Mechanical Verification of Automatic Synthesis of Fault-Tolerant Programs

Sandeep S. Kulkarni, Borzoo Bonakdarpour, and Ali Ebnenasir

Department of Computer Science and Engineering,
Michigan State University,
48824 East Lansing, Michigan, USA
{sandeep, borzoo, ebnenasi}@cse.msu.edu
http://www.cse.msu.edu/~{sandeep,borzoo,ebnenasi}

Abstract. Fault-tolerance is a crucial property in many systems. Thus, mechanical verification of algorithms associated with synthesis of fault-tolerant programs is desirable to ensure their correctness. In this paper, we present the mechanized verification of algorithms that automate the addition of fault-tolerance to a given fault-intolerant program using the PVS theorem prover. By this verification, not only we prove the correctness of the synthesis algorithms, but also we guarantee that any program synthesized by these algorithms is correct by construction. Towards this end, we formally define a uniform framework for formal specification and verification of fault-tolerance that consists of abstract definitions for programs, specifications, faults, and levels of fault-tolerance, so that they are independent of platform and architecture. The essence of synthesis algorithms involves fixpoint calculations. Hence, we also develop a reusable library for fixpoint calculations on finite sets in PVS.

Keywords: Fault-tolerance, PVS, Program synthesis, Program transformation, Mechanical verification, Theorem proving, Addition of fault-tolerance.

1 Introduction

Fault-tolerance is a necessity in most computer systems and, hence, one needs strong assurance of fault-tolerance properties of a given system. Mechanical verification of such systems is one way to get a strong form of assurance. The related work in the literature has focused on verification of concrete fault-tolerant programs. For example, Owre et al [1] present a survey on formal verification of a fault-tolerant digital-flight control system. Mantel and Gärtnert verify the correctness of a fault-tolerant broadcast protocol [2]. Qadeer and Shankar [3] mechanically verify the self-stability property of Dijkstra’s mutual exclusion token.

This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF grant EIA-0130724, and a grant from Michigan State University.

While the verifications performed in [1,2,3,5] enable us to gain confidence in the programs being verified, it is difficult to extend these verifications to other programs. A more general approach, therefore, is to verify algorithms that generate fault-tolerant programs.

With this motivation, in this paper, we focus on the problem of verifying algorithms that synthesize fault-tolerant programs. With such verification, we are guaranteed that all the programs generated by the synthesis algorithms indeed satisfy their fault-tolerance requirements. Towards this end, we verify the transformation algorithms presented by Kulkarni and Arora [7,8] using the PVS theorem prover. The algorithms in [7,8], focus on the problem of transforming a given fault-intolerant program to a fault-tolerant program. To verify these algorithms, first, we model a framework for fault-tolerance in PVS. This framework consists of definitions for programs, specifications, faults, and levels of fault-tolerance. Then, we verify that the programs synthesized by the algorithms are indeed fault-tolerant. By this verification, we ensure that any program synthesized by these algorithms is also correct by construction and, hence, there is no need to verify the individual synthesized programs.

We note that the algorithms in [7,8], are the basis for their extensions to deal with simultaneous occurrence of multiple faults from different types [9] and for synthesizing distributed programs [10,11]. Thus, the specification and verification of transformation algorithms in [7,8] is reusable in developing specification and verification of algorithms in [10,11,9]. Since fixpoint calculation is at the heart of the synthesis algorithms, we also develop a library for fixpoint calculations on finite sets in PVS. This library is reusable for other purposes that involve fixpoint calculations as well.

Contributions of the paper. The contributions of this paper are as follows:
(1) We verify the correctness of the synthesis algorithms in [7,8]. Thus, not only we ensure their correctness but also we guarantee that any program synthesized by the algorithms is also correct by construction. (2) We provide a foundation for formal specification and verification of later research work that are extensions of [7,8]. (3) We develop a reusable library in PVS for fixpoint calculations on finite sets.

Organization of the paper. The organization of the paper is as follows: We provide the formal definitions of programs, specifications, faults, and fault-tolerance in Section 2. Using these definitions, we formally state the problem of mechanical verification of synthesis of fault-tolerant programs in Section 3. In Section 4, first, we develop a theory for fixpoint calculations on finite sets. Then, based on the definitions in Section 2 and our fixpoint calculation library, we formally specify the synthesis algorithms proposed in [7,8] in PVS. In Section 5.

2 The URL http://www.cse.msu.edu/~borzoo/pvs contains the PVS specifications and formal proofs.