Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 48 (2006) 981–992
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Extended state identification and verification using a model checker

Christopher Robinson-Mallett [a,*], Peter Liggesmeyer [a], Tilo Mücke [b], Ursula Goltz [b]

[a] University of Kaiserslautern, Fraunhofer IESE, Germany
[b] University of Brunswick, Institute for Programming and Reactive Systems, Germany

Received 7 March 2006; accepted 21 March 2006
Available online 23 May 2006

Abstract

This article presents a method for the application of model checking, i.e., verifying a finite state system against a given temporal specification, on the problem of generating test inputs. The generated test inputs allow state characterization, i.e., the identification of internal states of the software under test by observation of the input/output behavior only. A test model is derived semi-automatically from a given state-based specification and the testing goal is specified in terms of temporal logic. On the basis of these inputs, a model checking tool performs the testing input generation automatically. In consequence, the complexity of our approach is strongly depending on the input model, the testing goal, and the model checking algorithm, which is implemented in the used tool. The presented approach can be adapted with small changes to other model checking tools. It is a capable test generation method, whenever a finite state model of the software under test exists. Furthermore, it provides a descriptive view on state-based testing, which may be beneficial in other contexts, e.g., education and program comprehension.

Keywords: State characterization; Automata-based testing; Conformance testing; Model checking; Test generation; Protocol testing; Checking sequence generation; Software testing

1. Introduction

Testing is the execution of a piece of software under test (SUT) for the purpose of fault detection. It is a mandatory quality assurance technique in each successful software development project and a frequent task in each programmer's everyday life. The selection of a fault sensitive, representative, and economic sample among the vast number of possible test inputs is after decades of intensive research still a problem of high relevance. Currently, test design automation is an important trend in industry and research, which allows the reduction of human errors and expenditures of time and money at a defined test quality. In this article, we present a method for the automated generation of test inputs on the basis of state-based specifications and coverage criteria. Furthermore, we aim at the verification and identification of internal states of the implementation, which is beneficial whenever black box testing is performed and observations are restricted to input/output behavior.

In the recent years, Model Checking, i.e., verifying a finite state system against a given temporal specification [6], has been established as a powerful static verification method. The two most prominent approaches to model checking have been introduced independently by Clarke and Emerson [5], based on Computational Tree Logic (CTL), and Quielle and Sifakis [17], based on Linear Temporal Logic (LTL). A detailed overview of both approaches can be found in [6, pp. 35–49]. In this article, we will concentrate on the application of the approach by Clarke and Emerson a small subset of CTL in order to generate test inputs from a finite state model of the SUT.

The construction of a *checking sequence*, also referred to as machine distinguishing experiment in [11] or machine identification experiment in [9] and [15], which can be used

* Corresponding author. Tel.: +49 631 6800 2236; fax: +49 631 6800 1499.
E-mail addresses: mallett@iese.fraunhofer.de (C. Robinson-Mallett), peter.liggesmeyer@iese.fraunhofer.de (P. Liggesmeyer), tmuecke@ips.cs.tu-bs.de (T. Mücke), goltz@ips.cs.tu-bs.de (U. Goltz).

0950-5849/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2006.03.006

to check conformance of an SUT against a state-based specification has been under research since the earliest days of computing [10]. The conformance check is based on the derivation of the internal states of the SUT from its input/output behavior. Therefore, a checking sequence is constructed from a cover set, e.g., inputs that execute each transition, and a set of input sequences that can be used to characterize each current state of the SUT. The construction of cover sets from state-based specifications has been under research for many years and seems to be well understood. Recent publication, e.g. [13], of model checking to the problem of generating test inputs on the basis of coverage criteria bridge the gap between static verification and testing of state-based specifications.

Characterization sets: A characterization set is a set of input sequences, referred to as state characterization sequences, on a minimal finite state machine (FSM) that produces different output for each different initial state. In this context, the initial state is the state in which the software resides before executing the input sequence. Referring to the example in Section 6 of this article, a simple form of characterization set consisting only of the input sequence *baa* decides each state of the finite state machine in Fig. 2. In this paper, we focus on the construction of three forms of characterization sets: *Distinguishing Sequence* (DS) [10,11,15,9], i.e., the characterization set is a singleton; *Unique Input Output Sequences* (UIOs), i.e., the characterization set contains for each state an input sequence that decides it from any other state of the FSM; and *W-sets*, i.e., a number of input sequences as a whole decide each state of the FSM.

In the domain of state characterization problems we briefly discuss the two most prominent: (1) *State Identification*: The problem is to identify the unknown initial state of a FSM under test. (2) *State Verification*: The problem is to verify that a state machine under test is in a particular state at a specific stage of the test. Both problems are addressed with the application of characterization sets. From all the different kinds of characterization sets some efficiently solve the first, e.g., DS [10], and others aim at the second, e.g., UIO [21]. A characterization set can be either preset, i.e., the input is fixed ahead execution, or adaptive, i.e., the next input is decided during execution [14]. However, any of the presented methods can be applied to both problems, more or less efficiently. Here, we follow a general approach of generating characterization sets without regarding whether state identification or state verification is addressed.

In this article, we present a method for the application of model checking in order to generate characterization sets that provide full fault coverage [4] on FSMs. Furthermore, we demonstrate the application of this method to extended finite state machines (EFSM). We present a general model and specifications in terms of computational tree logic that form the input for a model checking tool in order to generate characterization sets automatically.

This article is organized as follows: In Section 2, we discuss related work and position our approach into the research area of automata-based testing. In Section 3, the most important basics of state machines, timed automata, and the UPPAAL model checker are presented. In Section 4, our approach on generating characterization sets with UPPAAL is presented and its complexity is discussed in Section 5. In Section 6, an example of an EFSM is given. In Section 7 we conclude this article and present future research topics.

2. Related work

The development of automata-theoretic testing methods was originally motivated by checking problems of sequential circuits [15]. The adoption of these methods to software has been an important research topic over decades. A detailed overview of automata-based testing methods can be found in a number of articles, e.g. [3,20]. Any of these automata-based testing methods are demanding a minimal, completely specified FSM.

One of the earliest approaches on automata-based testing was based on DS [10,11,15], which produce relatively short checking sequences. A method based on UIOs was first presented by Sabnani and Dahbura in [21]. In [4], Chow presented the *W*-method that produces relatively long checking sequences, but which is applicable to each minimal FSM. Lee and Yannakakis proposed an efficient method for the construction of adaptive distinguishing sequences (ADS) [14].

Furthermore, in [14] Lee and Yannakakis presented a detailed study on the complexity of the construction of DS and UIO, with the negative result that the construction of both is PSPACE-hard, thus most probably only exponential algorithms exist, and that not each minimal FSM may possess one. Nevertheless, any of these methods provide defined fault coverage for a specification in the form of a minimal FSM [4,3,11]. In [4], Chow proved full fault coverage, i.e., the detection of missing states, missing and illegal transitions, for the *W*-method. The detection of additional states, also proposed by Chow in [4], is exponential in the number of additional states.

Furthermore, their number must be known in advance. Therefore, the detection of additional states is only of theoretical meaning and we classify W, UIO, and DS as methods with full fault coverage.

The generation of DS using a model checker was already presented in a preceding publication [19]. In the same paper, we also demonstrated the application of DS generation to EFSMs. In [18], we generalized this approach to the generation of preset DS, UIOs, *W*-sets on the basis of FSMs. In this paper, we extend our approach to the generation of ADS and EFSMs.

3. Preliminaries

In this section, we briefly describe the basics of FSM, EFSM, and UPPAAL Timed Automata. For a detailed introduction into the basics of state machines we