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


