

Why Don't They Do What We Want Them to Do?

Shmuel Schwarz, Mordechai Ben-Ari

Department of Science Teaching, Weizmann Institute of Science, Rehovot 76100 Israel
moti.ben-ari@weizmann.ac.il
shmuel.s@weizmann.ac.il

Abstract. This paper describes an investigation into the factors affecting a student's decision whether to construct a state transition diagram in order to verify the correctness of a concurrent program, or whether to verbally verify the program. We conjectured that the advantages of the visual formal tool would cause it to be adopted as a routine part of the students' practice, but in fact the verbal description was the dominant method of their practice. This paper describes the reasoning that the students used in choosing a proof method. Psychological factors such as personal commitment and evaluation of effort turned out to be more important than the appropriateness of the tool for achieving the goal.

1. Introduction

A problem that beginning students of programming have is the lack of an effective model of a computer, that is, a mental representation of the algorithmic processes that occur during the execution of a program (Ben-Ari, 1998). This is particularly problematic in concurrent programs, because the interleaving semantics of concurrent programs significantly changes the concept of program flow (Ben-Ari & Ben-David Kolikant, 1999). In sequential programming there is one output for every input, while for concurrent programs a single input or initial state can give rise to many scenarios.

Verification of correctness is also much more difficult because sequential programs are functional in the sense that a correctness specification describes the output as a function of the input, making it possible to test a program for representative input values; for a concurrent program global assertions must be satisfied in *all* scenarios. The concept of interleaving is very difficult for students, making it hard for them to mentally simulate the execution of a concurrent program. Even if they could, programs cannot be tested in the usual sense because of the astronomical number of scenarios.

Today the leading technique for verifying concurrent programs is *model checking* (Cleaveland & Smolka, 1996). Model checking is based on the fact that most concurrent algorithms have finite state, so that it is theoretically possible to create the entire state diagram for a program. Since all possible executions are represented by paths in the diagram, correctness specifications can be verified by examining the diagram.

