Efficient Computation of Recurrence Diameters*

Daniel Kroening and Ofer Strichman

Computer Science, Carnegie Mellon University, Pittsburgh, PA
{kroening|ofers}@cs.cmu.edu

Abstract. SAT based Bounded Model Checking (BMC) is an efficient method for detecting logical errors in finite-state transition systems. Given a transition system, an LTL property, and a user defined bound k, a bounded model checker generates a propositional formula that is satisfiable if and only if a counterexample to the property of length up to k exists. Standard SAT checkers can be used to check this formula. BMC is complete if k is larger than some pre-computed threshold. It is still unknown how to compute this threshold for general properties. We show that the longest initialized loop-free path in the state graph, also known as the *recurrence diameter*, is sufficient for $\mathbf{F}p$ properties. The recurrence diameter is also a known over-approximation for the threshold of simple safety properties ($\mathbf{G}p$). We discuss various techniques to compute the recurrence diameter efficiently and provide experimental results that demonstrate the benefits of using the new approach.

1 Introduction

SAT-based Bounded Model Checking (BMC)[1] was introduced several years ago as a complementary technique for the more traditional BDD-based symbolic model checking [2]. The basic idea of BMC is to search for a counterexample in traces whose length is bounded by some integer k. If no bug is found then the bound k is increased until either a bug is found, the problem becomes intractable, or some pre-computed *Completeness Threshold* $\mathcal{CT}$ is reached[1]. If the completeness threshold is reached without finding a bug, it is implied that the property holds in the given model. The BMC problem can be efficiently reduced to a propositional satisfiability problem, and can therefore be solved by standard SAT methods.

Knowing the completeness threshold $\mathcal{CT}$ is essential for making BMC complete. Without it, there is no way of knowing whether the property holds or rather the bound is not sufficiently high. The value of $\mathcal{CT}$ depends on the model M, the temporal property p,

* This research was sponsored by the Semiconductor Research Corporation (SRC) under contract no. 99-TJ-684, the National Science Foundation (NSF) under grant no. CCR-9803774, the Office of Naval Research (ONR), and the Naval Research Laboratory (NRL) under contract no. N00014-01-1-0796. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of SRC, NSF, ONR, NRL, the U.S. government or any other entity.

[1] In all previous publications on this matter, the term 'diameter' was used to denote what we refer to as the completeness threshold. Since the term diameter has a specific meaning in graph theory that coincides with the completeness threshold only for some properties (as we explain later), it is unsuitable for describing the general case.

L. Zuck et al. (Eds.): VMCAI 2003, LNCS 2575, pp. 298–309, 2003.

and the exact translation scheme used for obtaining the SAT instances. In this paper we refer to the original LTL translation of Biere et al.[3]. For this translation and $\mathbf{G}p$ (where p is a non-temporal expression) formulas, they show that $\mathcal{CT}$ is equal to d, the longest shortest-path from the initial state (the 'diameter', or, because the path has to start from an initial state, the 'radius' of the model [4]). For a given model and a number k, finding whether $d > k$ can be done by solving a Quantified Boolean Formula (QBF). Thus, finding d amounts to solving a sequence of QBF formulas with increasing k, which is computationally very expansive.

There are several approaches for making this process more efficient. Baumgartner et al. [5] perform an analysis of the circuit structure on the netlist level and identify frequently occurring components with pre-known diameters. The overall diameter of the circuit is then defined recursively over its individual components. With this analysis they found a large number of cases were the diameter can be computed in a short amount of time, and, more importantly, the diameter itself is relatively very small (less than 20 in many cases). Mneimneh and Sakallh[6] suggest a method for simplifying the QBF formulas: while normally one has to check whether each state that is reachable in k steps can be reached sooner, they observe that it is sufficient to check whether it can be reached within $k-1$ steps, if the initial state has a self loop (such a loop can always be added without affecting the value of the diameter). In [3], Biere et al. suggest to over-approximate the diameter with the *recurrence diameter* r, which is the longest loop-free path between two states. Since every shortest-path is a loop-free path, then obviously $d \leq r$. The recurrence diameter can be computed by solving a series of SAT instances, rather than QBF instances, and it is therefore typically easier to compute. The recurrence diameter is important not only as an over-approximation of d, but also because r characterizes the completeness threshold for $\mathbf{F}p$ properties.

In section 2 we suggest several refinements to the original definition of r that result in shorter paths that are still larger or equal than $\mathcal{CT}$. In section 4.1 we present an efficient way of computing r, based on *sorting networks*. While the currently known technique for computing it requires solving a SAT instance of size quadratic in k, our method requires solving a formula of size $O(k \log k)$. With these improvements we were able to compute the recurrence diameter in several cases that were otherwise impossible to compute with the original method. We provide experimental results to quantify the performance impact of our method.

2 The Completeness Threshold for Simple Properties

We begin with some notation and definitions:

1. A finite transition system $M = \langle \mathcal{S}, I, T \rangle$ is defined by a finite set of states $\mathcal{S}$, an initial-state predicate I, and a transition relation T. States are defined by valuations of the state variables st and input variables in. For simplicity we assume that these are the only type of variables in M.
2. Given a finite transition system M and an LTL property p, we write $M \models p$ if p holds for M, and $M \models_k p$ if p holds for M up to cycle k. Each 'cycle' corresponds to a single application of the transition function, or, in other words, an exploration of the immediate successor states.

3. The predicate $R(t,n)$, $t \in S, n \in \mathbb{N}$, is true if and only if t is reachable within n steps from an initial state:

$$R(t,n) \stackrel{\text{def}}{=} \exists s_0 \ldots s_n.\ I(s_0) \wedge \bigwedge_{i=0}^{n-1} T(s_i, s_{i+1}) \wedge s_n = t$$

4. The predicate $R_o(t,n)$ denotes that a state t is reachable in n steps from an initial state and cannot be reached via a shorter path.

$$R_o(t,n) \stackrel{\text{def}}{=} R(t,n) \wedge \nexists m, m < n.\ R(t,m)$$

5. The predicate $R(t)$, $t \in S$, is true if and only if t is reachable from an initial state:

$$R(t) \stackrel{\text{def}}{=} \exists n.\ R(t,n)$$

We use this terminology in our definition of the completeness threshold $\mathcal{CT}$ with respect to a model M and an LTL formula φ:

Definition 1 (Completeness threshold). *The* completeness threshold *of a finite transition system M and a property φ, denoted by $\mathcal{CT}(M,\varphi)$ is the minimal number such that if φ holds up to cycle $\mathcal{CT}(M,\varphi)$, it holds globally. Formally:*

$$\mathcal{CT}(M,\varphi) \stackrel{def}{=} \min\{i \mid M \models_i \varphi \rightarrow M \models \varphi\} \tag{1}$$

Note that $\mathcal{CT}(M,\varphi)$ can be arbitrarily large, because φ can specify an arbitrarily long path. The most widely used properties in practice are unnested formulas like $\mathbf{G}p$ and $\mathbf{F}p$, where p is a non-temporal expression (most safety properties can be reduced to $\mathbf{G}p$). This is the only type of formula for which a method for computing $\mathcal{CT}(M,\varphi)$ is currently known. The completeness threshold for $\mathbf{G}p$ formulas is the *reachability diameter*[2]:

Definition 2 (Reachability Diameter). *The reachability diameter $rd(M)$ is the minimal number of steps required for reaching all reachable states:*

$$rd(M) \stackrel{def}{=} \min\{i \mid \forall t, t \in S.\ \exists j, j \leq i.\ R(t) \rightarrow R(t,j)\} \tag{2}$$

It is easy to see why the reachability diameter is sufficient for $\mathbf{G}p$ formulas. A counterexample to $\mathbf{G}p$ is a path to a state that contradicts p. Since all states can be reached through paths of length $rd(M)$ or less, checking paths whose length is bounded by $rd(M)$ is sufficient for finding all reachable states that contradict p. On the other hand, the reachability diameter is not sufficient for finding all counterexamples to $\mathbf{F}p$ formulas. A counterexample for such a formula is a path ending in a back-loop, where all the states on the path satisfy $\neg p$. Figure 2 demonstrates such a path. While the top path is the shortest counterexample (of length four), all states are reachable through paths of length two or less (i.e. $rd(M) = 2$).
Thus, in order to find a counterexample to $\mathbf{F}p$ it is not sufficient to check all paths of length smaller or equal to $rd(M)$. For this type of formulas we need to look for the longest loop-free path from an initial state, or, in other words, for the reachability recurrence diameter [1], which is defined as follows:

[2] This definition is equivalent to the 'radius' in [4].

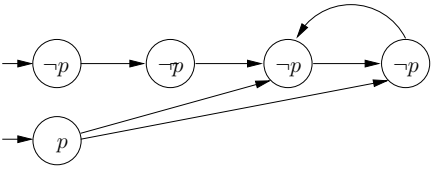
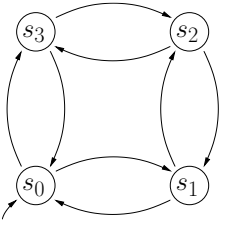


Fig. 1. While the reachability diameter $rd(M)$ in this model is equal to two, the shortest counterexample to the property $\mathbf{F}p$ is of length four.

Definition 3 (Reachability Recurrence Diameter). *The Reachability Recurrence Diameter with respect to a model M, denoted by $rrd(M)$, is the longest loop-free path in M starting from an initial state:*

$$rrd(M) \stackrel{def}{=} max\{i|\ \exists s_0 \ldots s_i.\ I(s_0) \wedge \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k\} \quad (3)$$

The difference between $rd(M)$ and $rrd(M)$ is demonstrated in the drawing below. It shows a structure in which the reachability diameter $rd(M)$ is three, because all states are reachable from the initial state through paths of length three, while $rrd(M)$, the maximal loop-free path starting from an initial state, is equal to four.



The recurrence diameter was originally suggested in [1] as an over-approximation to the completeness threshold of $\mathbf{G}p$ formulas. The reachability diameter, which is smaller but sufficient for this type of formulas, is too hard to compute because it is a QBF, as was mentioned in the introduction. Restricting the first state to an initial state, as definition 3 requires, is still conservative with respect to $\mathbf{G}p$, because clearly any shortest path to a $\neg p$ state is a loop-free path starting from an initial state. Thus, to summarize this section, the reachability recurrence diameter $rrd(M)$ is useful for both computing the completeness threshold for $\mathbf{F}p$ and for over-approximating it in the case of $\mathbf{G}p$ formulas.

From now on we will only refer to formulas of the form $\mathbf{G}p$ and $\mathbf{F}p$, and therefore, for conciseness, omit the word 'reachability'.

3 A Shorter Recurrence Diameter

Definitions 1–3 refer to states without explicitly saying by which variables these states are defined. Normally a state is defined to be the product of the valuation of all variables.

In subsections 3.1 and 3.2 we show that by ignoring the input variables and variables that are not in the bounded cone of influence of the property, we can find shorter thresholds. Although we prove the correctness of these improvements with respect to the recurrence diameter, they apply to the reachability diameter as well. We will use the following additional notation:

1. The value of a variable v in a state $s \in \mathcal{S}$ is denoted by $s(v)$.
2. Let $\hat{s}$ be the set of variables that define the state s. Given a set of variables v, s^v is the projection of s to variables that appear both in v and $\hat{s}$, i.e., if $\{v_1 \dots v_p\} \subseteq v \cap \hat{s}$, $s^v = (s(v_1), ..., s(v_p))$.
3. If v is some subset of $\hat{s}$, then we denote the recurrence diameter for reachability (as defined in equation 3) by $rrd(M, v)$, when the states are restricted to s^v. Thus, $rrd(M)$ of equation 3 can be written as $rrd(M, \hat{s})$.
4. We use superscripts for cycle numbers and subscripts for variable indices. For example, $in_0^k \dots in_n^k$ can represent $n + 1$ input variables in cycle k.

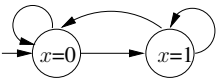
3.1 Ignoring the Inputs

In this section we show that there is no need to consider input variables when comparing states, and, consequently, that it is possible to find smaller completeness thresholds. In other words, when computing the recurrence diameter, we will treat all states that are equal modulo their input variables as a single state. Consider, for example, the transition system in Fig. 2. The system has one state variable x and n inputs. If we consider the inputs, from the right state ($x = 1$) we can progress to a $2^n - 1$ long loop-free path while maintaining $x = 1$ (all possible assignments to the input variables, except the one that assigns them all 0). Thus, the fact that we do not distinguish between these states shortens the recurrence diameter from $O(2^n)$ to 2.

```
input var i_0 ... i_n: boolean;
var x: boolean;

init(x) := 0;
next(x) := i_0 ∨ ... ∨ i_n;
```

(a)



(b)

Fig. 2. A transition system (a) and its corresponding Kripke structure (b). When considering inputs, the calculated recurrence diameter of this system is $O(2^n)$ long.

Definition 4 (State variables recurrence diameter). *The state variables recurrence diameter with respect to a model M, denoted by $rrd(M, st)$, is the longest loop-free path in M starting from an initial state, where two states are considered as equal if their state variables have the same value:*

$$rrd(M, st) \stackrel{def}{=} max\{i |\ \exists s_0 \dots s_i.\ I(s_0) \wedge \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j^{st} \neq s_k^{st}\} \quad (4)$$

Note that other than the way that states are compared to one another, $rrd(M, st)$ is defined exactly as the recurrence diameter $rrd(M)$.

It is not obvious that inputs can be ignored when computing $\mathcal{CT}$, because the property may refer to inputs. Thus, two states that can only be distinguished by their input variables are not equivalent with respect to a counterexample trace. The following lemma states that nevertheless, ignoring the inputs is safe:

Lemma 1. *If there exists a counterexample to* $\mathbf{F}p$ *of length greater than* $rrd(M, st)$ *(where the length is defined by the number of distinct states in the path), then there exists another counterexample to* $\mathbf{F}p$ *of length smaller or equal to* $rrd(M, st)$.

Proof. Assume that there exists a path π ending with a back-loop s.t. $|\pi| > rrd(M, st)$ and every state in π satisfies $\neg p$. Further, assume that π is the shortest path with this property. π must have at least two states that are equal in all state bits. Let s_l, s_r be such two states, where s_l is the state on the left (closer to the initial state). Given an assignment α that satisfies this path, we construct a new assignment α' as follows. First, α' is equal to α in all states to the left of s_l, and equal to α in s_l itself with respect to the state variables. Second, α' assigns the inputs of s_l the same values that α assigns to the inputs of s_r. Now α' satisfies $\alpha'(s_l) = \alpha(s_r)$, and we can therefore proceed from s_l to s_{r+1}. Thus, the third stage of the construction shifts the assignment of α to states $s_{r+1} \ldots s_{|\pi|}$ to states $s_{l+1} \ldots s_{|\pi|-(r-l)}$. Now α' corresponds to a path shorter than π that satisfies $\neg p$ in all of its states. The existence of such a path contradicts our assumption. □

Although Lemma 1 refers explicitly to $\mathbf{F}p$ formulas, it is clear that inputs can be ignored also when computing an over-approximation to $\mathbf{G}p$ formulas. Baumgartner et al. [5] ignore the inputs as well when they compute the reachability diameter.

3.2 Ignoring Variables That Do Not Affect the Property

It was observed in [5] that the reachability diameter should be computed while considering only the cone of influence of the property. We now show that this observation can be extended to the *bounded* cone of influence. Bounded cone of influence (BCOI) [7] is a reduction method that operates in two stages. First, it identifies the variables that affect the property at (or up to) the cycle bounding the search; second, it simplifies the formula by assuming an arbitrary value for the rest of the variables. For example, in the following transition system, the value of the state variables x_1 and x_2 affect the property with a delay of one and two cycles, respectively.

```
var x0, x1, x2: boolean;
init(x0) = 0; init(x1) = 0; init(x2) = 0;
next(x2) := !x2;
next(x1) := x2
next(x0) := x1
SPEC AG(!x0)
```

Consequently, when checking the property at cycle k, we can ignore the value of x_2 in cycles k and $k-1$, and ignore the value of x_1 in cycle k. The same idea can be used for finding a smaller completeness threshold.

Let $B^k(i)$, $i \leq k$ be the set of variables such that their value in cycle i affect the property at cycle k. Since $B^k(i)$ can include different variables for each cycle i, we need to redefine 'comparison' between two states. We define two states $s_i, s_j, j > i$ to be equal if and only if they are equal in all variables in $B^k(j)$, i.e., $\forall v \in B^k(j), s_i(v) = s_j(v)$. Given this definition, we now redefine the recurrence diameter:

$$rrdb(M) \stackrel{\text{def}}{=} max\{i |\ \exists s_0 \ldots s_i.\ I(s_0) \wedge \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j^{B^i(j)} \neq s_k^{B^i(k)}\} \tag{5}$$

To combine the improvement of section 3.1 with this definition, we can restrict the comparison further to state variables only (that is, each state s_j is restricted to $B^i(j) \cap st$), and denote the result of this restriction by $rrdb(M, st)$. For simplicity, however, we assume from here on that all variables in the bounded cone of influence are state variables.

Lemma 2. *If there exists a state t_1 that contradicts the property such that $R_o(t_1, n_1)$, $n_1 > rrdb(M)$, then there exists a state t_2 that contradicts the property as well and $R_o(t_2, n_2)$, $n_2 \leq rrdb(M)$.*

Proof. Assume the contrary. By Equation 5 and Lemma 1, this implies that in the path to t_1 there exists two states, say s_i and $s_j, j > i$, that are equal in their $B^{n_1}(j)$ variables but different in at least one other variable. A state equal to s_{j+1} in all $B^{n_1}(j+1)$ variables can be reached in one step from s_i. Let s_{i+1} denote this state. Then, a state equal to s_{j+2} in all $B^{n_1}(j+2)$ can be reached from s_{i+1}, and so forth. Thus, a sequence of states which is equivalent to the path from s_j to t_1 in all variables relevant to the property can be reached in one step from s_i, which means that we can reach a state contradicting the property in less than n_1 steps. This contradicts our assumption. □

As an example of where the restriction to BCOI variables can shorten the computed recurrence diameter, consider the diagram in Fig. 3, and a property $\mathbf{G}p$. The states in this diagram are defined by two state variables, say v_1 and v_2. The variables that affect p in cycle 4 (according to a BCOI analysis) are underlined. The states s_1 and s_2 are considered different if we consider all variables, but equal if we consider only variables that affect the property. Taking the second option implies that $s_0 \rightarrow s_1 \rightarrow s_3$ is the longest loop free path, and hence the threshold is reduced from four to three. Note that the path from s_2 to a state that contradicts p implies that there must exist such a state that can be reached in one step from s_1, possibly s_3 itself. This is because the value of p depends only on v_1, and we know that this variable has the same value in s_1 and s_2.

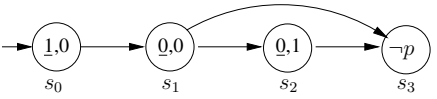


Fig. 3. Two possible paths, where the variables affecting the property p are underlined.

4 Finding the Recurrence Diameter with Sorting Networks

4.1 Sorting Networks for Loop Detection

The currently used technique [1] for computing the recurrence diameter (either $rrd(M)$ or $rrd(M, st)$) compares all pairs of states, as implied by Equations 3 and 4. The size of the resulting formula is therefore quadratic in the length of the path k.

We propose the following alternative: first, we generate an equation that represents the same set of states but in a sorted order; second, we compare the neighbors in the sorted sequence. Since we have to generate the equation without any actual knowledge of the states, the sequence of comparisons performed must be the same for all possible states. This is known as the Bose-Nelson sorting problem. A circuit that solves this problem is called a *sorting network* (see Knuth [8] for a survey).

Ajtai, Kolmós, Szemerédi [9] show that sorting networks for n inputs can be built with size $O(n \log n)$. However, there is a very high constant (several thousands) hidden in this complexity result that makes it impractical for our purpose. We therefore use a variant of a *bitonic sorting network* as described by Batcher [10], which has an asymptotic size of $O(n \log^2 n)$. Bitonic sorting networks have a recursive structure (see Fig. 4). The inputs are split into two parts that are sorted independently and then merged. This means that the sorting blocks can be replaced by any other sorting network. While for an arbitrary input size the minimal size of the sorting network is unknown, for small numbers there are known optimal or near optimal solutions. For these cases, we replace the bitonic sorting network with these known small sorting networks. Figure 5 shows a simple sorting network for three input states.

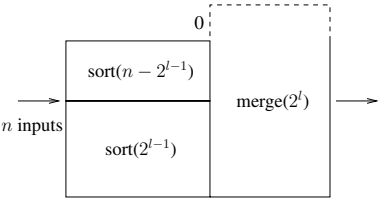


Fig. 4. Block diagram for a bitonic sorting network with n inputs and $l = \lceil \log_2 n \rceil$. If n is not a power of two, the smallest element (denoted by 0) is used for merging.

Let $s_0 \ldots s_{n-1}$ denote the n states of the path. Using the sorting network, we obtain an ordered permutation of these states $s'_0 \geq s'_1 \ldots \geq s'_{n-1}$. It is obvious that a sequence

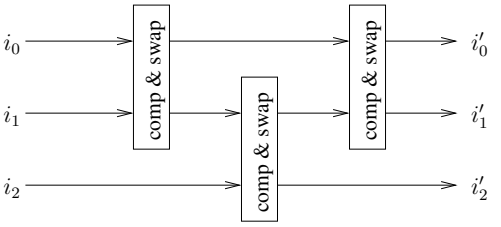


Fig. 5. Sorting network for three inputs. $i_0..i_2$ are the input unsorted states, and $i'_0..i'_2$ are the output sorted states.

of states contains two equal states if and only if its corresponding sorted sequence contains two equal neighboring states, or, formally:

$$\exists i : s'_i = s'_{i+1} \iff \exists l, j : l \neq j \wedge s_l = s_j \tag{6}$$

Thus, we now only have to compare all neighbors in this sequence. This can be done with $n - 1$ comparisons.

4.2 Ordering and Swapping

All sorting networks require a *compare* and *swap* operation. Two elements a and b are compared and, if $a > b$, are swapped. We implement the ordering operator by computing the last carry bit of the sum $a + (-b)$. Let a denote a bit vector of length β, and let a_i, $0 \leq i < \beta$ denote the i-th component of a. Let $\overline{b}$ denote the inverted vector b. Since $\overline{b} + 1$ is equal to $-b$, we can compute $a + (-b)$ by computing $a + \overline{b} + 1$. The first carry bit c_0 of this sum is:

$$c_0 := a_0 \vee \overline{b_0} \tag{7}$$

The i-th carry bit of this sum with $i \geq 1$ is:

$$c_i := (a_i \wedge \overline{b_i}) \vee (a_i \wedge c_{i-1}) \vee (\overline{b_i} \wedge c_{i-1}) \tag{8}$$

The value of the last carry bit $c_{\beta-1}$ determines whether we swap a and b. Let b' denote the new value (after swapping) of b. The equation for a' follows the same pattern.

$$b'_i = a_i \wedge c_{\beta-1} \vee b_i \wedge \overline{c_{\beta-1}} \tag{9}$$

Equation 7 is transformed into CNF using 1 new literal and 3 clauses, equation 8 requires 1 new literal and 6 clauses, and equation 9 requires 1 new literal and 4 clauses. The swapping has to be done for both a and b. Thus, the total cost of one compare/swap operation with β bits is 4β literals and 17β clauses.

5 Experimental Results

We experimented with several circuits from the ISCAS'89 benchmark netlists. First, we checked the influence of considering only state variables when comparing states. We were able to compute $rrd(M, st)$ (the reachability recurrence diameter of state variables) of 6 out of the 35 instances in the benchmark (see top 6 circuits in figure 6). On the other hand we could not compute $rrd(M)$ for any of them using a one hour time limit[3].

We also tried several other circuits. The recurrence diameter for the arithmetic circuits div8 and mult8, and the serial bus controller circuit IIC is computed easily (see the bottom part of figure 6).

Circuit	$rrd(M, st)$	time ($rrd(M, st)$) (SAT)	time($rrd(M, st) + 1$) (UNSAT)	$rrd(M)$	time($rrd(M)$) (SAT)
s27	5	$<$ 1s.	$<$ 1s.	?	*
s386	11	$<$ 1s.	$<$ 1s.	?	*
s510	46	$<$ 1s.	$<$ 1s.	?	*
s832	17	$<$ 1s.	1 s.	?	*
s820	17	$<$ 1s.	$<$ 1s.	?	*
s208.1	255	2.58 s.	3.35 s.	?	*
div8	11	$<$ 1s.	2.28 s	?	*
mult8	9	$<$ 1s.	$<$ 1s	?	*
IIC	31	$<$ 1s.	$<$ 1s	?	*

Fig. 6. Ignoring the inputs while comparing states shortens the diameter, and hence makes it easier to compute.

To check the efficiency of sorting networks versus the previously known all-pair method, we generated the CNF files of the s27 benchmark circuit with both methods for an increasing bound k. Figure 7 depicts our results. It shows that for k larger than about 50, the CNF file that is generated with sorting networks is smaller, both with respect to the number of variables, and the the number of clauses (the break-even point depends on the number of state-bits. This point is smaller if there are less state bits). Although the size of the CNF does not directly predict the time it takes to solve it, it does give some estimation of how hard the problem is, especially when the difference in size is significant.

6 Conclusions and Directions for Future Work

Finding the recurrence diameter of designs with respect to a given property is important for achieving completeness in Bounded Model Checking. We have presented several techniques for making this calculation easier than previously known techniques. Since

[3] We reached $k = 300$ with most of these files, but the recurrence diameter is apparently much higher.

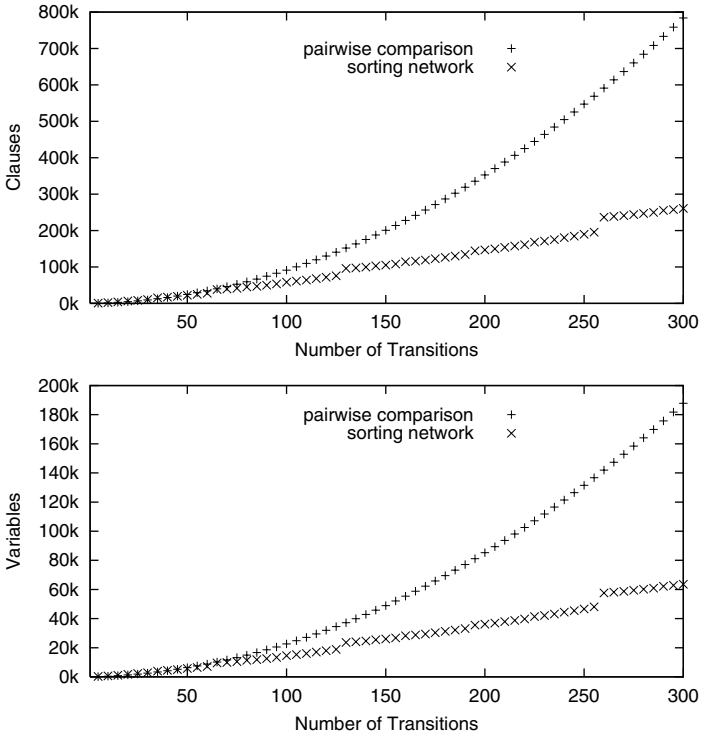


Fig. 7. Comparison of the CNF size (number of literals and clauses) with pairwise comparison and with sorting network

the recurrence diameter is typically very high (it can be exponential in the number of state variables), in many cases it is still impractical to find, let alone perform bounded model checking with such a high bound. Thus, finding more efficient ways to calculate the recurrence diameter is still an important research topic. The question of how to find the completeness threshold for a general LTL property is still open.

References

1. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, 1999.
2. E.M.Clarke, O.Grumberg, and D.Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.
3. A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, , and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Design Automation Conference (DAC'99)*, 1999.
4. A. Biere, C. Artho, and V. Schuppan. Liveness checking as safety checking. In *FMICS workshop 2002*, 2002.
5. J. Baumgartner, A. Kuehlmann, and J. Abraham. Property checking via structural analysis. In *Proc.* 14^{th} *Intl. Conference on Computer Aided Verification (CAV'02)*, 2002.

6. M. Mneimneh and K. Sakallah. SAT-based sequential depth computation. In *Constraints in formal verification workshop*, Cornell University, Ithaca, New York, Sep 2002.
7. A. Biere, E. Clarke, R. Raimi, and Y. Zhu. Verifying safety properties of a PowerPC^{TM} microprocessor using symbolic model checking without bdds. In N. Halbwachs and D. Peled, editors, *Proc.* 11^{th} *Intl. Conference on Computer Aided Verification (CAV'99)*, LNCS. Springer-Verlag, 1999.
8. D.E. Knuth. *The Art of Computer Programming, volume 3: Sorting and Searching*. Addison Wesley, 1973.
9. M. Ajtai, J. Komlós, and S. Szemerédi. An $O(N \log N)$ sorting network. In *Proceedings of the* 25^{th} *ACM Symposium on Theory of Computing*, pages 1–9, 1983.
10. K.E. Batcher. Sorting networks and their applications. In *Proc. AFIPS Spring Joint Comput. Conf.*, volume 32, pages 307–314, 1968.