Information and Software Technology 97 (2018) 110–134

Contents lists available at ScienceDirect

Information and Software Technology

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

Searching for violation of safety and liveness properties using knowledge discovery in complex systems specified through graph transformations

Einollah Pira[a], Vahid Rafe[a,*], Amin Nikanjam[b]

[a] Department of Computer Engineering, Faculty of Engineering, Arak University, Arak 38156-8-8349, Iran
[b] Faculty of Computer Engineering, K.N. Toosi University of Technology, Tehran 1631714191, Iran

ARTICLE INFO

Keywords:
Data mining
Bayesian network
Model checking
State space explosion
Graph transformation system

ABSTRACT

Context: Model checking is an automatic and precise technique in verification and refutation of software and hardware systems. Despite its advantages, the state space explosion problem may occur in large and complex systems. Recent studies demonstrate that using meta-heuristic and evolutionary algorithms are a proper solution to handle the state space explosion problem. In systems which are specified formally through graph transformations, the state space is constructed by applying all enable rules on all generated states. In such systems, there is a dependency between rules in each sequence of applied rules in the state space.

Objective: This fact motivates us to use knowledge discovery techniques to intelligently explore only a portion of the state space instead of exhaustive exploration. We propose two different techniques to acquire such knowledge form the model state space. In this paper, we propose a data mining-based approach in which the required knowledge is obtained from exploring a slight portion of the model state space. Another approach is proposed in which a Bayesian network is used to capture this knowledge. After acquiring the required knowledge, it is employed to explore only a portion of the model state space intelligently, to refute a property.

Results: The proposed approaches can be used to analyze the reachability, safety and liveness properties. To evaluate the proposed approaches, they are implemented in GROOVE, an open source toolset for designing and model checking of systems specified through graph transformations.

Conclusion: Experimental results on different set of benchmarks show that the proposed approaches are faster and more accurate in comparison with the existing meta-heuristic and evolutionary techniques in model checking of complex software systems specified through graph transformations.

1. Introduction

Model checking is now increasingly used as one of the well-known techniques in formal verification of different systems, such as circuits [1], the verification of various network or safety protocols [2], or the formal verification of program languages, such as C [3] or Java [4]. Also, it has been employed as a fitness function in genetic programming (GP) to automatic repair of software systems [5]. In model checking, all possible states (called state space) of a system are generated and then a given property is checked. If the system has a very large or infinite state space, the state space cannot be explored completely and the state space explosion problem will occur [6]. Recent studies demonstrate that using meta-heuristic and evolutionary algorithms such as Genetic Algorithm (GA) [7], Ant Colony Optimization (ACO) [8,9] and Particle Swarm Optimization (PSO) [10] can be a proper solution to handle the state space explosion problem. These approaches explore only a portion of the state space to verify (or refute) the given property intelligently. Although useful, these approaches might be unsuccessful in very large and complex systems. In systems which are specified formally through graph transformations, the state space is constructed by applying all enable rules on all generated states. In such systems, there is a dependency between rules in each ordered sequence of applied rules in the state space. Specially, some of these ordered sequences of rules are frequently appeared in different parts of the state space. This fact motivates us to use knowledge discovery techniques to intelligently explore only a portion of the state space instead of exhaustive exploration. In [11], a data mining-based approach was proposed to handle this problem. In the approach, to check complex and large models intelligently, a special knowledge (i.e. a frequent pattern) is obtained from checking some smaller models. This approach can work successfully when the large and corresponding smaller models are consistent with the same architectural styles. Additionally, generating an adequate smaller model in a specific style, especially in real and complex systems, can be a critical bottleneck.

* Corresponding author.
E-mail addresses: piraep@gmail.com (E. Pira), v-rafe@araku.ac.ir (V. Rafe), nikanjam@kntu.ac.ir (A. Nikanjam).

https://doi.org/10.1016/j.infsof.2018.01.004
Received 14 May 2017; Received in revised form 3 December 2017; Accepted 8 January 2018
Available online 31 January 2018

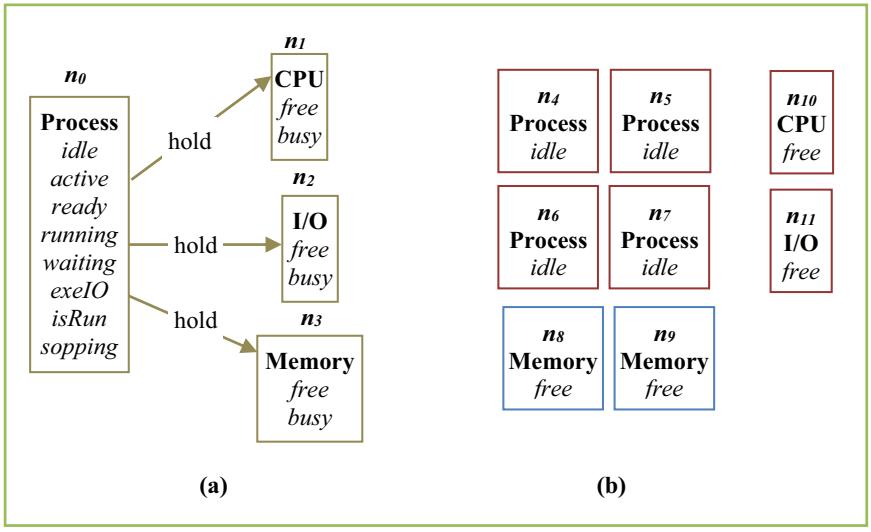

Fig. 1. The type and host graphs of the model of the process life cycle problem designed in GROOVE.

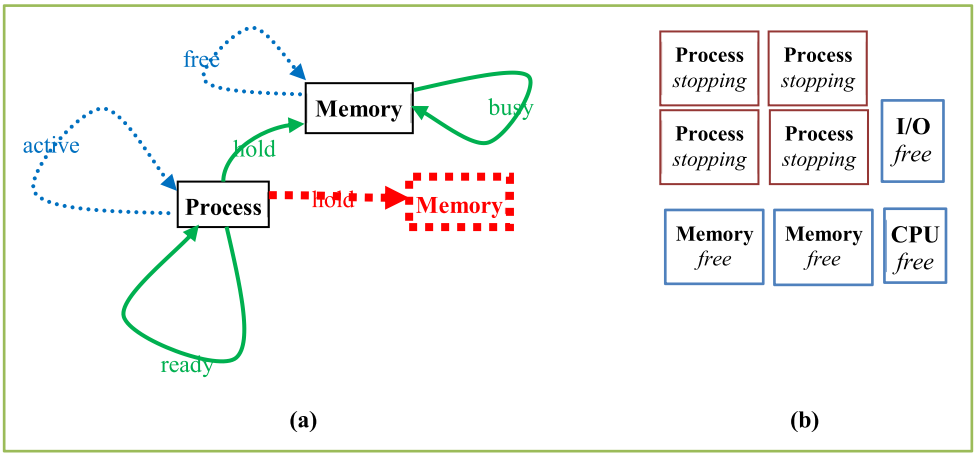

Fig. 2. An example of a transformation rule and a state property in the model designed by GROOVE.

In this paper, to alleviate this bottleneck, we propose an approach in which the required knowledge is acquired from exploring a slight portion of the model's state space instead of exhaustive exploration of the smaller model. In other words, the specific number of states is firstly explored by Breadth-First Search (BFS) algorithm and then the number of states is selected from the last level of the explored state space. These selected states (called promising states) have more chance than other states to reach a goal state (i.e. a state in which a given state property is satisfied or refuted). In our approach, all paths starting from an initial state of the state space and leading to a promising state are considered as a training dataset. To discover the required knowledge from this training dataset, similar to [11], we employ data mining techniques to detect a frequent pattern. This pattern is a sequence of rules which appears in the training dataset with frequency no less than a specific threshold [12]. There is a high probability that the repetition of applying this sequence of rules on promising states can cause a goal state to be reached sooner. Furthermore, it is highly probable that there are some dependencies between the applied rules in the training dataset. Since using Bayesian networks (BNs) is one way to capture these dependencies, we propose another approach in which a BN stores the existing dependencies. BN is a probabilistic model which is used to capture dependencies/independencies between variables of the problem [13]. After discovering a set of frequent patterns or constructing a BN, it can be used to explore the remainder of the model's state space intelligently until a goal state is reached.

The main contributions of the proposed approaches are as follows: (1) Using BNs and data mining to analyze the reachability properties and this capability is used to refute the safety properties. Also, they can detect a deadlock, a state with no subsequent state. Furthermore, they can refute the liveness properties. (2) They can analyze complex systems with very large state spaces in comparison with other existing meta-heuristic and evolutionary approaches and finally, (3) They can generate short counterexamples/witnesses.

In model checking, a system should be described by a formal modeling language. One of the proper formal languages is Graph Transformation System (GTS) [14]. In a system modeled by GTS, structural and behavioral aspects can be represented by graphs and graph transformations respectively. There are different tools such as