Applied Soft Computing 24 (2014) 169–180

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# A heuristic solution for model checking graph transformation systems

Rosa Yousefian, Vahid Rafe*, Mohsen Rahmani

*Department of Computer Engineering, Faculty of Engineering, Arak University, Arak 38156-8-8349, Iran*

ARTICLE INFO

*Article history:*
Received 24 April 2013
Received in revised form 5 April 2014
Accepted 27 June 2014
Available online 12 July 2014

*Keywords:*
State space explosion
Model checking
Graph transformation systems
Genetic algorithm

ABSTRACT

One of the commonly used techniques to verify software and hardware systems which have been specified through graph transformation system (GTS), especially safety critical ones, is model checking. However, the model checking of large and complex systems suffers from the state space explosion problem. Since genetic algorithm (GA) is a heuristic technique which can be used to search the state space intelligently instead of using exhaustive methods, in this paper, we propose a heuristic approach based on GA to find error states, such as deadlocks, in systems specified through GTS with extra large state space. To do so, in each step of space exploration our algorithm determines which state and path should be explored. The proposed approach is implemented in GROOVE, a tool for model checking graph transformation systems. The experimental results show that our approach outperforms significantly in comparison with existing techniques in discovering error states of models with large state space.



## Introduction

Correctness is one of the most desirable properties for any software system. A great part of each developing approach is devoted to assess the correctness of the system; however, faults and errors are common. The sooner errors are found the better results can be achieved [1]. Hence, model-based verification techniques should be used before implementation to decrease the cost and the risk and also to increase the quality of the product. To do so, it is necessary to use an accurate approach to specify models, and thus, using formal methods for this purpose can be one of the best solutions.

Graphs and diagrams provide a very useful, direct, and intuitive means to describe software systems [2]. Graph transformation systems (GTS) [2,3] provide the formal foundations to convert graphs and diagrams into formal specifications. Even though GTS has not been very popular for years, it is able to capture the systems behavior naturally and succinctly. However, modeling per-se is not enough. Proper verification solutions are mandatory to convince people of the actual benefits provided by modeling. Different approaches already exist to verify GTS through model checking [4–6]. All these approaches only deal with a priori bounded transition systems because of the nature of model checking techniques in general. However, in many cases, the transition system tends to be too large. Thus, the state space explosion problem occurs. In this case the usual verification approaches like [4–11] cannot be useful.

Although there exists different approaches, like symbolic verification [12,13], partial order reduction [14,15], symmetry checking methods [16–19] and scenario-driven model checking [1] for the systems specified through GTS, almost all of these methods use exhaustive search on the state space and thus suffer from the state explosion problem. For the cases in which the usual approaches cannot be employed due to the size of the state space, using heuristic approaches like GA can be a proper solution.

Genetic algorithms are usually used for optimization problems but in this work, we propose a method using GA to find errors such as deadlocks in systems specified through GTS. Since it is not possible to search all the states, our approach chooses a subset of them to be explored using GA. We implement the proposed approach through GROOVE [20,21], a toolset to generate state space for GTS specification. In this paper, we concentrate more on searching deadlocks which are a sort of safety properties.

The rest of the paper is organized as follows. In Related works Section, we review some of the already proposed solutions for dealing with state space explosion. In Background Section, we briefly introduce the required background. Proposed method Section presents our proposed approach to exploration of the state space using GA. Our algorithm implementation is described in Implementation Section. In Evaluation Section, we employ the proposed method for several known problems and consequently we discuss the results of our experiments. Finally, we conclude the paper and highlight the future works in Conclusion and future works Section.

* Corresponding author. Tel.: +98 86 32625524.
*E-mail addresses:* Rosa8a81@yahoo.com (R. Yousefian), v-rafe@araku.ac.ir, rafe@iust.ac.ir (V. Rafe), m-rahmani@araku.ac.ir (M. Rahmani).

http://dx.doi.org/10.1016/j.asoc.2014.06.055
1568-4946/

## Related works

Generally, proposed approaches for overcome the state space explosion that can be found in the literature fall into one of the following four broad categories:

### *Classic approaches*

To cope with the state space explosion problem in model checking systems, some classic solutions like the ones based on reducing the size of the model's state space [17–19,22–27] and those based on memory saving [28–32] have been presented. There also exist model abstraction techniques that reduce the state space by reducing the size of the model. The general idea in methods which are based on memory saving is to use various algorithms for reducing the memory space required for saving the states.

### *Simple heuristic approaches*

A simple heuristic search algorithm is the best-first search, which is applied for protocol validation to achieve significant gains over depth-first search [33]. Heuristics for choosing a search order that favors visiting first successor states that are most likely to lead to an error are discussed in [34] in the context of symbolic model checking and in [35] in the context of explicit model checking. A best-first search with binary decision diagram (BDD) based model checking within the Murϕ tool has been used in [36]. Bloem et al. [37] used heuristics to reduce the bottlenecks of image computation in symbolic model checking. In another work [38], the authors proposed an OBDD-based version of the A* algorithm. Friedman et al. [39] applied a Coverage First Search related to structural heuristics to generate test suites. Ganai and Aziz [40] also used coverage-based techniques to guide a state-space search for control-dependent hardware.

### *Meta heuristic approaches*

In recent years, some artificial intelligence algorithms have been used to improve the memory utilization. To bring up a few examples, frameworks based on reinforcement learning [41], ant colony algorithm [42–44] and GA in studying and exploring the large state space models can be mentioned.

A framework based on reinforcement learning has been presented for checking the linear temporal properties [41]. In this framework, the goal is to improve the memory utilization through increasing the probability of occurrence of the paths that lead to violation of the properties. In reinforcement learning algorithm the agent learns about the concepts by interacting with its environment through rewards and punishments.

By using the ant colony algorithm, inspired by the efforts of ants in searching for the shortest path to the food resources, model checking systems can explore the state space of the problem [42–44]. By using this algorithm, they have been able to find the optimal solution for the problem with the least amount of resources.

Most of the researches that have been conducted for model checking systems in the field of GA, for example [45,46], are based on the textual modeling. Godefroid and Khurshid [46] introduce a new framework for reducing the state space in concurrent reactive systems. This framework directs us into the error states by utilizing heuristic practical programs that use the GA. This framework is implemented using Verisoft [47] which is a tool for exploring the systems' state space. This framework, by using the GA and utilizing some heuristic methods, guides a search engine into finding errors such as deadlock and assertion violations in state space of a concurrent reactive system [46]. In Java Pathfinder a model checking system has been implemented that can identify the deadlocks [45]. For more details about other heuristic approaches interested readers can refer to [48].

### *Heuristic approaches on GTS*

In [49] the authors have formalized a framework for the application of heuristic search in order to analyze structural properties of systems modeled by graph transition systems. They believe that heuristic search is intended to reduce the analysis effort and also to deliver shorter solutions, (i.e. shorter paths in graph transition systems). To do so, they use A* algorithm. This framework is implemented in HSF-SPIN a heuristic model checker compatible with the successful model checker SPIN [50].

Most of the heuristic approaches mentioned till now are exhaustive search approaches. In fact, the employed heuristic approach helps the traversing algorithm to find the error states faster, but the state space explosion still can occur. In this paper, we propose a solution in which only a subset of the state space is considered to be searched.

Another important issue is that GTS is a graphical formal language which is used in different software development phases (e.g. meta-modeling [51], architectural style representation [52], refinement [53], refactoring [54], model transformation [55], performance analysis [56], etc.). So, it is very useful to propose a novel approach to overcome the state space explosion problem in GTS.

## Background

### *Model checking*

Utilization of formal methods as an efficient way for quality assurance and correctness checking of a designed system is highly necessary. Model checking is an automatic method for studying the properties given to a system and their verification. In model checking, properties can be classified into different groups like safety and liveness.

The safety property, informally, claims that "no bad event should occur" in a system or a "good incident" should always happen in a system. To find a state that leads into the violation of safety property, it is necessary to find a finite path into a violating state (e.g. deadlock state) [57].

If all states are valid, it becomes clear that this property is valid in the system and if it is rejected, the model checking approach can demonstrate the violated path and state.

Liveness properties assert that "something good" will happen eventually. In order to find a Liveness property violation, it is needed to find an infinite path in which a desired state never occurs (i.e. the expected good never happened) [58].

Since model checking needs to generate all the possible state space of the model to verify its validity and the given properties, the model must be relatively small. In other words, model checking of large models faces the problem of memory resource limitation which causes the state space explosion problem.

### *Graph transformation system*

Graph transformation is a graphical formal language for system modeling. The mathematical foundation of graph transformation systems returns to more than 40 years ago in response to shortcomings in the expressiveness of classical approaches to rewriting (e.g. Chomsky grammars) to deal with nonlinear grammars [59]. Graphs are suitable tools for explanation and modeling the structure of complex systems. One of the most fundamental features of the GTS is its formal and accurate mathematical basis [2].