A Fine-Grained Fullness-Guided Chaining Heuristic for Symbolic Reachability Analysis

Ming-Ying Chung, Gianfranco Ciardo, and Andy Jinqing Yu

Department of Computer Science and Engineering
University of California, Riverside
{chung, ciardo, jqyu}@cs.ucr.edu

Abstract. Chaining can reduce the number of iterations required for symbolic state-space generation and model-checking, especially in Petri nets and similar asynchronous systems, but requires considerable insight and is limited to a static ordering of the events in the high-level model. We introduce a two-step approach that is instead fine-grained and dynamically applied to the decision diagrams nodes. The first step, based on a precedence relation, is guaranteed to improve convergence, while the second one, based on a notion of node fullness, is heuristic. We apply our approach to traditional breadth-first and saturation state-space generation, and show that it is effective in both cases.

1 Introduction

BDD-based symbolic model checking is one of the most successful techniques to verify industrial hardware and embedded software systems, and symbolic reachability analysis is a fundamental step in symbolic model checking. It is well-known that the peak number of BDD nodes is often much larger than the final number of BDD nodes for symbolic reachability analysis. In this paper, we propose a new chaining technique to reduce this peak number.

For asynchronous concurrent systems, such as distributed software, network protocols, and various classes of Petri nets, chaining can reduce the peak memory usage and speed-up symbolic state-space generation by exploring events in a particularly favorable order. Chaining is normally applied as a modification of a strict breadth-first search (BFS), but it is also one of the factors behind the efficiency of the saturation algorithm. As introduced, however, chaining is limited to finding a good order in which to apply the high-level model events during the symbolic iterations.

In this paper, we propose a general and effective heuristic that uses a partial-order relation and the concept of decision diagram node fullness to guide the chaining order, independent of the high-level formalism used to model the system. Our definition of node fullness is related to, but different from, the BDD node density defined in. A detailed comparison can be found in Sect. 6.

* Work supported in part by the National Science Foundation under grants CNS-0501747 and CNS-0501748.
Sect. 2 gives background on structured models, decision diagrams, BFS-based and saturation-based symbolic state-space generation, and chaining. Sect. 3 details our main contribution, where a fine-grained chaining is applied dynamically using the current structure of the decision diagram, rather than the model events. Sect. 4 describes the modified symbolic state-space generation algorithms incorporating our heuristic and gives implementation details. Sect. 5 provides numerical results on a suite of models showing that our heuristic reduces the runtime and memory requirements of both BFS-based and saturation-based algorithms. Sect. 6 compares the newly proposed chaining heuristics with some related work. Finally, Sect. 7 concludes with directions for future research.

2 Preliminaries

We consider a discrete-state model \((\mathcal{S}, S^{\text{init}}, \mathcal{R})\), where \(\mathcal{S}\) is a finite set of states, \(S^{\text{init}} \subseteq \mathcal{S}\) are the initial states, and \(\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}\) is a transition relation. We assume the \((\text{global})\) model state to be a tuple of \(K\) local state variables, \((x_K, ..., x_1),\) where, for \(K \geq l \geq 1, x_l \in S_l = \{0, 1, ..., n_l-1\},\) with \(n_l > 0,\) is the \(l^{th}\) local state variable. Thus, \(\mathcal{S} = S_K \times \cdots \times S_1\) and we write \(R(i[K], ..., i[1], j[K], ..., j[1]),\) or \(R(i, j),\) if the model can move from current state \(i\) to next state \(j\) in one step.

2.1 Symbolic Encoding of State Space and Transition Relation

State-space generation consists of building the smallest set of states \(\mathcal{S} \subseteq \mathcal{S}\) satisfying \(\mathcal{S} \supseteq S^{\text{init}}\) and \(\mathcal{S} \supseteq \text{Img}(\mathcal{S}, \mathcal{R})\), where the image computation function gives the set of successor states: \(\text{Img}(\mathcal{S}, \mathcal{R}) = \{j : \exists i \in \mathcal{S}, \mathcal{R}(i, j)\}\). Most symbolic approaches to store the state space encode \(x_l\) using \(\lceil \log n_l \rceil\) boolean variables and a set of states \(Z\) using a BDD with \(\sum_{K \geq l \geq 1} \lceil \log n_l \rceil\) levels.

We prefer to discuss our approach in terms of ordered multi-way decision diagrams (MDDs) [14], where each variable \(x_l\) is directly encoded in a single level, using a node with \(n_l\) outgoing edges. MDDs can be implemented directly, the approach taken in our tool \texttt{SMARF} [3], or as an interface to BDDs [25].

Definition 1. An MDD over \(\mathcal{S}\) is an acyclic edge-labeled multi-graph where:

- Each node \(p\) belongs to a level in \(\{K, ..., 1, 0\}\), denoted \(p.lvl\).
- There is a single root node \(r^*\).
- Level 0 can contain only the terminal nodes, 0 and 1.
- A node \(p\) at level \(l > 0\) has \(n_l\) outgoing edges, labeled from 0 to \(n_l-1\). The edge labeled by \(i \in S_l\) points to node \(q\), with \(p.lvl > q.lvl\); we write \(p[i] = q\).

Then, to ensure canonicity, duplicate nodes are forbidden:

- Given nodes \(p\) and \(q\) at level \(l\), if \(p[i] = q[i]\) for all \(i \in S_l\), then \(p = q\), and we must use either the fully-reduced rule [1] that forbids redundant nodes:
  - No node \(p\) at level \(l\) can exist such that, \(p[i] = q\) for all \(i \in S_l\),
  - or the quasi-reduced rule [15] that restricts arcs spanning multiple levels:
  - The root is at level \(K\).
  - Given a node \(p\) at level \(l\), \(p[i].lvl\) is either \(l - 1\) or 0, for all \(i \in S_l\).