A Survey on Coping with the State Space Explosion Problem in Model Checking

Vahid Rafe, Mohsen Rahmani, Kianoosh Rashidi

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

Corresponding Author email: v-rafe@araku.ac.ir

ABSTRACT: Today, computer systems play a significant role in our daily lives. They are used in many different fields such as: smart cards, mobile phones, smart devices, telecommunication systems, e-commerce, banking system and etc. However, the more computer systems penetrate our lives, the more complicated they get. On the other hand, some of them are used in critical applications in which a bug or an error can be fatal, catastrophic and causes a huge loss of money, like: nuclear plants, chemical plants, aviation systems, biological devices and etc. Therefore, this kind of computer systems needs more accurate and precise type of verification than the traditional test and simulation techniques. Hence, formal software verification approaches use instead. Model checking is an effective and efficient type of formal verification which has been used for verification of safety-critical systems in last two decades. In model checking technique, the system which has to be verified, is modeled as a finite state machine in which nodes are system states and edges are transitions between those states. The major drawback of model checking approach is state space explosion which means the number of states needed to model the system exceeded the amount of available memory. There are several methods to fight the state space explosion. This survey provides a perspective on these techniques and reviews some of the previous articles.

INTRODUCTION

Model checking is a type of formal verification which is based on the state space exploration (explicitly or symbolically) of the system. In this technique, the system which is to be verified, is modeled as a finite state machine which describes all possible states of the system and known as transition system. Also, the property which is interested to check in the system model is described in a formal language such as temporal logic. Then, the transition system and the property which is described in a formal language are applied as input to a model checker. After that, the model checker starts exploring all reachable states of the transition system to check whether the property is true. (Bosnacki, 2001; Pelanek, 2009)

Based on how exploring the state space, model checking algorithms fall into two category: The techniques in which the transition system is built and states explore explicitly. These algorithms called Enumerative Algorithms. Other techniques, avoid building the transition system, instead they represent the system model implicitly by some Boolean functions. These approaches explore a set of states in a single step instead of enumerating one state at a time. These algorithms are known as Symbolic Algorithms.

Those techniques that traverse the state space explicitly fall into two categories based on the time when they build the state space: techniques that build the state space completely then start exploring all reachable states and on-the-fly techniques that build the state space dynamically on-demand during the model checking operations. (Bouajjani et al., 1997; Barnat et al., 2012)

Major methods which are used to fight the state space explosion problem are: Symbolic Model checking, On-the-fly Model checking, Partial order reduction Model checking and Symmetry Reduction.

In this article we describe each aforementioned technique. Section 2 discusses symbolic model checking and Binary decision diagrams. In Section 3, partial order reduction is described. In section 4, symmetry reduction is discussed. Section 5 considers on-the-fly model checking technique. Finally, we conclude the survey in section 6.
Symbolic Model Checking

Symbolic model checking approaches avoid building the finite state machine as a transition system to model systems; instead, they represent the system state space implicitly using Boolean functions and formulas in propositional logic. These functions and formulas are represented in a form of a structure called Binary Decision Diagram (BDD). Therefore, model checking operations become the exploration of the BDDs. Since, the memory size which is needed to store the Boolean functions is exponentially smaller than the memory size needed to store the explicit transition system, systems with large number of states that cannot be verified by enumerative algorithms due to the state explosion, now, can be verified by the symbolic model checking. The main elements of the symbolic model checking approaches are: Binary Decision Diagram and Symbolic Algorithm which are discussed in two next sections. (Clarke et al., 2001; Bryant, 1992)

Binary Decision Diagram

BDD is a rooted, directed acyclic graph that provides an effective way of representing the Boolean functions. In BDD each leaf node is labeled with 0 or 1 and other nodes that do not have 0- or 1- label are Boolean variables. Each variable node has two outgoing edge one is labeled with 1 and the other is labeled with 0. Fig. (1) shows the Ordered Decision Tree and Binary Decision Diagram each representing the Boolean function \( x \land (y \lor z) \). Decision Tree is also a data structure for representing a Boolean function. (Clarke et al., 2001; Bryant, 1992; Akers, 1978; Brace et al., 1990)

In practice, the term BDD is usually refers to OBDD (Ordered Binary Decision Diagram). ODBDDs are a form of reduced BDDs which give compact representation for a Boolean formula. A BDD is called Ordered if Boolean variables appear in the same order on all paths from root to leave. BDD and also OBDD which are representing a Boolean function can be derived from its corresponding decision tree. OBDD of a Boolean function is obtained by applying the following rules to its Decision Tree:
1. Combine any isomorphic sub-trees into a single tree
2. Eliminate any nodes whose left and right children are isomorphic.

In Fig. (2), decision tree of Boolean function \( a \lor b \land c \land d \) is depicted. The two mentioned rules are applied to the decision tree in a bottom up fashion and the corresponding OBDD is obtained in a linear time.

![Decision Tree](image-url)

Figure 2. Decision tree representing: \( a \lor b \land c \land d \) (McMillan, 1992)
In Fig. (2-2) the nodes in decision tree which has *-label represent the isomorphic sub-trees that should be merged. Since the node which has + -label does not have any effect on the result of the function is removed from the tree. Fig (3) shows the OBDD of the Boolean formula \((a \wedge b \vee c \wedge d)\) which is derived from the decision tree shown in Fig. (2).

![Figure 3. OBDD representing: a \(\wedge b \vee c \wedge d\) (McMillan, 1992)](image)

**Symbolic Algorithm**

As discussed before, the model checking techniques which are based on enumerative algorithms explore the state space of the system explicitly single state at a time. The main problem of these techniques is state space explosion. Even a small system can have a large state space and causes the problems in model checking process. Thus, symbolic algorithms are offered to combat the state space explosion. (Coudert et al., 1990) Symbolic algorithms avoid building the finite state machine and explicit exploring of state space. Instead they represent the system model using Boolean functions and propositional logic in the form of BDDs. On the other hand, Symbolic algorithms instead of exploring single state at a time operate on a set of states in a single step. (Clarke et al., 2001; Bryan, 1992) First, Symbolic model checking was an effective method in hardware design and digital circuits verification. OBDDs and BDDs have good capabilities in modeling the digital circuits. (Peled, 1998)

In (Burch et al., 1992), using symbolic state space instead of explicit state space, the author develops a model checking approach based on BDDs for verification of mu-calculus. The applicability of the method is evaluated by verifying a simple synchronous pipeline. Farn Wang in (Wang, 2003) suggests an approach which exploits the BDD-like structure to develop an approach in order to verify timed automata.

State space of some systems is inherently infinite. Therefore, special approaches need to be used in order to model those systems. Bounded Model Checking (BMC) is an approach is used in cases in which the state space is too big or infinite. In (Biere et al. 1999) a symbolic model checking approach is presented in which BDD is not used and also, a bounded model checking method is introduce for model checking of the properties which are described in Linear Temporal Logic (LTL). The tools check if the property violation can occur in \(k\) or fewer. These techniques only can be a proof of presence of errors and not the absence of them. Several tools have been developed based on Symbolic model checking approach such as: CadenceSMV(McMillan, 1999), NuSMV(Cimatti et al., 2000), VIS( Brayton et al., 1996).

**Partial Order Reduction**

Although, Symbolic model Checking gained some success in the field of model checking of relatively large systems and specially in hardware design, but the model checking of asynchronous software systems remains as a challenge. As the time goes by, the number of asynchronous components which work concurrently within a system increased. On the other hand, The more asynchronous component a system has, The larger its state space gets. Therefore, the partial order reduction was proposed to fight the state space explosion in concurrent systems with asynchronous components.

In model checking a concurrent system using formalisms such as temporal logics like LTL or CTL, the state space of the system is modeled as a complete set of interleaving sequences of its concurrent components. In fact, Partial Order, is an execution of a system that may contains a set of interleaving sequences of its concurrent components which all of them has a same effect on the system. So, the only differences between them are the order of concurrent events. On the other hand, modeling all set of the interleaving sequences of all executions is
redundant and will result in a huge state space and state space explosion problem. (Peled, 1998; Godefroid, 1994; Godefroid et al., 1995)

Thus, in partial order reduction approaches, instead of a general state space in which all interleaving sequences of an execution has been modeled, a reduced state space is used in which the extra interleaving sequences for executions are eliminated. Different partial order reduction approaches works based on three different type of subsets of states: Persistent, Ample and Stubborn (Peled, 1993; Valmari, 1991; Gueta, 2007), which are known as representative subsets. These subsets will determine which sequences of states should be eliminated and which sequences should be remained for model checking process.

(Godefroid et al., 1995) proposes an approach in which uses state space caching and exploits the partial order reduction in order to reduce the redundancy in exploring the state space. In (Gueta et al., 2007), the authors introduce a cartesian semantics which is combined by the partial order reduction technique and present a dynamic partial order reduction algorithm. In order to improve Depth-first Search algorithm for verifying the liveness and safety properties of a concurrent system, in (Holzmann and Peled, 1995) some modifications has been done to avoid the redundant interleaving sequences for an executions. Doron Peled in (Peled, 1994) studies an extention of SPIN model checker that implements a model checking technique in which the partial order reduction and on-the-fly model checking are combined together. Another version of partial order reduction model checking is proposed in (Kurshan et al., 1998) which is known as static partial order reduction model checking. In this approach, all reduction operations are done in the system specification compile time. On the other hand, in (Flanagan and Godefroid, 2005) dynamic partial order reduction approach is introduced which is based on the the tracking of the dynamic interactions between the system threads and processes. Also in (Vander Meulen and Pecheur, 2011), an approach is proposed in which partial order reduction is combined with symbolic model checking based on BDDs. This approach shows an effectiveness in model checking of properties which are described in LTL logic.

**Symmetry Reduction**

Symmetry reduction is one of the techniques for fighting state space explosion. This approach emphesize on the identical or isomorphic processes in a system. The isomorphic processes of a system cause the state space of the system to contain states which are identical. The main idea of this approach is to identify the isomorphic states in the state space in order to reduce the size of the state space, hence, model checker will operate on a smaller version of the state space instead of the general state space. This approach is based on the group theory. As a reminder, State space of a system is shown as triple \( K = (S, R, L) \), where \( S \) is a set of all states, \( R \) is set of transitions and \( L \) is a labeling function which assigns each state an atomic proposition. In Group Theory, a group of permutation is a group \( G \) whose elements are permutations of an arbitrary given set, here we assume this set is \( S \), and whose operation is a bijective function which maps \( S \) to itself and produce the permutations of all elements in \( S \). Now, assume \( G \) is a permutation group of \( S \) (acting over \( S \)), a permutation \( \sigma \in G \), which is a permutation of states in \( S \), is said to be a Symmetry of \( K \) if and only if it preserves the transition relations in set \( R \). On the other hand, \( G \) is a symmetry group for state space \( K \), if and only if every permutation in \( G \) is a symmetry of \( K \). Therefore, by applying the symmetry group \( G \) to set of states of \( K \), several equivalence classes produced which is called orbits. In other words, an orbit is a set of states sequences which is obtained from a state \( s \in S \) by applying permutations in \( G \). Eventually, a smaller version of the original state space \( K \) is obtaind which contains one member from each orbit. (Norris Ip and Dill, 1996; Emerson and Sisla, 1996; Clarke, 1998)

In (Clarke et al., 1993) a symmetry approach is proposed for both BDD base symbolic model checking and explicit model checking techniques. Dragan Bosnacki in (Bosnacki, 2002), exploits symmetry reduction technique for the verification of distributed systems which modeled as a Buchi automata. The algorithm is a modification of the nested depth first search (NDFS). In (Hendrikx et al., 2003), an approach which is based on symmetry reduction technique is proposed for model checking of real-time systmes in UPPAAL. Farn Wang and Karsten Schmidt in (Wang and Schmidt, 2002), present symbolic algorithms for verification of systmes using BDD-like data structure. Software model is defined with pointer data structures. The technique is implemented in a tool and the performance of the approach is compared with Murphu and SMC against several benchmarks.

**On-the-fly Model Checking**

As mentioned before, model checking techniques can be categorized into two groups based on the time when the transition system is built. Some techniques build the whole finite state machine of the system model and then start to explore the states while some other techniques do not build the whole transition system, instead they build just part of the state space which is needed and gradually expand the it on demand. Therefore, if model checker is verifying some properties which violates the system specification, then it just need to build the state...
space as far as the state in which the violation has been detected. (Barnat et al.,2012;Bhat et al.,1995; Valmari,1998) On the other hand, for a liveness property, the model checker needs to build the state space completely. On-the-fly techniques usually are suitable for model checking approaches which are based on the Depth-first Search traversal of the explicit finite state machine of a system. (Barnat et al.,2012)

Bart Vergauwen and Johan Lewiin (Vergauwen and Lewi, 1993) introduce a local model checking algorithm (ALMC) for CTL. This algorithm exploits the on-the-fly technique for the exploration of the state space of a system. In (Bhat et al.,1995) an efficient on-the-fly algorithm is proposed for model checking of a system to verify whether a property described in Temporal Logic CTL satisfies the system specification. In (Barnat et al.,2009) and (Barnat et al., 2012) one of the open research topic in parallel LT model checking discuss which is to develop an on-the-fly scalable algorithm with linear time complexity. Also, (Biallas et al.,2011) presents an approach which combines the path reduction technique with on-the-fly detection of the breking points during the state space generation.

CONCLUSION

In this paper we discussed the major approaches for fighting the state space explosion in model checking of software systems. There two type of model checking algorithms based on how exploring the state space: Enumerative algorithms and Symbolic algorithms. On the other hand, for explicit model checking approaches, based on the time when state space is built, there are two type of techniques: techniques which build the state space completely and then start to explore the states and On-the-fly techniques which build the state space incrementally on demand. There are, four major approaches for fighting state space explosion. Symbolic model checking which avoids building the finite state machine of the system model and exploit the boolean functions and BDDs to represents the state space implicitly. These techniques reduce the size of each states so they will need a smaller memory space to be stored. Partial order reduction and Symmetry reduction approaches are good for the concurrent systems which have asynchronous components. These techniques reduce the number of states by omitting the redundant interleaving sequences which have a same effect for one execution and identical states, respectively. On-the-fly techniques are also a model checking approaches trying to alleviate the state space explosion by constructing the state space incrementally and on demand fashion, So it will need a smaller memroy size to store the state space of the system model. Although there exist many good model checking approaches, the big challenge remaining is to develop such tools and algorithms with which can verify systems with larger state space.

ACKNOWLEDGEMENTS

We would like to thank Arak University for financial support of this project.

REFERENCES

Bpsnacki D. 2001."Enhancing State Space Reduction Techniques for Model Checking" Doctoral degree, Department of Mathematics and Computing Science,Technische Universiteit Eindhoven,