Formal Verification of Generalised State Machines

George Eleftherakis and Petros Kefalas

Computer Science Department
City College
Affiliated Institution of the University of Sheffield
13 Tsimiski Str., 54624 Thessaloniki, Greece
email: eleftherakis@city.academic.gr, kefalas@city.academic.gr

***Abstract*— The demand for more complex software is constantly increasing while at the same time the need for reliability leads modern software engineering to use more formally based development techniques. One of the most successfully employed formalisms to address the reliability issue were Finite State Machines (FSM) but they are too simple to capture the modelling needs of modern software that normally require manipulation of non-trivial data structures. X-machines is a formal method that provides such a data structure in form of a memory. On the other hand, FSM models are suitable for verification through model checking, i.e. to prove that certain properties are satisfied by a system model. However, with existing logics, it is obscure how one can describe properties that refer to the memory structure of an X-machine. This paper describes how a new logic, namely XmCTL, which extends temporal logic with memory quantifiers, facilitates model checking of X-machine models. XmCTL is defined and its use is demonstrated through the verification of a steam-boiler system which acts as a case study for our contribution.**

Keywords: **Formal Methods, Verification, Temporal Logic**

I. Introduction

One of the challenges that emerge in software engineering is to develop software models and implementations that are correct. The criteria for correctness are: (i) the initial model should match the user requirements, (ii) the model should satisfy any necessary properties in order to meet its design objectives, and (iii) the implementation should pass all tests constructed using a complete functional test generation method [1]. All the above criteria are closely related to modelling, verification and testing and they improve confidence on using the final product.

Although modern software engineering aims to manage the inherent complexity of software systems, little emphasis is given to methodologies proposed leading towards correct systems. There has been a strong debate on whether formal methods can achieve this goal. Academics and practitioners adopted extreme positions either for or against formal methods [2]. It is, however, apparent that the truth lies somewhere in between and that there is a need for use of formal methods in software engineering [3], [4].

In this paper, we use a formal method, namely X-machines (XM), proposing a formal approach to verify the properties of designs represented as XM models. XM offers an intuitive and practical way of modelling [1] and at the same time a formal testing strategy to test the implementation against the XM model [5]. We have analytically described elsewhere [6], [7], [8], [9], that XM and its extension, namely communicating X-machines [10], are particularly suitable for modelling. Here, we focus on the verification and we demonstrate the practicality and the intuitiveness of the method through a case study.

The paper is organised as follows. In Section 2, we describe the motivation behind and the contribution of the current work. Section 3 provides an introduction to X-Machines modelling and model checking and the new language devised for this purpose, together with a brief overview of model checking through Temporal Logic. Section 4 is the case study of a fairly complex model of a steam-boiler. Finally, the last section concludes with a discussion and suggestions for further work.

II. Motivation and Contribution

Among formal methods, the Finite State Machines (FSM) manage to model the essential feature of a system, which is the change of its internal state. The FSM, however, lacks the ability to model any non-trivial data structures. Using FSM to implicitly deal with different values is rather complicated, since the number of states increases in combinatorial fashion to the possible values.

A formal method in order to be useful to modelling of modern software systems should be able:

- to model both the data and the control of the system,
- to model separately the components of the system and the ways the components interact with each other to provide an efficient way to scale up to large scale systems,
- to be intuitive, practical and effective towards implementation, and
- to facilitate development of correct systems.

All the above are prominent characteristics of X-Machines. XM formed the basis for a possible specification language [1]. With the development of a formal testing strategy [5] and a methodology of building communicating systems out of XM components [11], most of the above requirements are met. In addition, a formal framework for the development of more reliable systems was proposed [12]. To complete the picture, the formal verification of XM models is the contribution of the current research.