Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 130 (2005) 323–343
www.elsevier.com/locate/entcs

Automatic Verification of Safety Rules for a Subway Control Software

Nelson Guimarães Ferreira and Paulo Sérgio Muniz Silva[1,2]

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, n.158 05508-900 São Paulo SP Brazil

Abstract

This paper proposes the introduction of an automatic verification phase for a subway control software development process in which bounded model checking (BMC) and induction proof would be used to anticipate error discovery and increase the quality of the final product. We report the tests we developed for some safety rules of two actual sections of a subway track and the results we achieved. We conclude that the technique seems feasible for the problem domain, but the issue requires extensive research to allow an exact understanding of which requirements the use of the BMC meets, and actual benefits this approach might bring to the project.

Keywords: Bounded model checking, safety requirements model checking, subway control software model checking.

1 Introduction

Computer-controlled systems had a boost in the last few years, as much as the problems that may arise due to software errors. The problem is even more serious when we consider that the systems to be controlled are becoming more and more complex, whereas the delivery times are the same as before, if not shorter. While software correctness may be a very important issue for non-safety-related systems, it is critical for systems controlling devices that can put human lives as well as expensive equipment and installation at risk.

[1] Email: nelson.ferreira@poli.usp.br
[2] Email: paulo.muniz@poli.usp.br

doi:10.1016/j.entcs.2005.03.017

This paper describes an effort to assess the feasibility of model checking the software controlling a section of an actual subway system. A subway track may be large enough to make it necessary to divide it into many sections and to assign to each section a controller running its own specific (safety-related) software. A section normally encompasses a small number of stations, which means that the software controlling it may be relatively complex. In order to mitigate the risks of delivering an implementation with errors, the company developing the controller defines a verification and validation (V&V) process with inspection and test activities in well-defined phases.

We believe that there are some solid reasons for implementing an automatic verification process for that kind of development. Firstly, it is a well-known fact that fixing a software error becomes more and more expensive as the project progresses. The use of model checking immediately after the software is ready for release to the inspection phase may anticipate error discovery, thereby generating savings. Secondly, it may also contribute to an improvement in the quality of the final product, because an automatic verification may uncover subtle errors that would not be uncovered otherwise. Thirdly, the verification activities presented here may represent reduced man-hours when compared to other activities in the development process. Finally, most of the work done for the first software release may be reused during the development of the next releases.

There is some research on the automatic verification of railway control software. Since, in general, the controlled railway and its associated software can be easily and automatically reduced to a Finite State Machine (FSM), it is natural that model checking and other techniques exploiting that kind of formalism have been used to tackle the problem. Examples in the literature are [8], [7], [6], [9], [11], and [12].

Our approach is close to [8], [7] and [6]. The first common point between all the works is the problem domain: railway control software written in languages that are close to each other. [8] describes the verification of some safety rules of a relatively simple Dutch station. The tool that was used is the Stålmarck theorem prover. The second work uses a Binary Decision Diagram (BDD) based model checker to verify both the same station that [8] verified as well as a more complex one. Both works are related to ours because, although they used different techniques than us, they intended to verify safety rules for railway control software using an exhaustive state-space exploration. Another shared point is that the techniques they used allow the verification of rules which may exhibit very general patterns.

[6] uses bounded model checking (BMC) to verify some safety invariants and proposes using it at the end of the implementation phase of the develop-