Automatic Generation of Network Invariants for the Verification of Iterative Sequential Systems[*]

June-Kyung Rho Fabio Somenzi

Department of Electrical and Computer Engineering
University of Colorado, Boulder 80309

Abstract. In this paper we present new results on the verification of iterative sequential systems. We address bilateral interconnections and circular arrangements of cells, and extend our previous treatment [13] of unilateral systems. Our approach is based on the new definition of boundedness, given in terms of regularity of the union of an infinite number of languages. The definition of boundedness provides sufficient conditions for the equivalence of iterative systems to be decidable. It also provides network invariants for inductive proofs. This new framework allows the derivation of some previously known results as well as the new ones presented here.

1 Introduction

In the verification of finite state systems, one of the most difficult problems is how to avoid the so-called state explosion problem. In general, compositional minimization techniques [4, 7, 12] can be used to reduce the size of the problem or the search space. When a finite system is composed of an arbitrary number of instances of the same basic cell, it is interesting to decide whether the verification can be reduced to a finite problem regardless of the size of the system. It is well known that this problem is undecidable [9]. We present new techniques to check whether there exists an invariant and, if so, to generate it automatically. A finite state system can be characterized by the set of strings it generates or accepts. We show that the verification of a system with an arbitrary number of cells can be done in a finite amount of time if the union of the sets of strings generated or accepted by any cell can be represented by a finite state system. The resulting finite state system is a network invariant.

Inductive proofs have been proposed for various problems concerning the properties of finite sets. However, it is difficult to automate such proofs. Kurshan and McMillan [11] discuss a structural induction procedure that requires an invariant produced manually. Wolper and Lovinfosse [17] have similar results based on process theory. Results also have been obtained in the context of model-checking [16, 1, 14].

[*] This work was supported in part by NSF/DARPA grant MIP-9115432 and by SRC contract 91-DJ-206.

In this paper, we mainly concentrate on the problem of verifying sequential iterative systems. The relation between binary decision diagrams (BDD's) and finite automata is investigated and exploited for that purpose. In [13], we studied the properties of unilateral iterative systems, based on the transpose of the iterative system and on language convergence. We extend our treatment to bidirectional and circular systems and provide new results for unilateral systems.

Even if we are interested in the properties of finite iterative networks, we may want to consider iterative networks with a countably infinite number of cells to guarantee that the properties hold regardless of the number of cells. Such properties, network invariants, are interesting to us only if they can be represented by another finite state system. For instance, the equivalence of two iterative networks can be decided if the set of strings that can be generated by any cell of the networks can be represented by a finite automaton.

The language of an iterative system is defined as the union of the countably infinite languages that are generated or accepted by any cell of the system. The iterative system is bounded if the language of the system is regular. The boundedness conditions identify classes of decidable equivalence problems for iterative systems.

After the preliminaries of Section 2, the relation between BDD's and finite automata is examined in Section 3. In Section 4, we present the concept of language boundedness and its application to unilateral systems, followed by the results for bilateral and circular systems. Section 5 describes how to apply the proposed techniques to an example. Finally, we present conclusions in Section 6.

2 Preliminaries

Binary Decision Diagrams (BDD's) [2] provide a convenient representation for logic functions. A BDD is a directed acyclic graph representing a multiple-output logic function f. In what follows, a finite state machine (FSM) is a 6-tuple $\langle I, S, O, \delta, \lambda, S^0 \rangle$, where I is an input alphabet, S is a set of states, O is a output alphabet, δ and λ are the next-state and output functions, respectively, and S^0 is the set of initial states. A finite automaton is a 5-tuple $\langle K, \Sigma, \Delta, F, S^0 \rangle$, where K is a finite set of states, Σ is an alphabet, Δ is a finite subset of $K \times 2^{\Sigma^*} \times K$, $S^0 \subset K$ is the set of initial states, and $F \subset K$ is the set of final or accepting states. An automaton is deterministic if S^0 is a singleton, Δ is right-unique, and for $k_1, k_2 \in K$, $P_1, P_2 \in 2^{\Sigma^*}$, $s_1 \in P_1$, $s_2 \in P_2$, and s_2 a prefix of s_1, $(k_1, P_1, k_2) \in \Delta$ implies that $\neg\exists k_3$ such that $(k_1, P_2, k_3) \in \Delta$. Defining Δ over finite strings instead of over elements of the alphabet does not affect finiteness, and provides more concise representations.

A state σ is *reachable* if there exist a sequence of states $s_1, \ldots, s_k$ and a sequence of strings $i_0, \ldots, i_k$ (possibly repeated), such that $\delta(s^0, i_0) = s_1$, $\delta(s_j, i_j) = s_{j+1}$, $j = 1, \ldots, k-1$, and $\delta(s_k, i_k) = \sigma$. The set of reachable states of a finite state machine or an automaton is of interest in verification and efficient algorithms have been developed to find it [5, 15, 8, 3].

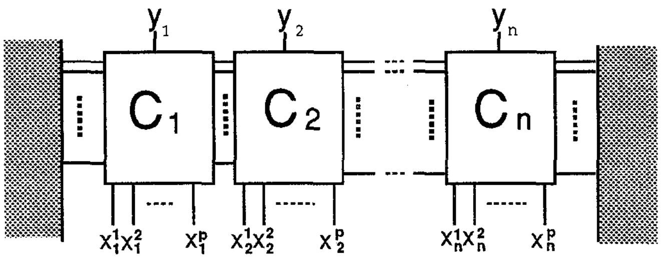

Fig. 1. Iterative Network.

Let $\Phi_F(\cdot)$ be the language transfer function of an FSM F. The language produced by an FSM F when language U is applied to its inputs, $\Phi_F(U)$, is the set of output strings produced by the machine when started in any state of S^0. When U is a regular language, $\Phi_F(U)$ is also regular. The language transfer function of an FSM is monotonic, i.e., for $U, V \subseteq \Sigma^*$, if $U \subseteq V$, then $\Phi_F(U) \subseteq \Phi_F(V)$. The language accepted by a finite automaton is the set of all the input strings that drive the automaton from any initial state to one of the accepting states.

An *iterative network* is a circuit obtained by interconnecting multiple instances of the same basic *cell*, as in Figure 1. The signals entering the network from the left and the right are called *boundary conditions*. The inputs of a cell coming from an adjacent cell are called *communication inputs* and the externally controllable inputs are called *local inputs*. An iterative network is *unilateral* if the *communication inputs* carry signals in only one direction; otherwise it is *bilateral*. If the cell is a combinational circuit, then the iterative network is said to be combinational. An *iterative system* is the class of all iterative networks with the same cell and boundary conditions and different numbers of cells. The symbol ω denotes the first infinite ordinal.

Sequential iterative networks correspond to two-dimensional combinational arrays. The transpose of a sequential iterative network is another sequential iterative network that is obtained by interchanging the time and space domains of the original iterative network [13]. More details on iterative networks can be found in [9, 10].

3 Finite Automata and BDD's

It is well known that a BDD can be regarded as a device computing a given function. The ordering of the variables of a BDD imposes the same constraints as the time constraints of finite automata. The *reduce* operation [2] on BDD's can be compared to the state minimization of finite automata. We show formally that BDD's correspond to a special class of finite automata and use this result for various verification problems. Let *M(n)* be the class of deterministic finite automata such that for every $M \in M(n)$, M is state-minimal and accepts the language $L \cdot (1+0)^*$, where $L \subseteq (1+0)^n$, and for $P \in 2^{\Sigma^*}$, $(k_1, P, k_2) \in \Delta$ only if $P = \sigma_1 \cdot (1+0)^{k-1}, \sigma_1 \in \{0,1\}$.

Lemma 1. *There exists a bijection between BDD's for $\{f : B^n \rightarrow B\}$ and $M(n)$.*

Proof. The number of possible functions in $\{f : B^n \rightarrow B\}$ is 2^{2^n}. So there are 2^{2^n} different BDD's for f in the set. Similarly, there are 2^n strings in $L = (1+0)^n$, and the number of the subsets of $(1+0)^n$ is also 2^{2^n}. State-minimized deterministic finite automaton are unique; hence, there are also 2^{2^n} unique finite automata for 2^L. Since the cardinality of the two finite sets is the same, we can define a bijection between them. □

One meaningful bijection maps the BDD for $f : B^n \rightarrow B$ into the finite automaton M which accepts the set of binary strings that satisfy f. Let ϕ be the bijection such that $\phi(D) = M$, where D is the BDD for $f : B^n \rightarrow B$ and $M \in M(n)$ is the finite automaton that accepts the language $\{s \cdot (1+0)^* : s \in (1+0)^n, f(s) = 1\}$.

Corollary 2. *The bijection ϕ and its inverse ϕ^{-1} induce a graph isomorphism between the BDD D and the finite automaton M.*

Figure 2 shows a BDD and a corresponding finite automaton. It is easily seen that the transitions, unaccepting states except sink states, the accepting sink state and the unaccepting sink state of M are isomorphic to the edges, internal nodes, constant '1' node, and the constant '0' node of the BDD, respectively. M is a bit serial implementation of the function f, which is optimal in terms of the number of states. Whether the BDD uses complement arcs is not essential. If complement arcs are used in the BDD, we can define a class of extended finite automata with complement attributes on the transitions. We show a condition for a class of functions to have linear-size BDD's in n, the number of variables, based on Lemma 1 and Corollary 2.

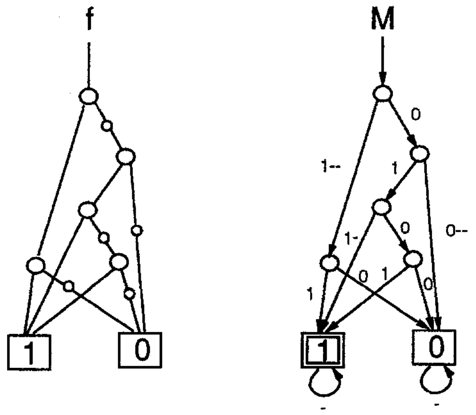

Fig. 2. BDD and Finite Automaton.

Lemma 3. *Let $\{f_i : i > 0\}$ be a class of functions, where $f_i : B^{ki+m} \to B$ for some k and m. Let $L = \bigcup_{i=1}^{\omega} L_i$ be the language such that $L_i = \{s \in (1{+}0)^{ki+m} : f_i(s) = 1\}$. The size of the BDD of f_i grows linearly in i if L is regular.*

Proof. (sketch) Let M be the state-minimal finite automaton that accepts L. Since L_i is composed of the set of strings of length $ki + m$ in L, the finite automaton M_i that accepts L_i can be obtained by unrolling M for $ki{+}m$ times. The set of states of M_i that are reachable at the same time by breadth first traversal, and originate from the same state of M, are necessarily equivalent to each other and therefore merged in M_i. Hence, the increase in the number of states, when the depth of breadth first search is increased by 1, is less than or equal to the number of states of M. The finite automaton that accepts $L_i \cdot (1 + 0)^*$ is easily obtained by replacing the set of accepting states and the set of unaccepting states reached at depth $ki{+}m$ with an accepting trap state and an unaccepting trap state, respectively. So the number of nodes of the BDD for f_i increases at most by the number of states of M, owing to Corollary 2. □

4 Generation of Network Invariants

Network invariants are properties of an iterative system that are satisfied by the system regardless of the number of cells. Once they are obtained, they can be used in inductions to prove some properties of interest. A network invariant can be defined by the set of strings that can be generated by any cell in the network with an infinite number of cells. We are only interested in the case when such a network invariant can be represented by a finite automaton.

Definition 4. A sequential iterative system S is bounded iff the language $L = \bigcup_1^{\omega} L_n$ is regular, where L_n is the language produced by the n-th cell of S.

The definition of boundedness can be extended to deal with properties that are related to other class of languages, like ω-regular languages [6]. Boundedness allows us to use a finite state system to model the behavior of the system with a countably infinite number of cells. Definition 4 generalizes the definition of stability of [13]. One immediate application is the verification of equivalence of two sequential iterative systems. The verification model to check the equivalence of two sequential systems based on cells C and C' is shown in Figure 3. Note that the verification model can be considered as another sequential iterative system.

Lemma 5. *The equivalence of two sequential iterative networks can be verified in constant time if their verification model is bounded.*

Proof. Suppose the verification model is bounded. Then there exists a finite automaton which accepts the language L of Definition 4. The verification problem is equivalent to checking the occurrence of the symbol which indicates the nonequivalence of the two systems in any string accepted by the automaton. This can be done in time that is finite and depends only on the size of the automaton accepting L and not on the number of cells. □

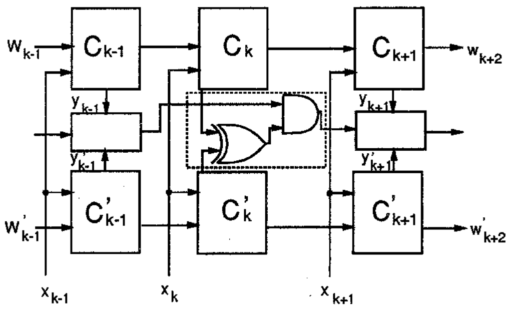

Fig. 3. Verification Model of Iterative System.

Whether a system is bounded is an undecidable problem. If not, the problem of equivalence of sequential iterative system would be decidable. However, we have some conditions that can be checked automatically and used to generate network invariants, if there exists any.

4.1 Unilateral Systems

In this section, we extend the results presented in [13] for unilateral systems to include more general cases and explain the older ones based on the new definition of boundedness.

Theorem 6. *Given a sequential unilateral iterative system S with cell C, the BDD's for the sets of reachable states of S have size linear in n, the number of cells, if the transpose system S^T is bounded.*

Proof. Let $L^T = \bigcup_1^\omega L_k^T$ be the language that is produced by S^T. By definition of transpose of a sequential unilateral system, L_k^T, the language produced by k-th cell of the transpose system is the set of the reachable states of S at time k. Hence, L^T can be expressed in terms of $L_n'^T$ as $L^T = \bigcup_1^\omega L_n'^T$, where $L_n'^T$ is the set of reachable states of the instance of S of length n. The class of functions representing the reachable states of S can be defined according to L^T as $\{f_i : i > 0, f_i(s) = 1, \forall s \in L_i'^T\}$. Semantically, f_i is the characteristic function of the set of reachable states of an i-cell system. Since L^T is regular, the BDD's for $\{f_i\}$ are linear by Lemma 3. □

Boundedness of a system guarantees that its transpose system has linear size BDD's for the sets of reachable states. Also, verification can be done in constant time if a given system is the verification model of two sequential iterative systems, thanks to Lemma 5. We now present sufficient conditions for boundedness.

Lemma 7. $<$ Boundedness 1 $>$ *A unilateral sequential system S is bounded if $L_l \subseteq \bigcup_{k=1}^{l-1} L_k$ for some l, where L_k is the language produced by the k-th cell of S.*

Proof. The monotonicity of the FSM language transfer function guarantees that the language L_{k+l} for $k > 0$ is included in the language $\bigcup_{k=1}^{l} L_k$, i.e., $L = \bigcup_{n=1}^{\omega} L_n = \bigcup_{n=1}^{l-1} L_n$. Hence, L, which is the union of the finite number of regular languages, is also regular. □

Lemma 7 with following corollary replaces the theorem for stability in [13].

Corollary 8. <Boundedness 2> *The unilateral sequential system S is bounded if $L_{k+l} = L_k$ for some k, l, where L_k is the language produced by the k-th cell of S.*

Corollary 8 is a special case of Lemma 7, but it is stated separately because it has further-reaching implications.

Lemma 9. <Boundedness 3> *The unilateral sequential system S is bounded if $L^T_{k+l} = L^T_k$ for some k, l, where L^T_k is the language produced by the k-th cell of S^T.*

Proof. Let L be the language that is generated by S, i.e., $L = \bigcup_{k=1}^{\omega} L_k$. L describes the set of reachable states of S^T [13]. We will show that the language L can be accepted by a finite automaton. For the rest of the proof, the system S' will be used instead of S^T by considering l adjacent cells of S^T as a single cell of S'. The condition $L^T_{k+l} = L^T_k$ can be replaced with $L'_{k+1} = L'_k$. Since the input languages to the each cell of S' are the same starting from the k-th cell, any m adjacent cells have the same reachable states as any other m cells for $m \geq 1$. If we consider the cases of $m = 1$ and $m = 2$, we see that the set of states of the $(j+1)$-th cell that can be reached at the same time as a state of the j-th cell is uniquely determined by the k-th and $(k+1)$-th cell for $j \geq k$. So an automaton M that accepts L can be defined as follows. $M = \{K_o \cup \{S\}, \Sigma, \Delta, F, \{S\}\}$, where $K_o = \{n$ reachable states of the k-th cell$\}$, $\Sigma = \{$all the reachable states of a cell$\}$, $\Delta = \{$up to n transitions $\subseteq \{S\} \times 2^{\Sigma^{k-1}} \times K_o\} \cup \{p$ transitions $\subseteq K_o \times \Sigma \times K_o\}$, $F = K_o$, and S is the initial state. p is the number of

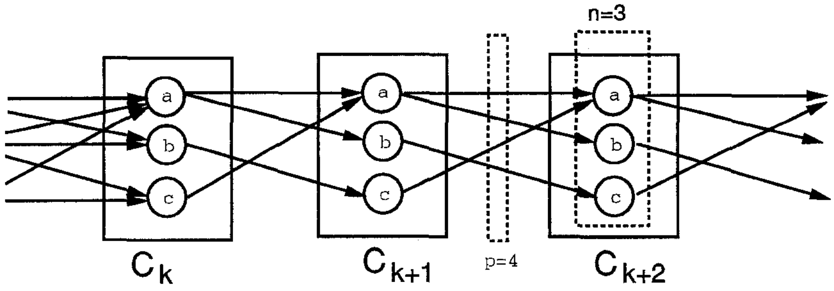

Fig. 4. Reachable States of Iterative System of Lemma 9.

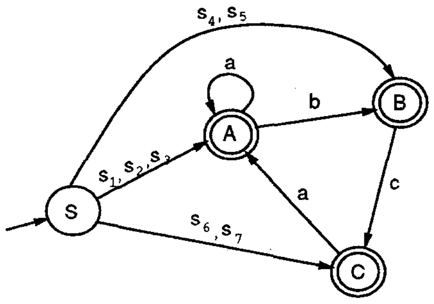

Fig. 5. Finite Automaton that Accepts the Reachable States of Lemma 9.

reachable pairs of states of the k-th and $(k+1)$-th cells, and n is the number of reachable states of the k-th cell. Since n and p are finite, L is regular. □

Figure 4 shows a case with $n = 3$ and $p = 4$. The arrows in the figure identify the pairs of states that are reachable at the same time in two adjacent cells. The corresponding finite automaton that accepts the reachable states of Figure 4 is shown in Figure 5. In Figure 5, $s_1, s_2, \ldots, s_7$ are the strings defined on the alphabet of reachable states of a cell. S is the initial state, and $\{A, B, C\}$ are the accepting states. If every state of K_o has at least two transitions, the number of reachable states is exponential in the number of cells.

Corollary 10. *Given the unilateral sequential system S, the transpose system S^T is bounded if $L_{k+l} = L_k$ for some k, l, where L_k is the language produced by the k-th cell of S.*

Corollary 10 derives from Lemma 9 by the duality of a unilateral iterative system and its transpose.

Corollary 11. *A unilateral sequential iterative system can be implemented by a finite state machine and a unilateral combinational iterative system, if the condition of Lemma 9 is satisfied and p is equal to n in its proof.*

Proof. Since every state in K_o has at least one transition in M, then from $p = n$ it follows that each state of M has exactly one transition out of it. The present state of the i-th cell depends only on the present state of the previous cell for $i > k$. So the outputs of the cell only depends on the cell inputs. Hence the function implemented by the cell is combinational. □

Table 1 shows the implications of boundedness on linearity of BDD's for the sets of reachable states and on the possibility of constant time verification for the original and the transpose systems. For instance, if Boundedness 1 is satisfied, the transpose system will have linear BDD's and constant time verification is possible for the original system, thanks to Theorem 6 and Lemma 5, respectively. Boundedness 2 and Boundedness 3 provide the conditions for the transpose

Table 1. Implications of Boundedness Conditions for Unilateral Systems.

	Boundedness 1	*Boundedness 2*	*Boundedness 3*
Linear BDD's	*T*	*O,T*	*O,T*
Constant Time Verification	*O*	*O,T*	*O,T*

system to be bounded as well as for the original system thanks to Lemma 9 and Corollary 8, respectively. Hence, they guarantee both the original and the transpose system to be bounded.

4.2 Bilateral and Circular Systems

A bilateral system is an iterative system which has information flows in both directions. A circular system is an iterative system whose cells are connected in a ring. In a bilateral network, the boundary conditions are given at the both ends. For the analysis of bilateral and circular systems, we define the *string image function* which is the transfer function of the cell of the given finite state system after abstraction of the local inputs. Let $\phi(s_b, s_p)$ be the string transfer function of a bilateral cell as shown in Figure 6, where s_b is the input string from the communication inputs (in both directions) and s_p is the string applied to the local inputs. Let $L_i^j[in]$ ($L_i^j[out]$) be the language that is applied (produced) between the left of the i-th cell and the right of the j-th cell, where $i \leq j$. We define the string image function F such that $F(s_b) = \{\phi(s_b, s_p) : s_p \in \Sigma_p^*\}$. The composition $\otimes$ of ϕ_b can be defined as in Figure 6. Similarly, the string image function F^k for k adjacent cells is defined as $F^k(s_b) = \{(\otimes_1^k \phi)(s_b, s_p) : s_p \in (\prod_1^k \Sigma_p)^*\}$.

Lemma 12. *A bilateral sequential iterative system is bounded if there exists k such that $F^k(s) = F^{k+1}(s)$ for all $s \in \Sigma_b^*$.*

Proof. Suppose we have an n-cell bilateral network with $n > 2k$. Consider the i-th cell of the system, where $1 \leq i \leq n$. As far as the number of cells on the left and on the right of the i-th cell is greater than k, the left- and right-most $k+1$ cells can be replaced with k cells recursively. Finally, the network can be reduced to have at most $2k+1$ cells including the i-th cell of the given system. So the language produced by i-th cell of the system is always included in the regular language of $L_{bound} = \bigcup_{j=1}^{2k+1} L_j^j[out]$, which is the union of a finite number of regular languages. □

Lemma 12 defines a homomorphism between $(\prod_1^k \Sigma_p)^*$ and $(\prod_1^{k+1} \Sigma_p)^*$. Let ϕ_1^k be $\otimes_1^k \phi$. For $U, V \subseteq (\prod_1^{k+1} \Sigma_p)^*$, one can verify that $f(U+V) = f(U) + f(V), f(U \cdot V) = f(U) \cdot f(V)$, and $f'(U) = f(U')$, where $f : (\prod_1^{k+1} \Sigma_p)^* \rightarrow (\prod_1^k \Sigma_p)^*$ and $\phi_1^k(s_b, L_1) = \phi_1^{k+1}(s_b, f(L_1))$. We can check the condition given in

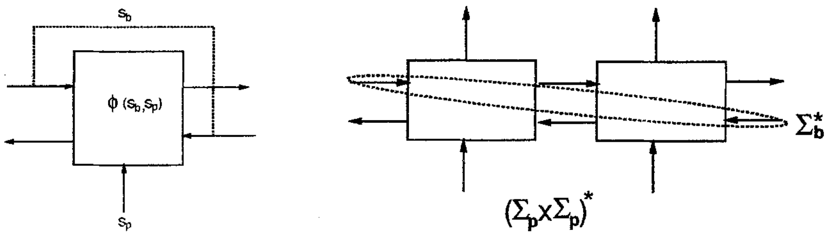

Fig. 6. A Bilateral Cell and the Composition $\otimes$ of Two Bilateral Cells.

Lemma 12 by abstracting local inputs in ϕ_1^k and ϕ_1^{k+1} and checking the equivalence of the two machines. This entails checking the equivalence of two nondeterministic finite state machines. To do this, we first convert the nondeterministic FSM's to deterministic machines with nondeterministic outputs and then the nondeterministic outputs are encoded with deterministic values. This allows us to use a conventional finite state machine equivalence checking program to check the condition of Lemma 12. Lemma 12 is guaranteed to be effective regardless of the boundary conditions. However, it is sometimes too strong. If the boundary conditions are known, we have a weaker condition for boundedness.

Lemma 13. *A bilateral iterative system is bounded if the communication languages observed between the i-th and the j-th cell are the same as the languages observed between the $(i+1)$-th and the j-th cell, or between the i-th and the $(j-1)$-th cell, i.e., $L_i^j[in] = L_i^{j-1}[in]$ and $L_i^j[out] = L_i^{j-1}[out]$, where $i \leq j$.*

The proof of Lemma 13 is similar to the one of Lemma 12. In a circular system, there are no boundary conditions. Let L_i be the language generated by the i-th cell of a circular system and L^j be the union of the languages generated by all the cells of a j-cell network, i.e., $L^j = \bigcup_{i=1}^{j} L_i$. For circular systems, we have the following results.

Lemma 14. *In a circular network with n cells, where all cells start from the same initial state, all cells have the same language, i.e., $L_k = L_m$, for $1 \leq k, m \leq n$.*

Proof. By symmetry. □

Note that L^i is not necessarily the same as L^j. The language depends on the number of cells of the network.

Lemma 15. *In a unilateral circular system, the language $L = \bigcup_{i=1}^{\omega} L^i$ is included in L^M, i.e., $L \subseteq L^M$, where L^M is the greatest fixed point language of the cell.*

Proof. L^i is a fixed point of the cell and always satisfies $L^i \subseteq L^M$. □

We present a special condition for L and L^M to be the same.

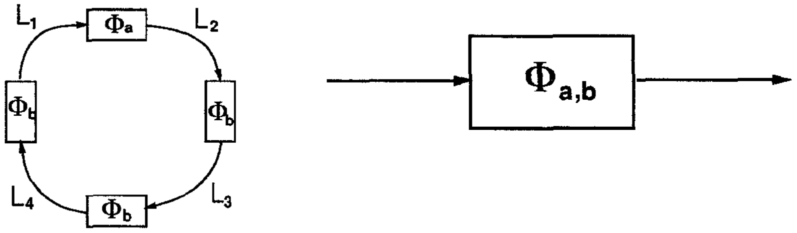

Fig. 7. Circular Iterative Network with Two Initial States and Nondeterministic Model.

Lemma 16. *If L^M is a regular language and the number of strings in L^M is countable, the system is bounded and $L = L^M$.*

Proof. In a circular system, a set of strings that is transitively closed under the *string image function* of the cell, i.e., $\exists s_1, s_2, \ldots, s_k$ such that $s_1 \in F(s_2 \in F(\cdots s_k \in F(s_1)))$, can be observed in the system. Since we consider a countably infinite number of cells in the iterative system, and L^M is a fixed point that has the same closure property as L_i, if L^M has a countable number of strings, then for all the strings in L^M, $k \leq \omega$ and hence they can be observed in the system. □

However, the number of strings in a regular language can be uncountably infinite. For instance, we have a bijection between 2^N and the language $(1+0)^*$ such that an element of $2^{\{1,2,\ldots,k\}}$ is mapped onto an element of $\{s \in (1+0)^* : |s| \leq k\}$, where 2^N is the power set of the natural numbers. Hence, verification with L^M works only in one direction. Now, we consider the case of more than one initial state.

Corollary 17. *In a unilateral circular system of cells not having the same initial states, the language $L = \bigcup_1^\omega L^i$ is included in L^M, where L^M is the greatest fixed point language of the nondeterministic cell having a set of initial states as shown in Figure 7.*

Proof. The language of the nondeterministic cell includes the languages of the deterministic cells. Hence, for an i-cell network, if we replace the cells with the nondeterministic cells, the language $L^i = \bigcup_{k=1}^i L_k$ is the input language as well as the output language of a cell. So L^i is the fixed point language of the nondeterministic cell and is included in L^M which is the greatest fixed point of the nondeterministic cell due to Lemma 15. □

In case of a bilateral circular system, we can use the same boundedness condition as for the bilateral system (Lemma 12).

4.3 Computation of Language Fixed Points

Due to its monotonicity, the transfer function of an FSM can be considered as a predicate transformer on an infinite partial order. The partial order is defined by the language inclusion relation. Since the set is not finite, fixed points of the predicate transformer except the trivial one (the empty string) do not necessarily exist. (The empty string is the least fixed point of any FSM.) In most cases, we are interested in the greatest fixed point. We use a simple iterative method to obtain the greatest fixed point if one exists. The languages of a unilateral iterative system define a partial order for the greatest fixed point. The greatest value is given by $L_{gv} = \{s | s \in \phi(s_{in}), \forall s_{in} \in \Sigma_{in}^*\}$. A unique greatest fixed point can be obtained, if it exists, by applying the greatest value of the partial order to the boundary of the unilateral iterative system and by checking its convergence. An FSM produces a regular language when its input language is regular. So the languages in the series produced from L_{gv} are also regular. A regular language is accepted by a unique finite automaton which is state-minimal. Hence, convergence can be checked by graph isomorphism on finite automata. Similarly, the inclusion relation of two regular languages can be checked by minimizing the number of states of the parallel composition of the two finite automata.

5 An example

In this section, we present a simple example to show how to apply the proposed techniques. Before presenting the example, it may be interesting to see when it is the case that two iterative systems are equivalent while their basic cells are not equivalent. In the verification of two different implementations, the most common cases are when states, inputs, or outputs of the basic cells are encoded differently. However, this case can be dealt with easily if the encodings are known. The more interesting case is dealt with when two iterative systems with non equivalent basic cells become equivalent due to don't care sequences. Don't care sequences form the language of the sequences that are never applied to a cell. Since the number of languages that can be produced by an FSM can be infinite, it is not always possible to determine the all the don't care sequences. The definition of *boundedness* identifies whether such don't care sequences form a regular language or not. If it is regular, the verification can be done by checking whether a string that violates the equivalence of two basic cells is included in

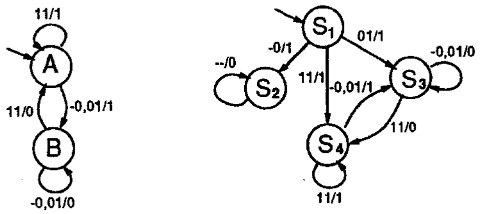

Fig. 8. Two Different Cells.

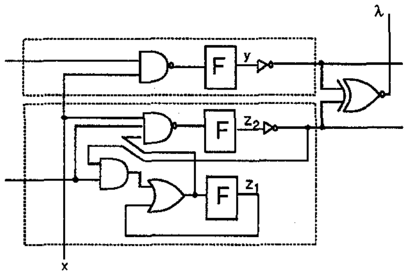

Fig. 9. Verification Model of Two Implementations.

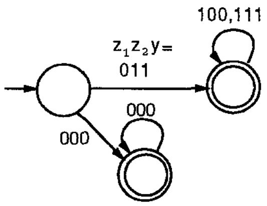

Fig. 10. Bounded Language.

that regular language.

Figure 8 shows two simple FSM's that are not equivalent. We want to verify if two iterative systems with those FSM's as basic cells are equivalent. Figure 9 shows one possible implementation of such iterative systems considering the first input as a local input and the second input as a communication input. The boundary condition of the first cells is 0. Note that this is not a case of different encodings, because we start from two minimized FSM's with different numbers of states. The transpose of the verification system satisfies condition Boundedness 1. The finite automaton that accepts the language of the reachable states of the original system is illustrated in Figure 10. The verification is done under those reachable states by verifying that $R \leq \lambda$, where R is the characteristic function of the reachable states and λ is the function of the output of the verification model. The condition is easily seen to be satisfied by observing that $y = z_2$ in all reachable states.

6 Conclusions

We have presented techniques to check whether there exist network invariants in an iterative finite system and, if an invariant exists, to generate it automatically. We applied our methods to verify properties of circular systems as well as to check the equivalence of two different implementations of sequential iterative systems. Compared to previous ones, our method is fully automatic and the procedure generates network invariants that can be used in inductive proofs. We introduced the concept of boundedness that applies to infinite networks. To summarize, a property is valid for an iterative system with an arbitrary finite number of cells, if it is valid for an infinite iterative system with a countably infinite number of cells. In some cases the condition is exact, but not always. We also showed the implications of boundedness on the size of BDD's for the reachable states of iterative systems. The relation between BDD's and a class of finite automata was investigated. It allows us to infer properties of BDD's and to explain some interesting results previously observed in iterative systems.

References

1. Browne, M. C., Clarke, E. M., and Grumberg, O.: Reasoning about networks with many identical finite state processes. *Information and Computation 81*, 1 (Apr. 1989), 13–31
2. Bryant, R. E.: Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers C-35*, 8 (Aug. 1986), 677–691
3. Burch, J. R., Clarke, E. M., McMillan, K. L., and Dill, D. L.: Sequential circuit verification using symbolic model checking. In *Proceedings of the Design Automation Conference* (June 1990), pp. 46–51
4. Chiodo, M., Shiple, T. R., Sangiovanni-Vincentelli, A., and Brayton, R. K.: Automatic reduction in CTL compositional model checking. In *Proceedings of the International Conference on Computer-Aided Design* (Santa Clara, CA, Nov. 1992), pp. 172–178
5. Cho, H., Hachtel, G. D., Jeong, S.-W., Plessier, B., Schwarz, E., and Somenzi, F.: ATPG aspects of FSM verification. In *Proceedings of the IEEE International Conference on Computer Aided Design* (Nov. 1990), pp. 134–137
6. Choueka, Y.: Theories of automata on ω-tapes: A simplified approach. *J. Comput. Syst. Sci. 8* (1974), 117–141
7. Clarke, E. M., Grunberg, O., and Long, D. E.: Model checking and abstraction. In *Proceedings of the 19th ACM Symposium on Principles of Programming Languages* (Jan. 1992)
8. Coudert, O., and Madre, J. C.: A unified framework for the formal verification of sequential circuits. In *Proceedings of the IEEE International Conference on Computer Aided Design* (Nov. 1990), pp. 126–129
9. Hennie, F. C.: *Iterative Arrays of Logical Circuits.* The M.I.T. Press and John Wiley, New York, 1961
10. Hennie, F. C.: *Finite-State Models for Logical Machines.* John Wiley, New York, 1968
11. Kurshan, R. P., and McMillan, K. L.: A structural induction theorem for processes. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing* (Edmonton, Alberta, Canada, Aug. 1989), pp. 239–247
12. Macii, E., Plessier, B., and Somenzi, F.: Verification of systems containing counters. In *Proceedings of the International Conference on Computer-Aided Design* (Santa Clara, CA, Nov. 1992), pp. 179–182
13. Rho, J.-K., and Somenzi, F.: Inductive verification for iterative systems. In *Proceedings of the Design Automation Conference* (Anaheim, CA, June 1992), pp. 628–633
14. Sistla, A. P., and German, S. M.: Reasoning with many processes. In *Proceedings of the Symposium on Logic in Computer Science* (Ithaca, NY, June 1987), pp. 138–152

15. Touati, H., Savoj, H., Lin, B., Brayton, R. K., and Sangiovanni-Vincentelli, A.: Implicit enumeration of finite state machines using BDD's. In *Proceedings of the IEEE International Conference on Computer Aided Design* (Nov. 1990), pp. 130–133
16. Wolper, P.: Expressing interesting properties of programs in propositional temporal logic. In *Proceedings 13th ACM Symposium on Principles of Programming Languages* (St. Petersburgh, Jan. 1986), pp. 184–192
17. Wolper, P., and Lovinfosse, V.: Verifying properties of large sets of processes with network invariants. In *Automatic Verification Methods for Finite State Systems, Lecture Notes in Computer Science 407*, J. Sifakis, Ed. Springer-Verlag, 1989, pp. 68–80