

## RESEARCH ARTICLE

# Compositional Verification Using Geodesic Distance via Assume-Guarantee Reasoning

XIAOYAN LIU 

School of Software, Henan Polytechnic University, Jiaozuo 454000, China

e-mail: xyanliu@hpu.edu.cn

This work was supported by the Horizontal Project "A Model Checking Method for Concurrent Systems" from Henan Polytechnic University, under Project 12230699.

**ABSTRACT** Assume-guarantee reasoning is indeed an effective compositional verification technique that can help mitigate the state explosion problem in model checking. However, the biggest challenge of applying assume-guarantee reasoning is how to best decompose a system. This paper presents a novel compositional verification frame based on geodesic distance. The proposed algorithm introduces novel techniques to decompose system components into groups. The algorithm's effectiveness and efficiency are evaluated through a comparative analysis with four state-of-the-art methods commonly used in the field. The results of the comparison consistently demonstrate that the proposed algorithm outperforms the state-of-the-art methods. This implies that the new algorithm exhibits superior performance in terms of decomposition quality and verification efficiency.


**INDEX TERMS** Model checking, compositional verification, assume-guarantee reasoning, group technology, combinatorial testing, geodesic distance, partition.

## I. INTRODUCTION

Formal verification is a comprehensive, precise, and automated validation approach that employs formalized languages and mathematical tools to prove that a system's design meets specific specifications or requirements, while eliminating any potential errors or vulnerabilities in the design. Model checking is a prominent and widely used formal verification technique due to its ability to provide automated analysis when both the system model and the desired properties are available. Model checking is a technique in formal verification that automatically explores a finite-state model of a system to detect whether that model violates a given property. It systematically checks whether a given system satisfies a temporal logic formula, which describes a property that the system should possess. The process of model checking typically involves three main steps: constructing a model of the system, specifying the desired properties in a temporal logic formula, and running the model checker to verify whether the model satisfies the properties. However, as the scale of a system increases, particularly with the

addition of concurrent components or processes, the number of possible system states can grow exponentially. *State space explosion* is a widely acknowledged issue, which is one of the challenges faced by model checking [1], [2], [3], [4], [5]. Compositional verification employs the principle of "divide and rule" to address this problem in system verification. The idea behind compositional verification is breaking down the verification process into smaller, more manageable parts and analyzing them separately before combining the results to verify the overall system.

*Assume-guarantee reasoning* (AGR) is a powerful compositional verification technique that involves breaking down a system into components and verifying their behavior based on assumptions and guarantees [6], [7], [8], [9], [10]. AGR provides rules for transforming the global verification of a system into local, more manageable verifications of individual components. These rules are based on the notion of refinement and enable us to verify the behavior of subsystems in isolation, given assumptions about their environment and guarantees about their behavior. Let  $M_1, M_2, \varphi$  be components and a property of a system, respectively. The notation  $M_1 || M_2$  represents the composition of two concurrent components,  $M_1$  and  $M_2$ . Consider the following assume-guarantee

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet .

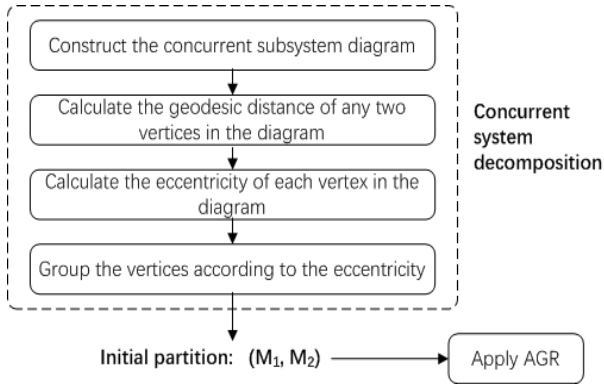


FIGURE 1. Overall flow.

reasoning rule:

$$\frac{M_1 \parallel A_2 \models \varphi \quad \text{and} \quad M_2 \leq A_2}{M_1 \parallel M_2 \models \varphi}$$

$M_2$ 's abstraction is  $A_2$ .  $A_2$  represents a simplified or summarized version of  $M_2$ . It captures the key characteristics or behaviors of  $M_2$  while disregarding certain specific details. The notation  $M_2 \leq A_2$  means  $A_2$  can replicate the behavior of  $M_2$  through simulation. Informally, the rule says that to show  $M_1 \parallel M_2 \models \varphi$ , it suffices to find an assumption  $A_2$  such that  $A_2$  simulates  $M_2$ , and  $M_1$  composed with  $A_2$  satisfies  $\varphi$  as well. However, the biggest challenge in applying AGR is how to split the system into two distinct parts,  $M_1$  and  $M_2$ . Each part represents a subset of the original system's parallel components, allowing for separate analysis or processing. If there is no good decomposition strategy, the efficiency of AGR may be lower than that of overall model checking [11].

The partition approach proposed in this work helps address the decomposition problem in AGR-based compositional verification. The heart of this approach comes from the idea of graph clustering. Figure 1 shows overall flow of AGR compositional verification based on geodesic distance. In this work, the proposed model consists of several individual components running concurrently. These components interact by accessing and modifying shared variables to exchange information and synchronize their actions, making up the overall behavior of the model. Firstly, a concurrent subsystem diagram (Definition 1) is constructed. Then the geodesic distance (Definition 2) between any two vertices in the diagram is calculated, and the eccentricity of each vertex (Definition 3) in the diagram is calculated [12]. Next, group the vertices according to the eccentricity. Based on the grouping results, the initial partition of compositional verification can be obtained. In this step, the key idea is to group the system diagram to ensure that the components in the same group are interrelated, and the components in different groups are more distant and less interrelated. Finally, the advantages of this method are demonstrated through experiments using AGR compositional verification based on geodesic distance.

This work's primary contributions can be summarized as shown below:

- 1) A new compositional verification framework based on geodesic distance is proposed.
- 2) A novel system decomposition method based on geodesic distance is proposed to solve the decomposition problem in AGR compositional verification.

The rest of this paper is described below. Section II summarizes previous related work. Section III introduces the preliminaries used in this paper. Section IV describes our partition algorithm. Section V shows how to apply compositional verification based on geodesic distance. Section VI illustrates our approach with a complete example. Experimental results are presented in section VII. section VIII provides a comprehensive conclusion and highlights avenues for future research.

## II. RELATED WORK

Jamieson et al. [13] introduce a framework that leverages the  $L^*$  algorithm [14] and the noncircular assume-guarantee (AG-NC) rule to automate the generation of assumptions for AGR.  $L^*$  algorithm is utilized to learn a language which is not yet identified or categorized as a regular language.  $L^*$  algorithm can generate a Deterministic Finite-State Automaton (DFA) that can recognize and accept this language. References [15], [16], and [17] improve  $L^*$  algorithm by reducing the size of learning alphabet.

The framework proposed by Shang-Wei et al. [18] applies a systematic approach to analyze the system model and identify the root causes of the counterexamples. It then performs appropriate modifications or refinements to eliminate these problematic behaviors. Whenever a model checker gives a counterexample, a class of counterexamples that exhibit similar problematic behavior will be eliminated. The process of eliminating counterexamples in the proposed framework follows a compositional approach.

By formulating the problem as a hypergraph partitioning problem and implementing a proof rule that exhibits circularity and symmetry, Wonhong et al. [19] simplify the counterexample elimination process by automatically grouping components and minimizing the number of premises and assumptions required. The hypergraph partitioning algorithm used by them decomposes the set of variables in the system model into  $n$  disjoint subsets, enabling more efficient processing and analysis of the counterexample elimination problem. Each of these variable partitions  $X_i$  resulting from the hypergraph partitioning algorithm corresponds to a specific component  $M_i$  within the system.

By providing a systematic approach to identifying and addressing blocking behavior in a compositional verification framework, the algorithms proposed by Robi et al. [20] contribute to improving the reliability and correctness of discrete event systems. They introduce the counterexample calculation process of two kinds of common abstract rules.

David et al. [21] present  $CSim^2$ , a framework for top-down validation using a compositional rely-guarantee-based approach. The Isabelle/HOL theoremproving machine is used. They create a new concurrent imperative language called CSimpl. To maintain soundness and assurance properties across different CSimpl standards or abstract levels, a simulation-based framework is introduced.

Yang et al. [22] contribute a random and meristic assume guarantee compositional verification framework for probabilistic automata by utilizing the  $NL^*$  algorithm. The framework can rapidly identify and conclude when the system fails to satisfy the expectant properties. They use PRISM tool to implement the framework.

By combining deep learning, Ruiyang et al. [23] aim to address the limitations of traditional model checking approaches and enhance their performance. Using first-order recursive logic, the model checking issue is transformed into a game for loss and win with complete information. By improving the neural MCTS algorithm to handle loops, Xu et al. address a significant challenge in searching the state space. Their enhancements enable more robust and efficient exploration, improving the overall performance and effectiveness of the algorithm in complex environments. Experiments were conducted based on two labeled transition systems, numerical game and Dining Philosophers.

Mbarka et al. [24] use the CSP language and PAT model checker to verify properties of Hadoop schedulers. The analysis of the OpenCloud scheduler serves as a practical case study to demonstrate the real-world applicability of their work. They verify schedulability, resources-deadlock freeness and fairness of Hadoop by integrating model checking techniques and simulation.

Asma et al. [25] use the SPIN model checker and Linear Temporal Logic to verify the peer to peer system called WebRTC. They utilize an intermediate format called IF as an intermediate language for generating a Promela model. Promela is a modeling language. By expressing desired properties as LTL properties, the verification and validation process provides a formal and systematic approach to ensure that the model behaves as intended and complies with specified requirements.

Kenji et al. [26] utilize two different strategies to verify the reflective system. To elaborate, a metalevel labeled transition system (MLTS) is a formalism used for modeling a reflective system. SPIN is used as model checker to verify a system. They demonstrate that symmetry reduction and divergence-sensitive stutter bisimulation techniques can be used effectively in the context of MLTS models. For the reconnaissance robot system, MLTS is used to model its self-adaptation capabilities. In the case of the internet-of-things system, MLTS is employed to model its dynamic evolution.

AlSobeh [27] integrates techniques such as statistical model checking, AOP modularity, and formal methods, and employs a propositional model to observe the properties and constraints of Electronic Human Resource (EHR) systems,

providing a rigorous approach for verifying the reliability of EHR systems.

AlSobeh et al. [28] utilize Aspect-Oriented Programming (AOP) to formally verify the dynamic behaviors of blockchain systems. Through a case study of a cryptocurrency system, they outline the workflow that ranges from analyzing contextual data, to conducting statistical model checking using aspects, and finally to evaluating service quality.

The most relevant method is that of Lin [1]. This method solves the partition problem in AGR by dissatisfied cores of bounded model checking (BMC) formulas. The components associated with the unsatisfied core are grouped into  $M_1$ . They refine the abstraction of  $M_2$  based on interpolants. This method is effective, but it does not work well for some systems. Here, geodesic distance is used to solve the partition problem.

### III. PRELIMINARIES

This section presents concise explanations of important terms.

*Definition 1 (Concurrent Subsystem Diagram):* The concurrent subsystem diagram is denoted by  $G(V,E)$ .  $V$  is the set of vertices and  $E$  is the set of edges. If two components share variables, there is an undirected edge between the vertices of the two components.

*Definition 2 (Geodesic Distance):* For vertex  $v_1 \in V$ ,  $v_2 \in V$ , the geodesic distance of any two vertices is the number of edges in the shortest path between two vertices, denoted as  $geodis(v_1, v_2)$ . The geodesic distance between two non-connected vertices in the diagram is defined as infinity.

*Definition 3 (Eccentricity):* For vertex  $v \in V$ , the eccentricity of  $v$  is the maximum geodesic distance between  $v$  and the other vertices  $u \in V - \{v\}$ , denoted as  $eccen(v)$ . The eccentricity of  $v$  captures how far  $v$  is from the farthest vertex in the diagram.

*Definition 4 (Diameter of Diagram):* The diameter of a diagram is the maximum eccentricity for all vertices of the diagram. The diameter represents the maximum distance between all vertex pairs in the diagram.

*Definition 5 (Peripheral Vertices):* Peripheral vertices are vertices on the diameter.

### IV. GROUP THE VERTICES TO GET INITIAL PARTITION

Our grouping objectives are as follows:

- The vertices in each group are connected.
- The similarity (Definition 7) of each pair of vertices within each group is as high as possible.
- The eccentricity of the vertices in each group has at most two values (This is to ensure that each group does not contain too many vertices).
- At most one group has only one vertex.

The following illustrates how to group the vertices according to the eccentricity and get the initial partition.

To group vertices, follow these steps: (1) Get the set of peripheral vertices (Definition 5). Neighbors of peripheral vertices are grouped with peripheral vertices. If multiple peripheral vertices have the same immediate neighbors, they are merged into a group. (2) Get the ungrouped set of vertices. If the ungrouped set is empty, the algorithm ends. (3) Find the vertex set with the greatest eccentricity (denoted as MAXECC) among the ungrouped vertex sets. For vertex  $i$  ( $\text{eccen}(i) = \text{MAXECC}$ ),  $j$  is ungrouped vertex, if  $\text{geodis}(i, j) = 1$  (that is to say,  $j$  is the immediate neighbor of  $i$ ), combine  $j$  and  $i$  into a group. For vertex  $i$  ( $\text{eccen}(i) = \text{MAXECC}$ ), vertex  $m$  ( $\text{eccen}(m) = \text{MAXECC}$ ), if  $i$  and  $m$  have the same immediate neighbor, combine  $i$  and  $m$  into a group. Repeat the process until all vertices are grouped. Algorithm 1 offers a clear and concise representation of the proposed grouping algorithm's logic and operations. The time complexity of Algorithm 1 is  $O(n^4)$ .

Next, initial partition  $M_1, M_2$  can be obtained according to the group number of each component. First, put component  $C_i$  that is related to the verified property into  $M_1$ . If component  $C_j$  and  $C_i$  have the same group number, i.e.,  $\text{group}(j) = \text{group}(i)$ , then put component  $C_j$  into  $M_1$ . If Component  $C_k$  is the neighbor of  $C_i$  (i.e.  $\text{geodis}(C_k, C_i) = 1$ ) and  $C_k$  is not a member of  $M_1$ , then put component  $C_k$  into  $M_1$  as well. Components that do not belong to  $M_1$  are placed in  $M_2$ . Algorithm 2 shows the procedure for obtaining the initial partition. The time complexity of Algorithm 2 is  $O(n)$ .

Take a simple system as an example. Figure 2(a) is the system diagram and Figure 2(b) is the result of its grouping. The details of obtaining the initial partition  $M_1, M_2$  is as follows:

- 1) For a given diagram  $G(V, E)$ ,  $i \in V, j \in V$ , calculate the geodesic distance  $\text{geodis}(i, j)$  between vertex  $i$  and  $j$ .
- 2) Calculate the eccentricity of each vertex.  $\text{eccen}(a) = 3$ ,  $\text{eccen}(b) = 2$ ,  $\text{eccen}(c) = 4$ ,  $\text{eccen}(d) = 4$ ,  $\text{eccen}(e) = 3$ ,  $\text{eccen}(f) = 4$ ,  $\text{eccen}(g) = 3$ ,  $\text{eccen}(h) = 3$ ,  $\text{eccen}(i) = 4$ ,  $\text{eccen}(j) = 3$ . Vertices  $c, d, f$  and  $i$  have the greatest eccentricity, so they are peripheral vertices.
- 3) Get one vertex out from vertices with the greatest eccentricity: vertex  $c$ . Set  $\text{group}(c) = 1$ . Because  $a$  and  $d$  are immediate neighbors of  $c$  (for  $\text{geodis}(c, a) = 1$ ,  $\text{geodis}(c, d) = 1$ ), set  $\text{group}(a) = 1$  and  $\text{group}(d) = 1$ .
- 4) Get one vertex with the greatest eccentricity from ungrouped vertices: vertex  $f$ . Set  $\text{group}(f) = 2$ . Because  $e$  and  $g$  are immediate neighbors of  $f$  (for  $\text{geodis}(f, e) = 1$ ,  $\text{geodis}(f, g) = 1$ ), set  $\text{group}(e) = 2$  and  $\text{group}(g) = 2$ .
- 5) Get one vertex with the greatest eccentricity from ungrouped vertices: vertex  $i$ . Set  $\text{group}(i) = 3$ . Because  $h$  and  $j$  are immediate neighbors of  $i$  (for  $\text{geodis}(i, h) = 1$ ,  $\text{geodis}(i, j) = 1$ ), set  $\text{group}(h) = 3$  and  $\text{group}(j) = 3$ .
- 6) Vertex  $b$  is the unique ungrouped vertex, so  $\text{group}(b) = 4$ .

So far, each vertex has a group number. Group 1 has the component  $a, c$  and  $d$ . Group 2 has the component  $e,$

$f$  and  $g$ . Group 3 has the component  $h, i$  and  $j$ . Group 4 has the component  $b$ .

- 7) If the verified property has relation with component  $a$ , put components  $a, c$  and  $d$  into  $M_1$  (Because component  $a$  is in a group that contains  $c$  and  $d$ ).  $M_1 = \{a, c, d\}$ . Component  $b$  is the immediate neighbor of  $a$  (for  $\text{geodis}(a, b) = 1$ ), so put  $b$  into  $M_1$  as well.  $M_1 = \{a, b, c, d\}$ .
- 8) In the end, we get the initial partition:  $M_1 = \{a, b, c, d\}$ ,  $M_2 = \{e, f, g, h, i, j\}$

The benefits of such grouping are explained below through the similarity between vertices.

*Definition 6 (Neighbourhood of Vertex):* Given undirected graph  $G=(V, E)$ , for vertex  $u \in V$ , the neighbourhood of  $u$  is defined as:

$$\Gamma(u) = \{v \mid (u, v) \in E\} \cup \{u\}$$

*Definition 7 (Similarity between Two Vertices):* Use normalized common neighborhood sizes to measure the similarity between two vertices  $u, v \in V$ :

$$\sigma(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{\sqrt{|\Gamma(u)| \times |\Gamma(v)|}}$$

The greater  $\sigma(u, v)$ , the greater similarity between vertices  $u$  and  $v$ .

The neighborhood of each vertex in Figure 2 is as follows:  $\Gamma(a) = \{abcd\}$ ,  $\Gamma(b) = \{abegh\}$ ,  $\Gamma(c) = \{acd\}$ ,  $\Gamma(d) = \{acd\}$ ,  $\Gamma(e) = \{bef\}$ ,  $\Gamma(f) = \{efg\}$ ,  $\Gamma(g) = \{bgf\}$ ,  $\Gamma(h) = \{bhi\}$ ,  $\Gamma(i) = \{hij\}$ ,  $\Gamma(j) = \{bij\}$ .

$$\sigma(a, c) = \frac{\sqrt{3}}{2}, \sigma(a, d) = \frac{\sqrt{3}}{2}, \sigma(a, b) = \frac{1}{\sqrt{5}}.$$

Two vertices with the same eccentricity have the same similarity to their common neighbor if their neighborhood is the same, as shown in Figure 2, vertex  $c$  and  $d$ .

In Figure 2, vertices  $a, c$ , and  $d$  are grouped together while vertex  $b$  is not grouped with  $a$ . This is because  $\sigma(a, c) = \sigma(a, d)$  and  $\sigma(a, c) > \sigma(a, b)$ .

For each group obtained by algorithm 1, the similarity of the vertices within the group is high, that is to say, the components within the group are the most relevant.

## V. COMPOSITIONAL VERIFICATION BASED ON GEODESIC DISTANCE

In this section, we introduce the application of AGR compositional verification using geodesic distance. The verification process, shown in Figure 3, are as follows:

- 1)  $K$ , the geodesic distance of the vertex, the initial value is 2.
- 2)  $A_2$ , an over-approximation of  $M_2$ , the initial value is TRUE. Then check whether the condition  $M_1 \parallel A_2 \models \varphi$  holds. If it holds, then the conclusion  $M \models \varphi$  can be got. If it does not hold, assume it reaches  $\neg\varphi$  in  $m$  steps.
- 3) Use the  $m$ -step bounded model checking formula. If the formula is satisfiable, then the conclusion  $M \models \varphi$  does not hold. If the formula is unsatisfiable, go to step 4.
- 4) Analyze whether  $k \leq s-1$  ( $s$  is the number of subsystems). If  $k > s-1$ , the conclusion  $M \models \varphi$  holds.



**Algorithm 1** GROUPING

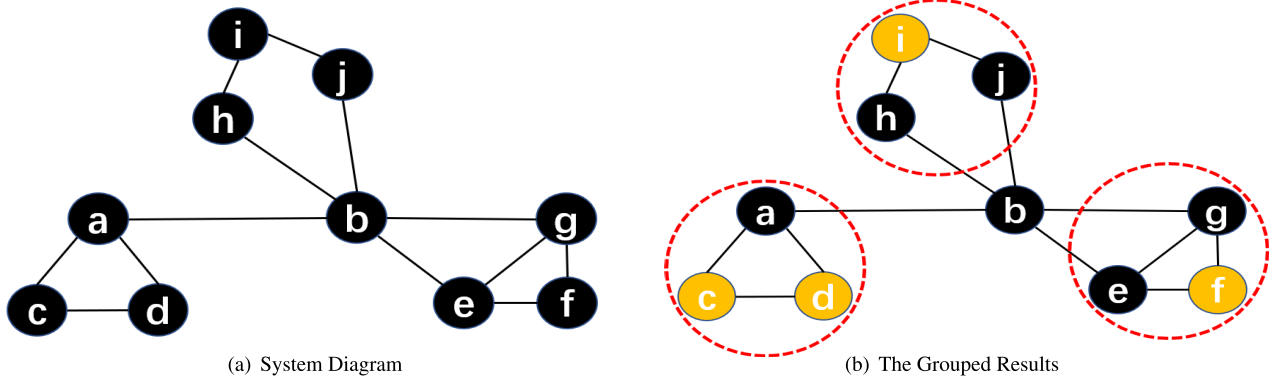
---

```

Input:  $C_1, C_2, \dots, C_n$  // a set of components
Output:  $\text{group}(1), \text{group}(2), \dots, \text{group}(n)$  // the group number of each component
begin
   $\text{group}(i)=0$ ; //  $i=1 \dots n$ ,  $n$  is the number of components
   $k=1$ ; //  $k$  is the group number
  calculate  $\text{geodis}(i, j)$ ; // the geodesic distance between vertex  $i$  and  $j$ 
  calculate  $\text{eccen}(i)$ ; // the eccentricity of vertex  $i$ ; Vertex  $i$  denotes component  $C_i$ 
  while There are ungrouped vertices do
    Calculate the maximum eccentricity(MAXECC) of the ungrouped vertices;;
    for  $i = 1; i \leq n; i++$  do
      Find the ungrouped vertices  $i$  and  $\text{eccen}(i)=\text{MAXECC}$ ;
       $\text{group}(i) \leftarrow k$ ;
      for  $j = 1; j \leq n; j++$  do
        find the ungrouped neighbors  $j$  of vertex  $i$ ;
         $\text{group}(j) \leftarrow k$ ;
        for  $jn = 1; jn \leq n; jn++$  do
          find the ungrouped neighbors  $jn$  of vertex  $j$  and  $\text{eccen}(jn)=\text{eccen}(i)$ ;
           $\text{group}(jn) \leftarrow k$ ;
        end
      end
    end
     $k++$ ;
  end
end
Output  $\text{group}(1), \text{group}(2), \dots, \text{group}(n)$ ;
end

```

---



**FIGURE 2.** A Simple System. The yellow vertices  $c, d, f,$  and  $i$  in the figure are peripheral vertices.

**Algorithm 2** PARTITION

---

```

Input:  $C_1, C_2, \dots, C_n$ ; // a set of components
Output:  $M_1, M_2$ 
 $M_1 \leftarrow \emptyset$ ;
 $M_2 \leftarrow \emptyset$ ;
Suppose Component  $C_i$  is related to the property  $\varphi$ ;
for  $j = 1; j \leq n; j++$  do
  if  $\text{group}(j)=\text{group}(i) \vee \text{geodis}(C_j, C_i)=1$  then
     $M_1 \leftarrow M_1 \cup \{C_j\}$ ;
  end
  //  $\text{group}(j)=\text{group}(i)$  denotes  $C_j$  and
  //  $C_i$  belong to the same group;
  //  $\text{geodis}(C_j, C_i)=1$  denotes  $C_j$  is a
  // direct neighbor of  $C_i$ ;
end
 $M_2 \leftarrow \text{AllComponents} - M_1$ ;

```

---

If  $k \leq s-1$ , find the neighbors ( $k$ -NN) with geodesic distance  $k$  of the subsystems related to the property.

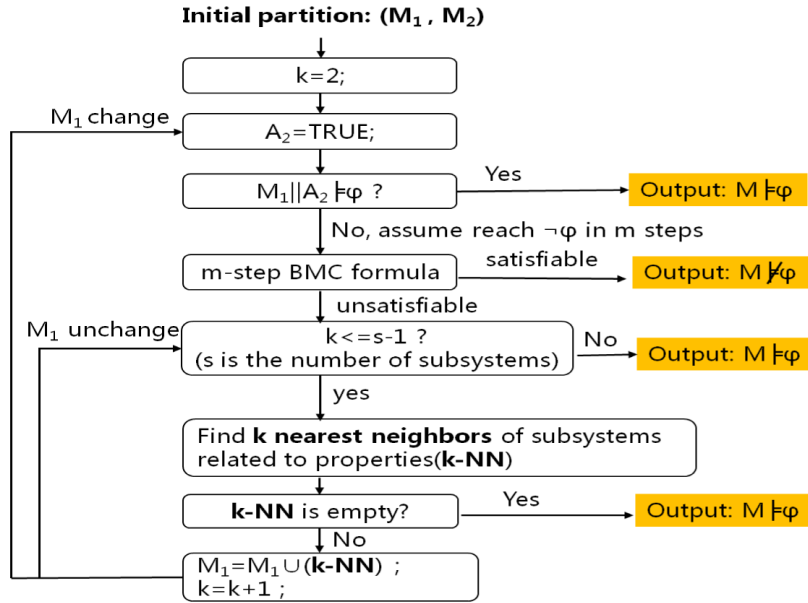
If  $k$ -NN is empty, the conclusion  $M \models \varphi$  holds. If  $k$ -NN is not empty, change  $M_1$  to the union of  $M_1$  and  $k$ -NN.  $k = k + 1$ .

- 5) If  $M_1$  unchange, go to step 4.
- 6) If  $M_1$  change, go to step 2.

**VI. A COMPLETE EXAMPLE**

The following illustrates how the proposed approach works using a complete example. First, use the approach to get the initial partition of the system. Then, verify the system using compositional verification based on geodesic distance. In [1], the authors employ a technique called interpolation to estimate the behavior of a system as it transitions from one state to another.

We develop a model of a six-bit counter that is composed of six individual components, as shown in Figure 4. Each component  $unit_i$  for  $i \in \{1, 2, 3, 4, 5, 6\}$  has three variables which are all of Boolean type:  $bit_i$ ,  $in_i$ ,  $out_i$ .  $bit_i$  represents the current binary value of  $unit_i$ .  $in_i$  represents whether to assert the carry value of  $unit_i$ .  $out_i$  is a signal that determines



**FIGURE 3. Verification Process.**  $M \models \varphi$  denotes the system  $M$  satisfies the property  $\varphi$ .  $M \not\models \varphi$  denotes the system  $M$  does not satisfy the property  $\varphi$ .

whether the current bit value of  $unit_i$  should be propagated or carried over to the next stage.  $I_i$  is the initial condition and  $T_i$  denotes the relation that describes how each of the two components transition from one state to another in the system. Their encoding are as described below.

- $I_1: \neg bit_1 \wedge in_1$
- $I_i: \neg bit_i \quad // i=2, \dots, 6$
- $T_1: (bit_1' \leftarrow bit_1 \oplus in_1) \wedge (out_1' \leftarrow bit_1 \wedge in_1) // \oplus$  is XOR
- $T_i: (in_i' \leftarrow out_{i-1}) \wedge (bit_i' \leftarrow bit_i \oplus in_i) \wedge (out_i' \leftarrow bit_i \wedge in_i) // i=2, \dots, 6$

Suppose to verify the property  $\varphi = \neg(out_2 \wedge bit_2 \wedge in_2)$ . First, construct the system diagram. For  $i \in \{1, 2, \dots, 6\}$ ,  $unit_i$  denotes vertex  $C_i$  of the diagram.  $Out_1$  variable which belongs to  $unit_1$  appears in the  $unit_2$ , so there is an edge between vertex  $C_1$  and vertex  $C_2$ . By analogy, the system diagram is shown in Figure 5(a).

Next, group the system. The eccentricity of each vertex is shown in Table 1. The geodesic distance of two vertices is shown in Table 2. Find the first vertex with the greatest eccentricity  $C_1$ , and set the grouping number of  $C_1$  to 1, that is,  $group(C_1) = 1$ . For  $geodis(C_1, C_2) = 1$ , set  $group(C_2) = 1$ . Find vertex  $C_6$  with the highest eccentricity among the ungrouped vertices, and set  $group(C_6) = 2$ . For  $geodis(C_5, C_6) = 1$ , set  $group(C_5) = 2$ . Find vertex  $C_3$  with the highest eccentricity among the ungrouped vertices, and set  $group(C_3) = 3$ . For  $geodis(C_3, C_4) = 1$ , set  $group(C_4) = 3$ .

So far, the system has been divided into 3 groups, as shown in Figure 5(b). Group 1 has component  $C_1$  and  $C_2$ . Group 2 has component  $C_5$  and  $C_6$ . Group 3 has component  $C_3$  and  $C_4$ .

**TABLE 1. Eccentricity of vertices.**

Vertex	Eccentricity
$C_1$	5
$C_2$	4
$C_3$	3
$C_4$	3
$C_5$	4
$C_6$	5

**TABLE 2. Geodesic distance between vertices.**

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$	$C_6$
$C_1$	0	1	2	3	4	5
$C_2$	1	0	1	2	3	4
$C_3$	2	1	0	1	2	3
$C_4$	3	2	1	0	1	2
$C_5$	4	3	2	1	0	1
$C_6$	5	4	3	2	1	0

Then obtain the initial partition of the system. Since the property  $\varphi$  has relation with  $C_2$  and  $C_2$  belongs to group 1, put  $C_1$  and  $C_2$  which are included in group 1 into  $M_1$ . For  $geodis(C_2, C_3)=1$ , put  $C_3$  into  $M_1$  too. Therefore, the initial partition of the system is obtained:  $M_1 = \{C_1, C_2, C_3\}$ ,  $M_2 = \{C_4, C_5, C_6\}$ .

Finally, use AGR compositional verification to verify the system. Construct an abstraction  $A_1$  for  $M_1$  and an abstraction  $A_2$  for  $M_2$ .

In the initial state, make  $A_2$ 's transition relation true, and set  $A_1$  to be  $M_1$ . TRUE is known as the weakest over-approximation, which means that it assumes all possible transitions are valid or true.  $A_1$  contains variables  $bit_1, in_1, out_1, bit_2, in_2, out_2, bit_3, in_3, out_3$ . Variables contained in  $A_2$  denoted by  $X_2$ .  $X_2$  can be any values. For  $i \in \{0, 1, \dots, 8\}$ ,

```

MODULE unit1
  var bit1:bool;
  var in1:bool;
  var out1:bool;

  init(bit1):=FALSE;
  init(in1):=TRUE;

  next(bit1):=bit1 xor in1;
  next(out1):=bit1 & in1;
END MODULE

```

```

MODULE uniti
  var biti:bool;
  var ini:bool;
  var outi:bool;
  //i=2, 3, 4, 5, 6
  init(biti):=FALSE;

  next(ini):=outi-1;
  next(biti):=biti xor ini;
  next(outi):=biti & ini;
END MODULE

```

FIGURE 4. The counter example.

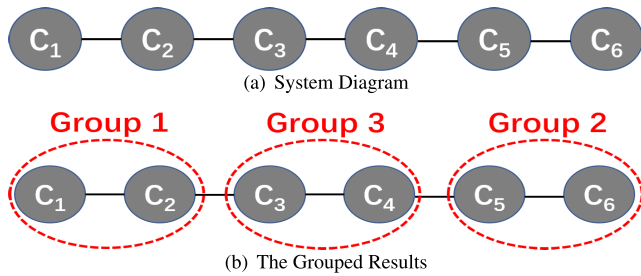


FIGURE 5. 6-bit counter.

$State_i$  is the state of  $A_1 \parallel A_2$ , as shown in Figure 6. For convenience, the value of 0 is FALSE, and the value of 1 is TRUE.  $State_0$  is the initial state.  $A_1 \parallel A_2$  is always switching between  $state_1, state_2, \dots$ , and  $state_8$ . The value of  $bit_3, bit_2$  and  $bit_1$  is from 000, 001, 010, 011, 100, 101, 110 to 111. In these states of the system,  $out_2, bit_2$  and  $in_2$  are never TRUE at one time. After the first iteration, it can be concluded that  $A_1 \parallel A_2 \models \varphi$ . Because  $A_1, A_2$  are abstractions of  $M_1, M_2$  respectively, the conclusion can be drawn that  $M_1 \parallel M_2 \models \varphi$ .

## VII. EXPERIMENTS

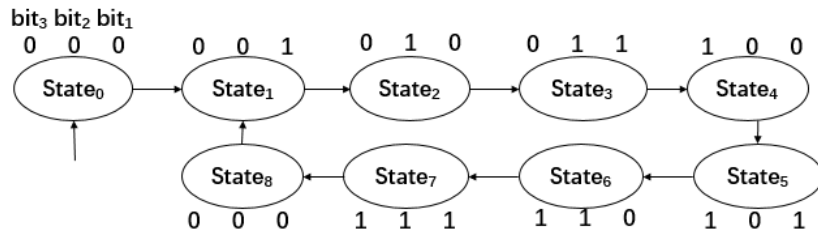
Use following systems as benchmarks to highlight the advantages and strengths of our approach over existing methods.

- **Flexible Manufacturing Systems (FMS).** A FMS [29] is responsible for manufacturing blocks with cylindrical painted pins by transforming raw blocks and raw pegs through a series of manufacturing operations. Buffers are used to connect devices, and each buffer has a capacity of one part. Buffer overflow and underflow issues can occur within a FMS. Verify that each buffer within a system should not overflow.
- **Dining Philosophers (DP).** The DP [30] problem serves as an example of a resource sharing challenge in concurrent programming. In this problem, a group of philosophers are seated in a manner where they form a circle around a table, with each philosopher having a fork to their left and right. Each philosopher requires two

forks to eat their meal. Verify the property that it is not possible for any two neighboring philosophers to eat at one time.

- **AIP Manufacturing System (AIP).** AIP [31] manufacturing system is a production system that involves the use of two distinct types of materials to produce two different products. It consists of several key components, including the I/O station, transport units, assembly stations, external loops, and a central loop. The AIP system has the problem of disordered manufacturing. Verify that the paths or directions of the two distinct types of materials should be reversed.
- **Synchronous Bus Arbiters (SBA).** The SBA [30], [32] refers to a specific protocol used in synchronous digital circuits for bus arbitration. A bus consists of nodes connected in a loop. A special token is sequentially passed from one node to another in a predetermined order. The token circulates among the nodes in the network, and only the node possessing the token is permitted to use the bus for transmitting data. Verify the properties that only one node has the permission to transmit data on the bus, that is, two nodes cannot control the bus at one time.
- **MSI Cache Coherence Protocol (MSI).** In the MSI [30], [32] cache coherence protocol,  $n$  nodes share a memory. Each individual node is equipped with a cache. The memory and the caches of the nodes are connected by a bus. Verify the property that only one node has the permission to transmit data on the bus, that is, two nodes cannot control the bus at one time.

The PAT model checker [33] is used to obtain the provided experimental results. The experimental environment is a 64-bit win7 laptop with 4GB Random Access Memory and i5-3230M processor. The language used for describing system models in our system is a simplified version of the input language used by NuSMV. In all of our experimental scenarios, we have observed that every tested property holds true and is satisfied. Compare four verification methods: McMillan's interpolation-based compositional verification (denoted as Mc-ITP), Lin's compositional verification (denoted as C-ITP), C-ITP method plus abstracting  $M_1$  (denoted as C-ITP<sub>A</sub>),



	$M_1$									$M_2$
	$C_3$			$C_2$			$C_1$			$C_4  C_5  C_6$
	bit <sub>3</sub>	in <sub>3</sub>	out <sub>3</sub>	bit <sub>2</sub>	in <sub>2</sub>	out <sub>2</sub>	bit <sub>1</sub>	in <sub>1</sub>	out <sub>1</sub>	$X_2$
State <sub>0</sub>	0			0			0	1		ANY
State <sub>1</sub>	0	0	0	0	0	0	1	1	0	ANY
State <sub>2</sub>	0	0	0	1	1	0	0	1	1	ANY
State <sub>3</sub>	0	0	0	1	0	0	1	1	0	ANY
State <sub>4</sub>	1	1	0	0	1	1	0	1	1	ANY
State <sub>5</sub>	1	0	0	0	0	0	1	1	0	ANY
State <sub>6</sub>	1	0	0	1	1	0	0	1	1	ANY
State <sub>7</sub>	1	0	0	1	0	0	1	1	0	ANY
State <sub>8</sub>	0	1	1	0	1	1	0	1	1	ANY
State <sub>1</sub>	0	0	0	0	0	0	1	1	0	ANY

FIGURE 6. The state of 6-bit counter system.

TABLE 3. Verification results.

system	n	$\varphi$	1					2					3					4					5				
			Mc-ITP	C-ITP	C-ITP <sub>A</sub>	C-ITP <sub>P+A</sub>	Geo-AGR	Mc-ITP	C-ITP	C-ITP <sub>A</sub>	C-ITP <sub>P+A</sub>	Geo-AGR	Mc-ITP	C-ITP	C-ITP <sub>A</sub>	C-ITP <sub>P+A</sub>	Geo-AGR	Mc-ITP	C-ITP	C-ITP <sub>A</sub>	C-ITP <sub>P+A</sub>	Geo-AGR	Mc-ITP	C-ITP	C-ITP <sub>A</sub>	C-ITP <sub>P+A</sub>	Geo-AGR
FSM_02	8	6	2.17	1.25	2	0.38	1.71																				
FSM_04	16	12	7.49	12.06	7.35	0.64	2.867																				
FSM_06	24	18	20.72	38.41	19.44	1.19	4.346																				
FSM_08	32	24	*	76.77	46.18	1.87	5.828																				
FSM_10	40	30	83.32	265.45	272.97	2.82	7.292																				
FSM_12	48	36	*	*	*	4.06	8.756																				
FSM_14	56	42	191.9	257.6	191.3	5.44	10.122																				
FSM_16	64	48	297.2	359.1	283.2	7.18	11.571																				
FSM_18	72	54	367.1	513.8	431.4	9.25	13.02																				
FSM_20	80	60	505.0	716.3	577.6	11.62	14.481																				
FSM_24	96	72	-	1018.7	884.7	17.85	17.39																				
FSM_30	120	90	-	-	-	30.86	21.751																				
DP_04	8	4	288.23	9.24	2.82	1.01	0.74																				
DP_06	12	6	11.62	*	4.59	2.42	1.168																				
DP_08	16	8	16.74	*	7.94	3.95	1.574																				
DP_10	20	10	24.06	*	14.04	6.05	6.692																				
DP_20	40	20	64.81	*	52.71	27.64	15.454																				
DP_30	60	30	280.36	*	306.84	37.83	29.012																				
AIP_01	8	2	2.34	11.61	4.8	5.35	3.267																				
AIP_02	16	4	7.52	172.61	14.76	42.84	7.345																				
AIP_04	32	8	34.20	*	94.04	86.66	19.179																				
AIP_06	48	12	89.78	*	309.97	132.46	24.709																				
AIP_08	64	16	196.5	*	651.27	185.43	31.038																				
AIP_10	80	20	350.3	*	1069.1	244.6	38.824																				
AIP_11	88	22	450.7	*	1546.2	273.71	42.707																				
AIP_12	96	24	613.1	*	-	315.9	46.626																				
SBA_02	8	12	*	137.33	12.82	3.9	4.646																				
SBA_03	12	18	*	593.58	211.1	7.07	7.35																				
SBA_04	16	24	*	1067.1	422.79	11.61	10.2																				
SBA_05	20	30	*	1600.9	649.6	17.2	13.313																				
SBA_06	24	36	*	-	898.3	24.23	16.69																				
SBA_07	28	42	*	-	1035.7	33.12	20.238																				
SBA_08	32	48	*	-	1287.2	44.3	24.153																				
SBA_09	36	54	*	-	1533.5	55.66	28.067																				
SBA_10	40	60	*	-	-	70.06	32.181																				
MSL_02	8	1	0.42	2.87	0.7	1.51	0.345																				
MSL_03	11	3	3.2	-	-	4.5	1.791																				
MSL_04	14	6	-	-	-	13.62	4.646																				
MSL_05	17	10	*	*	*	*	8.529																				

n: number of components; | $\varphi$ |: number of verified properties;  
 \*: out of memory; -: time out(30 minutes) verification time: in seconds

is denoted as Geo-AGR. Table 3 shows the results of experiments.

Mc-ITP approach gets initial partition randomly. In C-ITP and C-ITP<sub>A</sub>,  $M_1$  contains the first four components and  $M_2$  contains the remaining components. Users can specify the input order. C-ITP<sub>P+A</sub> approach uses a partition method that is derived from unsatisfiability core of the BMC theorem. The initial decomposition is obtained by running this decomposition method in two steps.

Using our geo-distance based compositional verification method, the verification time is greatly reduced. For system FSM\_12, no matter which method Mc-ITP, C-ITP, or C-ITP<sub>A</sub> is adopted, it is always out of memory, but the result can be obtained within 10 seconds after using our method. For system FSM\_30, all the first three methods are time out (over 30 minutes), but the result can be obtained within 25 seconds after using our method. For system DP\_06, D\_08, DP\_10, DP\_20, DP\_30, AIP\_04, AIP\_06, AIP\_08, AIP\_10, AIP\_11, AIP\_12, SBA\_06, SBA\_07, SBA\_08, SBA\_09, and SBA\_10, using method C-ITP, all of them are out of memory or time out, but the results can be obtained quickly after using our method.

C-ITP<sub>P+A</sub> method uses a partition heuristic, so the validation efficiency of this method is much higher than that of Mc-ITP, C-ITP, and C-ITP<sub>A</sub>. However, our algorithm is much more efficient than C-ITP<sub>P+A</sub>, except for FSM\_02, FSM\_04, ... FSM\_20 (See the last 2 columns in Table 3).

Assume-guarantee reasoning compositional verification needs to decompose the system into  $M_1$  and  $M_2$ . The efficiency of verification has a great correlation with



partition. The proposed partition algorithm can ensure that  $M_1$  contains fewer components which are related to verification properties. This is the main reason why the proposed algorithm is superior to C-ITP<sub>P+A</sub>.

## VIII. CONCLUSION AND FUTURE WORK

This paper introduces a new decomposition algorithm for assume-guarantee reasoning. The decomposition algorithm can greatly reduce the number of components that  $M_1$  contains in AGR. A compositional verification framework based on geodesic distance is proposed. This framework offers a solution to mitigate the issue of state space explosion that often arises during compositional verification. In five parameterized test cases, the method takes the shortest time compared with McMillan's approach and Lin's interpolation-guided compositional verification. The grouping algorithm presented in this paper does not yield a unique result when applied to a system which graph is cyclic. For systems with a very large number of components, calculating the geodesic distance for every pair of components can become computationally expensive, especially for dense graphs. Future work will aim to address the limitations of the current grouping algorithm by exploring modifications to handle cyclic graphs, implementing efficient approximations for geodesic distance calculations, and investigating alternative grouping algorithms that are more suitable for large, dense systems.

## REFERENCES

- [1] S.-W. Lin, J. Sun, T. K. Nguyen, Y. Liu, and J. S. Dong, "Interpolation guided compositional verification (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Lincoln, NE, USA, Nov. 2015, pp. 65–74, doi: [10.1109/ASE.2015.33](https://doi.org/10.1109/ASE.2015.33).
- [2] R. Jhala and K. L. Mcmillan, "Interpolant-based transition relation approximation," *Log. Methods Comput. Sci.*, vol. 3, no. 4, pp. 1–17, Nov. 2007, doi: [10.2168/LMCS-3\(4\):12007](https://doi.org/10.2168/LMCS-3(4):12007).
- [3] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Logics of Programs*, vol. 131. Berlin, Germany: Springer, 1981, pp. 52–71, doi: [10.1007/BFb0025774](https://doi.org/10.1007/BFb0025774).
- [4] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 5, pp. 1512–1542, Sep. 1994, doi: [10.1145/186025.186051](https://doi.org/10.1145/186025.186051).
- [5] Q. Jean-Pierre and S. Joseph, "Specification and verification of concurrent systems in CESAR," in *Proc. Int. Symp. Programming*, vol. 137, M. Dezani-Ciancaglini and U. Montanari, Eds. 1982, pp. 337–351, doi: [10.1007/3-540-11494-7\\_22](https://doi.org/10.1007/3-540-11494-7_22).
- [6] E. M. Clarke, D. E. Long, and K. L. Mcmillan, "Compositional model checking," in *Proc. 4th Annu. Symp. Log. Comput. Sci.*, 1989, pp. 353–362.
- [7] O. Grumberg and D. E. Long, "Model checking and modular verification," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 3, pp. 843–871, May 1994, doi: [10.1145/177492.177725](https://doi.org/10.1145/177492.177725).
- [8] T. A. Henzinger, S. Qadeer, and S. K. Rajamani, "You assume, we guarantee: Methodology and case studies," in *Proc. Int. Conf. Comput. Aided Verification*, Vancouver, BC, Canada, 1998, pp. 440–451, doi: [10.1007/BFb0028765](https://doi.org/10.1007/BFb0028765).
- [9] A. Pnueli, "In transition from global to modular temporal reasoning about programs," in *Logics and Models of Concurrent Systems* (NATO ASI Series), vol. 13, K. R. Apt, Ed., Berlin, Germany: Springer, 1984, pp. 123–144, doi: [10.1007/978-3-642-82453-1\\_5](https://doi.org/10.1007/978-3-642-82453-1_5).
- [10] Q. Xu, W.-P. de Roever, and J. He, "The rely-guarantee method for verifying shared variable concurrent programs," *Formal Aspects Comput.*, vol. 9, no. 2, pp. 149–174, Mar. 1997, doi: [10.1007/bf01211617](https://doi.org/10.1007/bf01211617).
- [11] J. M. Cobleigh, G. S. Avrunin, and L. A. Clarke, "Breaking up is hard to do: An investigation of decomposition for assume-guarantee reasoning," in *Proc. Int. Symp. Software Test. Anal.*, 2006, pp. 97–108, doi: [10.1145/1146238.1146250](https://doi.org/10.1145/1146238.1146250).
- [12] H. Jiawei, P. Jian, and T. Hanghang, *Data Mining: Concepts and Techniques*. Burlington, MA, USA: Morgan kaufmann 2022.
- [13] J. M. Cobleigh, D. Giannakopoulou, and C. S. Pasareanu, "Learning assumptions for compositional verification," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science), vol. 2619, H. Garavel and J. Hatcliff, Eds., Berlin, Germany: Springer, 2003, pp. 331–346, doi: [10.1007/3-540-36577-X\\_24](https://doi.org/10.1007/3-540-36577-X_24).
- [14] D. Angluin, "Learning regular sets from queries and counterexamples," *Inf. Comput.*, vol. 75, no. 2, pp. 87–106, Nov. 1987, doi: [10.1016/0890-5401\(87\)90052-6](https://doi.org/10.1016/0890-5401(87)90052-6).
- [15] S. Chaki and O. Strichman, "Optimized L-based assume-guarantee reasoning," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science), vol. 4424, O. Grumberg and M. Huth, Eds., Berlin, Germany: Springer, 2007, pp. 276–291, doi: [10.1007/978-3-540-71209-1\\_22](https://doi.org/10.1007/978-3-540-71209-1_22).
- [16] M. Gheorghiu, D. Giannakopoulou, and C. S. Pasareanu, "Refining interface alphabets for compositional verification," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science), vol. 4424, O. Grumberg and M. Huth, Eds., Berlin, Germany: Springer, 2007, pp. 292–307, doi: [10.1007/978-3-540-71209-1\\_23](https://doi.org/10.1007/978-3-540-71209-1_23).
- [17] S. Nishant and C. Edmund, "SAT-based compositional verification using lazy learning," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 4590, W. Damm and H. Hermanns, Eds., Berlin, Germany: Springer, 2007, pp. 39–54, doi: [10.1007/978-3-540-73368-3\\_8](https://doi.org/10.1007/978-3-540-73368-3_8).
- [18] S.-W. Lin and P.-A. Hsiung, "Counterexample-guided assume-guarantee synthesis through learning," *IEEE Trans. Comput.*, vol. 60, no. 5, pp. 734–750, May 2011, doi: [10.1109/TC.2010.94](https://doi.org/10.1109/TC.2010.94).
- [19] N. Wonhong and A. Rajeev, "Learning-based symbolic assume-guarantee reasoning with automatic decomposition," in *Automated Technology for Verification and Analysis* (Lecture Notes in Computer Science), vol. 4218, S. Graf and W. Zhang, Eds., Berlin, Germany: Springer, 2006, pp. 170–185, doi: [10.1007/11901914\\_15](https://doi.org/10.1007/11901914_15).
- [20] R. Malik and S. Ware, "On the computation of counterexamples in compositional nonblocking verification," *Discrete Event Dyn. Syst.*, vol. 30, no. 2, pp. 301–334, Jun. 2020, doi: [10.1007/s10626-019-00305-w](https://doi.org/10.1007/s10626-019-00305-w).
- [21] D. Sanan, Y. Zhao, S.-W. Lin, and L. Yang, "CSim 2: Compositional top-down verification of concurrent systems using rely-guarantee," *ACM Trans. Program. Lang. Syst.*, vol. 43, no. 1, pp. 1–46, Mar. 2021, doi: [10.1145/3436808](https://doi.org/10.1145/3436808).
- [22] Y. Liu and R. Li, "Compositional stochastic model checking probabilistic automata via assume-guarantee reasoning," *Int. J. Networked Distrib. Comput.*, vol. 8, no. 2, pp. 94–107, 2020, doi: [10.2991/ijndc.k.190918.001](https://doi.org/10.2991/ijndc.k.190918.001).
- [23] R. Xu and K. Lieberherr, "On-the-fly model checking with neural MCTS," in *NASA Formal Methods* (Lecture Notes in Computer Science), vol. 13260, J. V. Deshmukh, K. Havelund, I. Perez, Eds., Cham, Switzerland: Springer, 2022, pp. 557–575, doi: [10.1007/978-3-031-06773-0\\_30](https://doi.org/10.1007/978-3-031-06773-0_30).
- [24] M. Soualhia, F. Khomh, and S. Tahar, "Failure analysis of Hadoop schedulers using an integration of model checking and simulation," 2021, *arXiv:2109.04196*.
- [25] A. El Hamzaoui, H. Bensaid, and A. En-Nouary, "Model checking of WebRTC peer to peer system," *Comput. Inf. Sci.*, vol. 12, no. 4, pp. 56–71, Oct. 2019, doi: [10.5539/cis.v12n4p56](https://doi.org/10.5539/cis.v12n4p56).
- [26] K. Tei, Y. Tahara, and A. Ohsuga, "Towards scalable model checking of reflective systems via labeled transition systems," *IEEE Trans. Softw. Eng.*, vol. 49, no. 3, pp. 1299–1322, Mar. 2023, doi: [10.1109/TSE.2022.3174408](https://doi.org/10.1109/TSE.2022.3174408).
- [27] A. AlSobeh, "OSM: Leveraging model checking for observing dynamic 1 behaviors in aspect-oriented applications," 2024, *arXiv:2403.01349*.
- [28] A. M. R. AlSobeh and A. A. Magableh, "BlockASP: A framework for AOP-based model checking blockchain system," *IEEE Access*, vol. 11, pp. 115062–115075, 2023.

- [29] M. H. de Queiroz, J. E. R. Cury, and W. M. Wonham, "Multitasking supervisory control of discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 15, no. 4, pp. 375–395, Dec. 2005.
- [30] Y. F. Chen, E. M. Clarke, A. Farzan, M. H. Tsai, Y. K. Tsay, and B. Y. Wang, "Automated assume-guarantee reasoning through implicit learning," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 6174, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Germany: Springer, 2010, pp. 511–526, doi: [10.1007/978-3-642-14295-6\\_44](https://doi.org/10.1007/978-3-642-14295-6_44).
- [31] R. J. Leduc, M. Lawford, and P. Dai, "Hierarchical interface-based supervisory control of a flexible manufacturing system," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 4, pp. 654–668, Jul. 2006, doi: [10.1109/TCST.2006.876635](https://doi.org/10.1109/TCST.2006.876635).
- [32] K. L. McMillan, "Symbolic model checking," in *Symbolic Model Checking*. Boston, MA, USA: Springer, 1993, doi: [10.1007/978-1-4615-3190-6\\_3](https://doi.org/10.1007/978-1-4615-3190-6_3).
- [33] J. Sun, Y. Liu, J. S. Dong, and J. Pang, "PAT: Towards flexible verification under fairness," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 5643, A. Bouajjani and O. Maler, Eds., Berlin, Germany: Springer, 2009, pp. 709–714, doi: [10.1007/978-3-642-02658-4\\_59](https://doi.org/10.1007/978-3-642-02658-4_59).



**XIAOYAN LIU** received the B.S. degree in computer science and technology from Henan Polytechnic University, Jiaozuo, in 2004, and the M.S. degree in information science from Xidian University, Xi'an, in 2007.

From 2007 to 2022, she was a Teacher with the College of Computer Science and Technology, Henan Polytechnic University, where she has been a Teacher with the School of Software, since 2022. She is the author of two books, more than ten articles. She holds three patents. Her research interests include big data application technology, deep learning, model checking, software engineering, and bioinformation.

• • •