An Open-Bisimilarity Based Automated Verification Tool for \( \pi \)-Calculus Family of Process Calculi

Shahram Rahimi, Southern Illinois University, USA
Rishath A. S. Rias, Southern Illinois University, USA
Elham S. Khorasani, Southern Illinois University, USA

ABSTRACT

The complexity of designing concurrent and highly-evolving interactive systems has grown to a point where system verification has become a hurdle. Fortunately, formal verification methods have arrived at the right time. They detect errors, inconsistencies and incompleteness at early development stages of a system formally modeled using a formal specification language. \( \pi \)-calculus (Milner, 1999) is one such formal language which provides strong mathematical base that can be used for verifying system specifications. But manually verifying the specifications of concurrent systems is a very tedious and error-prone work, especially if the specifications are large. Consequently, an automated verification tool would be essential for efficient system design and development. In addition, formal verification tools are vital ingredient to fully harness the potential of component-based software composition. The authors developed such an automated verification tool which is highly portable and seamlessly integrates with the visualization, reduction and performance evaluation tools introduced (Ahmad & Rahimi, 2008; Rahimi, 2006; Rahimi et al., 2001, 2008) to provide a comprehensive tool for designing and analyzing multi process/agent systems. Open-Bisimulation (Sangiorgi, 1996) concept is utilized as the theoretical base for the design and implementation of the tool which incorporates an expert system implemented in Java Expert System Shell (JESS) (Friedman-Hill, 2003).

Keywords: Automated Verification Tool, Calculus, Java Expert System Shell, Mathematics, Open-Bisimulation

1. INTRODUCTION

The advent of concurrent and multi-threaded programming has led to the development of complex and highly-evolving interactive systems which concurrently run two or more independent,
interactive processes with dynamically evolving structures (Davis, 2000). The introduction of mobile computing introduced a whole new layer of abstraction in these systems – **mobility**. Systems comprising of mobile agents or mobile systems in general, contain dynamic links which are constantly changing. New links are created on-the-fly, existing links transferred or deleted (Fernandez & Mounier 1992). This dynamics makes the mobile systems much more complex than their counterparts and comparatively difficult to model. Such concurrent mobile systems may be implemented via processes and/or threads which execute in a relatively independent way. However, since they are acting together towards some common goal, they need to communicate and coordinate. The two most common communication techniques in processes and threads are through **message passing** and **shared variables** respectively. We try to avoid the shared variables paradigm because it forces the usage of complicated and time-delaying **mutual exclusion** techniques as multiple processes making assignments to the same shared variable can lead to unpredictable behavior if the assignments are made concurrently. Now, if an unnoticed bug gets introduced in the design of such a system it will propagate into every development stage of the system. Depending on the application design, the entire system might break after its initial deployment. Traditional testing techniques are of limited help here, since test coverage is bound to be only a minute fraction of the possible behaviors of the system. Moreover, bugs in concurrent systems are usually very hard to reproduce and hence hard to understand. Thus, formal verification methods that detect errors, inconsistencies and incompleteness at early development stages will help improve the software development cycle especially at the Testing and Debugging stage.

For a system to be verified, its system specification and its intended behavior have to be described in a precise and formal way. This will allow verifying mathematically, as to whether it is behaving according to the designer’s intensions and to reason about the behavior of the system. One way to formally describe a system is to use process algebra. Process algebras are standard paradigm for describing and analyzing communicating concurrent systems (Milner, 1989). It refers to the algebraic approach of studying concurrent processes. We use algebraic languages for specifying processes and statements about them, and a calculus for verifying the statements. The three main classical algebraic approaches to concurrency are Calculus of Communicating Systems (CCS) (Milner, 1989), Communicating Sequential Processes (CSP) (Hoare, 1985) and Algebra of Communicating Processes (ACP) (Bergstra & Klop, 1985). CCS is one of the first of process calculi and was developed by Robin Milner. It models the pure communication and synchronization aspects of concurrent systems and includes primitives for describing parallel composition, choice between actions and scope restriction.

CSP was introduced in early 1980’s by Hoare (1985). It is a formal language for describing patterns of interaction. As its name suggests, CSP allows a user to describe systems as a number of (parallel) components (processes) that operate independently and communicate with each other solely over well-defined channels. It introduces a process algebra which can be used to describe a process’ communications with its environment. In addition, it has been successfully used as the basis for the programming language Occam, and in recent years has been extended into a wider domain, for example to address tasks in hardware co-design. In ACP (Bergstra & Klop, 1985) on the other hand, **process algebra** was extended with communication to yield its theory. It emphasizes the algebraic aspect: there is an equational theory with a range of semantical models. Also, ACP has a more general communication scheme. However, with the aforementioned process algebras it is difficult to model systems where the structure of communication changes dynamically as these mobile systems contain dynamic links that are constantly changing. These dynamics make the mobile systems much more complex than their counterparts and comparatively difficult to model.
Related Content

An Introduction to Multi-Objective Optimization
[www.igi-global.com/chapter/introduction-multi-objective-optimization/26949?camid=4v1a](www.igi-global.com/chapter/introduction-multi-objective-optimization/26949?camid=4v1a)

ISW: Interest-Based Small World Network of P2P
[www.igi-global.com/article/isw/88925?camid=4v1a](www.igi-global.com/article/isw/88925?camid=4v1a)
Application of Structural Properties of Seismic Data to Prediction of Hydrocarbon Distribution

Zhou Wei, Guo Haimin and Yaoting Lin (2018). International Journal of Software Science and Computational Intelligence (pp. 41-52).
www.igi-global.com/article/application-of-structural-properties-of-seismic-data-to-prediction-of-hydrocarbon-distribution/207744?camid=4v1a

A Comparative Study of Unsupervised Video Shot Boundary Detection Techniques Using Probabilistic Fuzzy Entropy Measures

www.igi-global.com/chapter/comparative-study-unsupervised-video-shot/72495?camid=4v1a