Available online at www.sciencedirect.com

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 113 (2005) 123–143

www.elsevier.com/locate/entcs

Simulation of Simultaneous Events in Regular Expressions for Run-Time Verification

Usa Sammapun Arvind Easwaran Insup Lee Oleg Sokolsky[1],[2]

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA

Abstract

When specifying system requirements, we want a language that can express the requirements in the simplest and most intuitive form. Although the MaC system provides an expressive language, called MEDL, it is generally awkward to express certain features like temporal ordering of complex events, timing constraints, and frequencies of events which are inherent in safety properties. MEDL-RE extends the MEDL language to include regular expressions to easily specify timing dependencies and timing constraints. Due to simultaneous events generated by the MaC system, monitoring regular expressions by simulating DFAs would result in a potential problem. The DFA simulations would involve concurrent multi-path simulations and result in exponential running time. To handle simultaneous events inexpensively, we generate a dependency graph to identify possible simultaneous events. Further, we augment the original DFAs with alternative transitions, which will substitute for multi-path simulations.

Keywords: Runtime verification, DFA simulation, DFA, MEDL.

1 Introduction

The monitoring, checking and steering (MaC) framework [9,10,11] has been designed to ensure that the execution of a real-time system is consistent with its requirements at run-time. It provides a language, called MEDL, to specify safety properties based on LTL [13]. The safety properties include both

[1] This research was supported in part by NSF CCR-9988409, NSF CCR-0086147, NSF CCR-0209024, and ARO DAAD19-01-1-0473.

[2] Email: {usa,arvinde,lee,sokolsky}@saul.cis.upenn.edu

doi:10.1016/j.entcs.2004.01.030

computational and timing requirements. The safety properties are defined in terms of events, conditions, auxiliary variables, and auxiliary functions. Events are instantaneous incidents such as variable updates or the start/end of a method call. Conditions are propositions about the program that may be true or false for a duration of time. Those events and conditions can also be composed using connectives described in Section 2. Auxiliary variables are temporary storage, which allows us, for example, to count the number of occurrences of an event. Auxiliary functions return values and time stamps of events. The MEDL language provides an elegant and intuitive way to specify computational requirements. It, however, does not provide an intuitive way to specify timing requirements, such as temporal ordering of events with complex timing dependencies, timing constraints or counting of specific events in a time interval.

The extension of MEDL called MEDL-RE [14] adds the ability to specify ordering of events in the form of regular expressions (RE) over a customized set of events, which offers users with clearer and less error-prone specifications. In this paper, we propose an efficient simulation of the corresponding DFAs at runtime. By observing a sequence of events occuring in a target system, a DFA generated by MEDL-RE matches the sequence of events with a specific regular expression. However, because the composite events can be triggered simultaneously and cannot be temporally ordered in any way, the DFA must recognize these events for any ordering of the events. This means if a regular expression has some inherent ordering of simultaneous events, the DFA must accept all different permutations of such order. We refer to those permutations as *linearizations*. To build such a DFA, we augment the original DFA with alternative transitions to provide paths from one linearization to another. We then prove that the original DFA and the augmented DFA are equivalent. Only those DFAs whose underlying regular expressions have candidate simultaneous events in their relevant sets are augmented. The candidate simultaneous events can be statically detected by building and traversing a dependency graph described in Section 4.

The paper is organized as follows. Section 2 briefly explains an overview of the MaC framework. Section 3 introduces an extension MEDL-RE. Section 4 discusses the construction of the dependency graph. Section 5 presents and proves our augmented DFA algorithm. Section 6 presents related work. Lastly, section 7 concludes the paper.