Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 243 (2009) 49–67
www.elsevier.com/locate/entcs

An Improvement of Software Architecture Verification

Zuohua Ding [1]

Center of Math Computing and Software Engineering
Zhejiang Sci-Tech University
Hangzhou, 310018, P.R.China

Jing Liu [2,3]

Shanghai Key Lab of Trustworthy Computing
East China Normal University
Shanghai, 200062, P.R.China

Abstract

Static analysis may cause state space explosion problem. In this paper we explore differential equation model that makes the task of verifying software architecture properties much more efficient. We demonstrate how ordinary differential equations can be used to verify application-specific properties of an architecture description without hitting this problem. An architecture behavior can be modeled by a group of ordinary differential equations containing some control parameters, where the control parameters are used to represent deterministic/nondeterministic choices. Each equation describes the state change. By checking the conditions associated with the control parameters, we can check whether an equation model is feasible. After solving a feasible equation model, based on the solution behavior and the state variable representation, we can analyze properties of the architecture. A WRIGHT architecture description of the Gas Station problem has been used as the example to illustrate our method. All of the equations have been computed with Matlab tool.

Keywords: Static analysis; architecture; ordinary differential equation; Wright.

1 Introduction

Static analysis is an approach to program behavior verification without execution. The approach is particularly useful in identifying program design errors prior to implementation. It has been demonstrated that detecting errors early in the lifecycle greatly reduces the cost of fixing those errors. A number of static analysis techniques

[1] Email:zuohuading@hotmail.com
[2] Email:jliu@sei.ecnu.edu.cn
[3] Corresponding author

doi:10.1016/j.entcs.2009.07.005

have been proposed. They span such approaches as reachability-based analysis techniques, symbolic model checking, flow equations, and dataflow analysis.

These techniques and approaches have been used in several analysis tools such as *Flow equation* is used in INCA [5], *data flow analysis* is used in FLAVERS [11], *Reachability Analysis* is used in SPIN [14], *Symbolic model checking* is used in SMV [4] and SMC [21].

In general all existing approaches appear to be very sensitive to the size of the program being analyzed in terms of the use of concurrency constructs and the number of asynchronous processes. Particularly, reachability analysis may cause state space explosion problem since it has to exhaustively explore all the reachable state space to detect concurrency errors. Although many techniques have been proposed to combat this explosion, such as state space reductions, compositional techniques, abstraction, the state explosion problem still is the main technical obstacle to transition from research to practice.

Current concurrent systems are described as discrete event system models which are suited to model system concurrency. However, the discrete will lead to state explosion problem since model checkers build a finite state transition system and exhaustively explore the reachable state space searching for violations of the properties under investigation [3]. Hence, to thoroughly solve the state explosion problem, one solution is that the discrete event system model should be continunized to continuous system model, such that the systems can be described with analytic expressions. Therefore, instead of counting states, we can analyze the solutions of the analytic expressions.

Petri net seems a good candidate that bridges discrete event systems and continuous systems. On one hand, Petri nets have been used extensively as tools for the modeling, analysis and synthesis of discrete event systems. Petri nets offer advantages over finite automata, particularly when the issues of model complexity and concurrency of processes are of concern. On the other hand, a continuous system can be approximated by a Petri net [20] and a Petri net model is used as discrete event representation of the continuous variable system by Lunze et al. [17].

However, Petri nets also suffer from the state explosion problem while doing reachability analysis[18] even through there are some net reduction methods. One way to tackle that problem is to use some kind of relaxation by removing the integrality constraints. This relaxation leads to a continuous-time formalism: Continuous Petri Net (CPN) by David and Alla[6][7]. A continuous Petri net, in fact, is an approximation of the timed (discrete) Petri net. The semantics of a continuous Petri net is defined by a set of ordinary differential equations (ODEs), where one equation describes the continuous changes over time on the marking value of a given place. Different firing styles in the CPN can lead to different semantics of CPN. In this paper, we consider a modified VCPNs in which the instantaneous firing speeds depend on the markings such that the markings are continuous without points of discontinuity.

Based on CPN, a concurrent system can be modeled by a group of ordinary differential equations containing some control parameters, where the control param-