Interval Diagram Techniques for Symbolic Model Checking of Petri Nets

Karsten Strehl and Lothar Thiele

Computer Engineering and Networks Lab (TIK)
Swiss Federal Institute of Technology (ETH)
Gloriastrasse 35, 8092 Zurich, Switzerland
eMail: {strehl,thiele}@tik.ee.ethz.ch
WWW: http://www.tik.ee.ethz.ch

Abstract

Symbolic model checking tries to reduce the state explosion problem by implicit construction of the state space. The major limiting factor is the size of the symbolic representation mostly stored in huge binary decision diagrams. A new approach to symbolic model checking of Petri nets and related models of computation is presented, outperforming the conventional one and avoiding some of its drawbacks. Our approach is based on a novel, efficient form of representation for multi-valued functions called *interval decision diagram* (IDD) and the corresponding image computation technique using *interval mapping diagrams* (IMDs). IDDs and IMDs are introduced, their properties are described, and the feasibility of the new approach is shown with some experimental results.

1 Introduction

During the last years, a promising approach named *symbolic model checking* [2] was applied to many areas of system verification, even in industrial applications. This approach makes use of *binary decision diagrams* (BDDs) [1] that are an efficient representation of Boolean functions and allow for very fast manipulation.

Petri nets are commonly used to model and analyze the dynamic behavior especially of concurrent and asynchronous systems. They are related to other models such as many high-level and data-flow oriented models or models consisting of finite-state components communicating via FIFO channels. When applying conventional symbolic model checking techniques to such models of computation, several difficulties occur that question their usefulness in this area.

The traditional BDD-based method of automated verification suffers from the drawback that a binary representation of the Petri net and its state is required. Even the use of *multi-valued decision diagrams* (MDDs) [4] instead of BDDs cannot resolve the problem. *Interval diagram techniques*—using *interval decision diagrams* (IDDs) and *interval mapping diagrams* (IMDs)—have shown to be convenient for formal verification of, e.g., process networks [5]. They remedy some deficiencies of traditional approaches and often provide advantages regarding computation time and memory resources. The major enhancements of symbolic model checking with IDDs and IMDs are:

- No state variable bounds due to binary coding or complementation are necessary as with conventional symbolic model checking.
- The transition relation representation is quite compact as only "state distances" are stored instead of combinations of state and successor. Accordingly, an innovative technique for image computation is used.
- Due to the enhanced merging capabilities of IDDs and the abandonment of binary coding, state set descriptions are more compact than using BDDs.

In this paper, interval diagram techniques are applied to symbolic model checking of Petri nets. We briefly present the used interval diagrams and verification techniques and compare their runtime behavior with that of the conventional BDD approach.

2 Interval Diagram Techniques

For formal verification of, e.g., process networks [5], interval diagram techniques—using interval decision diagrams (IDDs) and interval mapping diagrams (IMDs)—have shown to be a favorable alternative to BDD techniques. This results from the fact that for this kind of models of computation, the transition relation has a very regular structure that IMDs can conveniently represent. While BDDs have to represent explicitly all possible state variable value pairs before and after a certain transition, IMDs store only the *state distance*—the difference between the state variable values before and after the transition. Especially for models with large numbers of tokens, this approach is reasonable and useful. IDDs are used to represent state sets during computations.

2.1 Interval Decision Diagrams

IDDs are a generalization of BDDs and MDDs—*multi-valued decision diagrams* [4]—allowing diagram variables to be integers and child nodes to be associated with intervals rather than single values. In Figure 1 a), an example IDD is shown. It represents the Boolean function $s(u, v, w) = (u \leq 3) \wedge (v \geq 6) \vee (u \geq 4) \wedge (w \leq 7)$ with $u, v, w \in [0, \infty)$.

Equivalent to BDDs, IDDs have a reduced and ordered form, providing a canonical representation of a class of Boolean functions—which is important with respect to efficient fixpoint computations often necessary for formal verification. Methods as the *If-Then-Else* operator ITE are defined similar to their BDD equivalents and may be computed as usual for decision diagram applications using a computed table to improve performance.

2.2 Interval Mapping Diagrams

IMDs are represented by graphs similar to IDDs. Their edges are labeled with functions mapping intervals onto intervals. The graph contains only one terminal node. Figure 1 b) shows an example IMD.

With regard to transition relations, IMDs work as follows. Each edge is labeled with a condition—the *predicate interval*—on its source node variable and the kind and amount of change—the *action operator* and the *action interval*—the variable is to undergo.

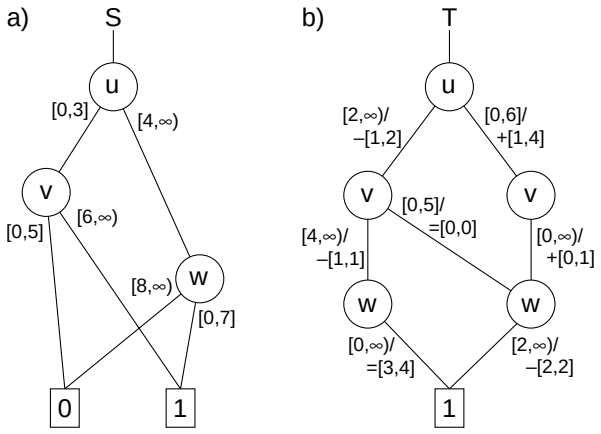

Figure 1: Interval decision diagram and interval mapping diagram.

Each path represents a possible state transition which is executable if all edges along the path are enabled. The combination of predicate and action interval parameterizes the mapping function and completely defines its behavior.

2.3 Image Computation

In [5], an efficient algorithm is described to perform forward or backward image computation using an IDD S for the state set and a IMD T for the transition relation, resulting in an IDD S' representing the image state set. This algorithm may be used to perform reachability analysis or symbolic model checking by fixpoint computation. IMDs are dedicated to image computation especially for Petri nets as the state distance (action interval) combined with the respective firing condition (predicate interval) may be stored more efficiently than many state pairs.

3 Symbolic Model Checking

Symbolic model checking allows for the verification of certain temporal properties of state transition systems, where the explicit construction of an entire reachability graph is avoided by implicitly depicting it using symbolic representations. Often, the propositional branching-time temporal logic CTL (*Computation Tree Logic*) is used. Petri net properties to be checked may be specified as CTL formulae and verified with well-known techniques. A few questions out of the wide variety of system properties to be checked using CTL are, e.g., "may places p_1 and p_2 simultaneously contain no tokens?", "can transitions t_1 and t_2 be fired concurrently?", or "must all firing sequences eventually result in marking M_1?". Additionally, specialized algorithms exist for the verification of many common Petri net properties [3] and are straightforward adaptable to interval diagram techniques, e.g., for deadlock freeness and diverse levels of liveness, or boundedness, persistence, and home state property.

3.1 Experimental Results

Several different system models based on Petri nets have been investigated which led to promising results. The set of reachable states has been calculated by a series of image computations. Some results for different initial configurations are presented, comparing IDDs and IMDs to BDDs. Our investigations yielded promising results concerning the number of nodes and edges as well as the computation time.

Figure 2 shows the size of the diagram representing the set of reachable states of a model of a flexible manufacturing system with automated guided vehicle for increasing initial configurations m.

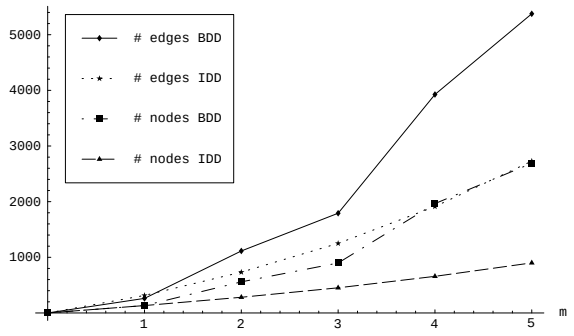

Figure 2: Size of state set diagram.

In Figure 3, the computation time T to determine the set of reachable states is shown depending on the initial configuration m. For both criteria, IDDs and IMDs turn out to be superior to the conventional approach using BDDs.

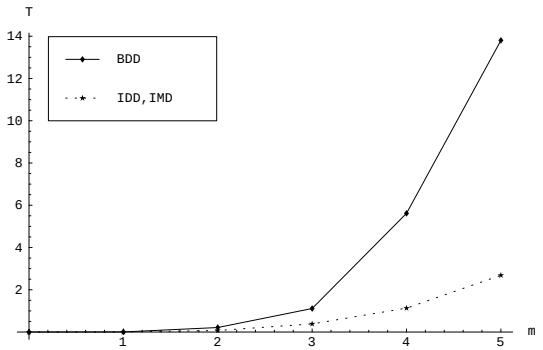

Figure 3: Computation time in 10^3 seconds.

References

[1] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):667–691, August 1986.

[2] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[3] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia. Petri net analysis using boolean manipulation. In *15th International Conference on Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 416–435. Springer-Verlag, 1994.

[4] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton. Algorithms for discrete function manipulation. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, 1990.

[5] Karsten Strehl and Lothar Thiele. Symbolic model checking of process networks using interval diagram techniques. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD-98)*, pages 686–692, 1998.