Jean-Christophe Filliâtre and Andrei Paskevich. Why3 — where programs meet provers. In Matthias Felleisen and Philippa Gardner, editors, *Proceedings of the 22nd European Symposium on Programming*, volume 7792 of *Lecture Notes in Computer Science*, pages 125–128. Springer, March 2013.
13. Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: Formal verification of an OS kernel. *Communications of the ACM*, 53(6):107–115, June 2010.
14. Xavier Leroy. A formally verified compiler back-end. *Journal of Automated Reasoning*, 43(4):363–446, 2009.
15. Nadia Polikarpova, Julian Tschannen, and Carlo A. Furia. A fully verified container library. In Nikolaj Bjørner and Frank D. de Boer, editors, *FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings*, volume 9109 of *Lecture Notes in Computer Science*, pages 414–434. Springer, 2015.
16. Wolfram Schulte, Songtao Xia, Jan Smans, and Frank Piessens. A glimpse of a verifying C compiler. `http://www.cs.ru.nl/~tews/cv07/cv07-smans.pdf`.
17. The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.6*, 2016. `http://coq.inria.fr`.
18. Julian Tschannen, Carlo A. Furia, Martin Nordio, and Nadia Polikarpova. Autoproof: Auto-active functional verification of object-oriented programs. In *21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer, 2015.