Symbolic State Exploration

Fabio Somenzi [1]

Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO, USA

Abstract

The exploration of the state space of the model is at the heart of model checking. Symbolic algorithms often use Binary Decision Diagrams for the representation of sets of states, and have to pay attention to the size of the BDDs. We review the techniques that have been proposed for this task in the framework of closed sequences of monotonic functions.

1 Introduction

Formal verification techniques like model checking [13] explore the state space of the system to be verified. To combat the exponential growth of the number of states with the number of state variables, symbolic algorithms for state exploration refrain from dealing explicitly with individual states: Instead, they manipulate the characteristic functions of sets of states. In the late eighties, the foundations for symbolic state exploration were laid by the work of McMillan and co-workers [3], Coudert, Berthet, and Madre [8,9], and Pixley [17]. The common trait of those early works was the formulation of decision procedures for classes of properties in terms of state reachability, and the solution of the reachability problem by breadth-first search of the state transition graph, with graph and state sets represented by Binary Decision Diagrams (BDDs) [2].

In cases of properties like $\mathsf{EF}\,p$ (there exists an execution of the system during which p holds at some time), the decision procedure had to establish the reachability of a state labeled p from other states. This could be done by moving either forward or backward in the state transition graph. In the case of properties like $\mathsf{EG}\,p$ (there exists an execution of the system during which p holds at all times), the decision procedure had to establish the existence of a cycle labeled by p reachable from some other states along a path also labeled

[1] This work was supported in part by SRC contract 98-DJ-620.

by p. The first problem translates into the computation of a least fixpoint, while the second entails the computation of a greatest fixpoint.

In general, the logics commonly used in model checking can be expressed in terms of μ-calculus [11,13,10], and in practice, a model checker spends most of its time computing fixpoints. State exploration and computation of fixpoints are therefore closely related in the context of symbolic model checking. Fixpoint computation is founded on Tarksi's theorem [22], which states that the least and greatest fixpoints of monotonic functions over finite lattices can be computed by repeated application of the functions, starting from either the least elements of the lattices (for least fixpoints) or the greatest elements of the lattices (for greatest fixpoints). The powerset of the set of states is the lattice on which the (formally) monotonic functions expressing the fixpoints of interest in model checking operate. The repeated application of functions maps quite naturally onto symbolic breadth-first search.

Symbolic breadth-first search, however, is not always the most efficient approach to state exploration. Consider the computation of the states reachable from a designated state s. At each iteration, a lower bound to the fixpoint is computed. Specifically, the initial lower bound is $\{s\}$, and at the k-th iteration of the symbolic depth-first search algorithm, all states such that the shortest path from s to them has length k are added to the previous bound to create a new one. The sequence of lower bounds—the iterates of the fixpoint computation—obviously form a monotonic sequence. However, the sizes of the BDDs that represent the iterates seldom form a monotonic sequence: In many cases, the BDD sizes reach a maximum that may be several times the value at convergence. An example is shown in Figure 1. The reason for such a behavior of the BDD sizes is that the states at a given maximum distance from the initial states may not be well clustered in the the boolean space. This leads to inefficient representation in terms of BDDs. As the computation proceeds, the gaps between states may be filled by other states, leading to better BDDs. Sometimes large intermediate sizes are the result of sub-optimal BDD variable orders, and can be remedied by applying dynamic reordering [12]. Other times, there is no "good" order, and the growth in BDD size triggers several expensive reorderings to no avail.

The problem with intermediate results being much larger than the final results occurs also within each step of the breadth-first search procedure. Each step computes either the successors or the predecessors of a set of states by a series of conjunctions and quantifications. The BDDs for the intermediate results are often much larger than those encountered at the end. This problem compounds the one of the sizes of the fixpoint iterates, and often prevents or seriously hinders completion of a model checking run.

Symbolic breadth-first has other problems besides the size of the intermediate results. On the one hand, even an optimal search strategy may produce excessively large BDDs, either at some intermediate point, or at the end. Computing an approximation to a fixpoint may be the only alternative to

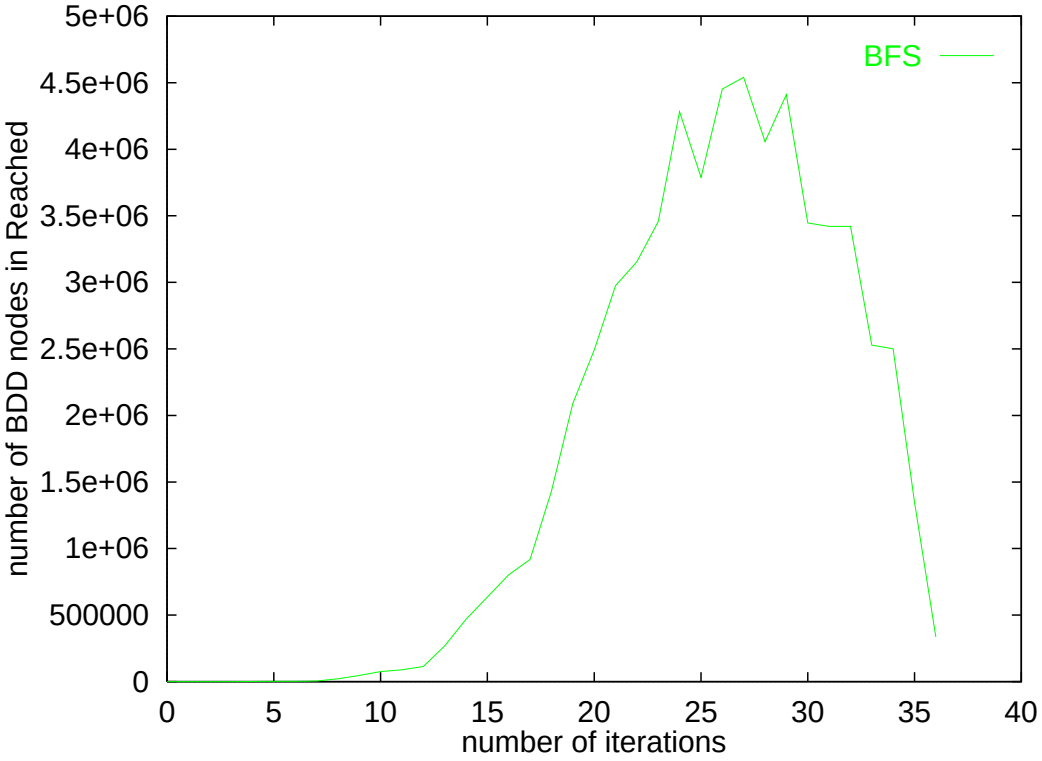

Fig. 1. Size of the BDDs for fixpoint iterates.

giving up on the exact computation. On the other hand, obtaining the entire fixpoint may be a waste of time, as in the case in which it is sufficient to establish reachability of one state from some other states: Once a path is found that connects the desired states, the computation can be terminated. The ideal search strategy proceeds towards the target states, possibly disregarding states that are closer to the origin.

In summary, the size of intermediate results, the impossibility to compute the fixpoints of some large problems exactly, and the desire to focus the search in the directions that cause early termination has motivated the development of alternatives to breadth-first search for fixpoint computation. It goes without saying that the opposite of breadth-first search—depth-first search—is usually much worse in the context of symbolic state exploration. Therefore, the methods that have been devised combine elements of depth-first search in a basic scheme that is still quite close to breadth-first in that it tries to manipulate large sets of states at once.

This paper reviews some of the search strategies that have been proposed in a uniform framework. This framework—closed sequences of monotonic functions—is presented in Section 2. It is a natural extension of the results on fixpoints of monotonic functions to the case in which multiple functions need to be evaluated. Section 3 formulates several search strategies in terms of closed sequences, and Section 4 concludes.

2 Closed Sequences of Monotonic Functions

Definition 2.1 Let L be a finite lattice and let $T = \{\tau_1, \ldots, \tau_k\}$ be a finite set of monotonic functions $L \to L$. For $l_1, l_2 \in L$, let $l_1 + l_2$ be the least upper bound of l_1 and l_2. For σ a finite sequence over T, let τ_σ be the function $L \to L$ obtained by composing all the functions in σ in the order specified by the sequence. Let $|\sigma|$ be the length of sequence σ. Let (σ, ρ) designate the concatenation of sequences σ and ρ. We say that σ is *closed* if for $i = 1, \ldots, k$ we have $\tau_i(\tau_\sigma(0)) = \tau_\sigma(0)$.

If σ is closed, $\tau_\sigma(0)$ is a fixpoint of every $\tau_i \in T$. If σ is closed over T, then it is closed over any subset of T.

Lemma 2.2 *Let σ be a finite sequence over T. Then τ_σ is monotonic.*

Proof. By induction on $|\sigma|$. If $|\sigma| = 0$ there is nothing to prove. Suppose $|\sigma| = i + 1$. Let σ^i be the prefix of length i of σ. Then by the inductive hypothesis and the monotonicity of σ_{i+1}:

$$l_1 \leq l_2 \Rightarrow \tau_{\sigma^i}(l_1) \leq \tau_{\sigma^i}(l_2) \Rightarrow \sigma_{i+1}(\tau_{\sigma^i}(l_1)) \leq \sigma_{i+1}(\tau_{\sigma^i}(l_2)).$$

Therefore $\tau_\sigma(l_1) \leq \tau_\sigma(l_2)$. □

Lemma 2.3 *Let $\sigma = (\sigma_1, \sigma_2)$ be a finite sequence over T. Then $\tau_{\sigma_2}(0) \leq \tau_\sigma(0)$.*

Proof. From $0 \leq \tau_{\sigma_1}(0)$ and the monotonicity of τ_{σ_2}:

$$\tau_{\sigma_2}(0) \leq \tau_{\sigma_2}(\tau_{\sigma_1}(0)) = \tau_\sigma(0).$$

□

Lemma 2.4 *Let σ be a finite closed sequence over T. Let ρ be any finite sequence over T. Then $\tau_\rho(0) \leq \tau_\sigma(0)$.*

Proof. By Lemma 2.3 and closure of σ:

$$\tau_\rho(0) \leq \tau_{(\sigma,\rho)}(0) = \tau_\sigma(0).$$

□

Corollary 2.5 *Let ρ, σ be two closed sequences over T. Then*

$$\tau_\rho(0) = \tau_\sigma(0).$$

Proof. By Lemma 2.4, we have:

$$\tau_\rho(0) \leq \tau_\sigma(0) \text{ and } \tau_\sigma(0) \leq \tau_\rho(0).$$

□

Lemma 2.6 *Let $\sigma = (\sigma_1, \ldots, \sigma_p)$ be a sequence of monotonic functions over L. Let $\rho = (\rho_1, \ldots, \rho_p)$ be a sequence of functions over L such that $\rho_i \geq \sigma_i$ for $i = 1, \ldots, p$. Then $\tau_\rho(0) \geq \tau_\sigma(0)$.*

Proof. By induction on p. For $p = 0$ we have $0 \geq 0$. Let σ^i (ρ^i) be the prefix of length $p-1$ of σ (ρ). Assume $\tau_{\rho^i}(0) \geq \tau_{\sigma^i}(0)$. Then

$$\rho_{i+1}(\tau_{\rho^i}(0)) \geq \sigma_{i+1}(\tau_{\rho^i}(0)) \geq \sigma_{i+1}(\tau_{\sigma^i}(0)),$$

where the first inequality comes from $\rho_{i+1} \geq \sigma_{i+1}$, and the second comes from the inductive hypothesis and the monotonicity of σ_{i+1}. □

Lemma 2.7 *Let σ be a finite closed sequence over T. Let $\delta \leq \tau_\sigma(0)$. Then:*

$$\tau_\sigma(\delta) = \tau_\sigma(0).$$

Proof. From $0 \leq \delta \leq \tau_\sigma(0)$, we get by monotonicity and closure of τ_σ:

$$\tau_\sigma(0) \leq \tau_\sigma(\delta) \leq \tau_\sigma(\tau_\sigma(0)) = \tau_\sigma(0).$$

□

Theorem 2.8 *Let σ be a closed sequence over T. Let $\gamma = \sum_{\tau \in T} \tau$ be the pointwise least upper bound of all the functions in T. Then*

$$\tau_\sigma(0) = \mu X.\gamma(X).$$

Proof. By definition of γ and closure of σ we have:

$$\gamma(\tau_\sigma(0)) = \sum_{\tau \in T} \tau(\tau_\sigma(0)) = \sum_{\tau \in T} \tau_\sigma(0) = \tau_\sigma(0).$$

Hence, $\tau_\sigma(0)$ is a fixpoint of γ, which implies $\tau_\sigma(0) \geq \mu X.\gamma(X)$. Let ρ be the sequence obtained by repeating γ $|\sigma|$ times. Then by Lemma 2.6, we have:

$$\mu X.\gamma(X) \geq \tau_\rho(0) \geq \tau_\sigma(0).$$

Hence, $\tau_\sigma(0)$ is the least fixpoint of γ. □

Theorem 2.8 only requires monotonicity of the functions in T. In reachability analysis we can assume that $\tau(x) \geq x$, but this additional assumption is unnecessary. Notice that the function over the boolean algebra $B = \{0, a, b, 1\}$ defined by $\tau = \lambda x.a$ shows that for a monotonic function it is possible to have $\tau(x) < x$, and also $\tau(x)$ not comparable to x.

However, there may not exist closed sequences for a given T. Consider $T = \{\tau_0, \tau_1\}$, with $\tau_0 = \lambda x.0$ and $\tau_1 = \lambda x.1$. Let σ be a sequence over T. If σ ends with τ_0, then $\tau_\sigma(0) = 0 \neq \tau_1(\tau_\sigma(0)) = 1$. Conversely, if σ ends with τ_1, then $\tau_\sigma(0) = 1 \neq \tau_0(\tau_\sigma(0)) = 0$.

Definition 2.9 A function $\tau : L \to L$ is *upward* if $\tau(x) \geq x$ for all $x \in L$.

Definition 2.10 An infinite sequence $\sigma = (\sigma_1, \ldots, \sigma_n, \ldots)$ over T is *fair* if, for every $n > 0$ and for $0 < j \leq k$ there exists $i > n$ such that $\sigma_i = \tau_j$.

Theorem 2.11 *If all functions in T are upward, then every fair sequence over T has a finite prefix that is closed.*

Proof. Let σ^i be the prefix of length i of a fair sequence σ. Since σ_{i+1} is upward, $\tau_{\sigma^{i+1}}(0) \geq \tau_{\sigma^i}(0)$. From the finiteness of L it follows that there exists an $n > 0$ such that for $i > n$, $\tau_{\sigma^i}(0) = \tau_{\sigma^n}(0)$. Let m_j be the smallest $i > n$

such that $\sigma_i = \tau_j$, $1 \leq j \leq k$. Then $\sigma_{m_j}(\tau_{\sigma^{m_j-1}}(0)) = \sigma_{m_j}(\tau_{\sigma^n}(0)) = \tau_{\sigma^n}(0)$. Hence, σ^n is closed. □

Lemma 2.12 *Let $\gamma : L \to L$ be a monotonic function and let $\hat{\gamma} = \lambda x.x + \gamma$. Then $\mu x.\gamma = \mu x.\hat{\gamma}$.*

Proof. By Lemma 2.6, $\mu x.\gamma \leq \mu x.\hat{\gamma}$. We prove by induction that $\mu x.\gamma \geq \mu x.\hat{\gamma}$. For the basis we observe that $0 \leq \mu x.\gamma$. For the inductive step, we assume $\hat{\gamma}^{i-1}(0) \leq \mu x.\gamma$. We then have:

$$\hat{\gamma}^i(0) = \hat{\gamma}(\hat{\gamma}^{i-1}(0)) \leq \hat{\gamma}(\mu x.\gamma) = \mu x.\gamma + \gamma(\mu x.\gamma) = \mu x.\gamma.$$

□

Theorem 2.13 *Let $\gamma = \sum_{\tau \in T} \tau$ be the pointwise least upper bound of all the functions in T. Let $\hat{T} = \{\hat{\tau}_1, \ldots, \hat{\tau}_k\}$, with $\hat{\tau}_i = \lambda x.x + \tau_i$. Let σ be a fair sequence over $\hat{T}$. Then σ has a finite prefix ρ that is closed, and such that $\tau_\rho(0) = \mu x.\gamma$.*

Proof. By construction, $\hat{\tau}_i$ is upward. Hence, by Theorem 2.11, σ has a prefix ρ closed over $\hat{T}$. By Theorem 2.8 and Lemma 2.12, $\tau_\rho(0) = \mu x.\hat{\gamma} = \mu x.\gamma$. □

Theorem 2.14 *Let σ be a closed sequence over T. Let $\sigma_\delta : L \to L$ be defined by $\sigma_\delta = \lambda x.x + \delta$, for $\delta \leq \tau_\sigma(0)$. Let σ^i be the sequence obtained from σ by inserting σ_δ after σ_i. Then σ^i is closed and*

$$\tau_\sigma(0) = \tau_{\sigma^i}(0).$$

Proof. Let $\rho = (\sigma, \sigma_\delta)$. We have:

$$\tau_\rho(0) = \tau_\sigma(0) + \delta = \tau_\sigma(0).$$

Hence, ρ is closed over $T \cup \{\sigma_\delta\}$. By Lemma 2.4, $\tau_\rho(0) \geq \tau_{\sigma^i}(0)$. Let $\sigma_{i\delta} = \sigma_i + \sigma_\delta$. Let σ^δ be the sequence obtained from σ by replacing σ_i with $\sigma_{i\delta}$. From Lemma 2.6 it follows that $\tau_{\sigma^\delta}(0) \geq \tau_\sigma(0)$. From $\tau_{\sigma^\delta}(0) = \tau_{\sigma^i}(0)$ we then get $\tau_{\sigma^i}(0) \geq \tau_\sigma(0)$, and finally $\tau_\sigma(0) = \tau_{\sigma^i}(0)$ and that σ^i is closed over T. □

Except for Lemma 2.2 which is its own dual, the remaining results have duals that apply to greatest fixpoint computations. Here we state explicitly the main results.

Definition 2.15 For $l_1, l_2 \in L$, let $l_1 \cdot l_2$ be the greatest lower bound of l_1 and l_2. We say that σ is *1-closed* if for $i = 1, \ldots, k$ we have $\tau_i(\tau_\sigma(1)) = \tau_\sigma(1)$. A function $\tau : L \to L$ is *downward* if $\tau(x) \leq x$ for all $x \in L$.

Theorem 2.16 *Let σ be a 1-closed sequence over T. Let $\beta = \prod_{\tau \in T} \tau$ be the pointwise greatest lower bound of all the functions in T. Then*

$$\tau_\sigma(1) = \nu X.\beta(X).$$

Theorem 2.17 *If all functions in T are downward, then every fair sequence over T has a finite prefix that is 1-closed.*

Theorem 2.18 *Let $\beta = \prod_{\tau \in T} \tau$ be the pointwise greatest lower bound of all the functions in T. Let $\hat{T} = \{\hat{\tau}_1, \ldots, \hat{\tau}_k\}$, with $\hat{\tau}_i = \lambda x.x \cdot \tau_i$. Let σ be a fair sequence over $\hat{T}$. Then σ has a finite prefix ρ that is 1-closed, and such that $\tau_\rho(1) = \nu x.\beta$.*

Theorem 2.19 *Let σ be a 1-closed sequence over T. Let $\sigma_\Delta : L \to L$ be defined by $\sigma_\Delta = \lambda x.x \cdot \Delta$, for $\Delta \geq \tau_\sigma(1)$. Let σ^i be the sequence obtained from σ by inserting σ_Δ after σ_i. Then σ^i is 1-closed and*

$$\tau_\sigma(1) = \tau_{\sigma^i}(1).$$

3 Search Strategies

In symbolic state exploration we assume we are given a monotonic function γ that operates on sets of states of the model to be verified. For instance, the computation of $\mathsf{EF}\, p$ entails the computation of $\mu Z.p \vee \mathrm{Pre}(Z)$, where $\mathrm{Pre}(Z)$ computes all the predecessors of the states in Z. In this case $\gamma = \lambda Z.p \vee \mathrm{Pre}(Z)$. This section shows how, for different search strategies, a set $T = \{\tau_1, \ldots, \tau_k\}$ is derived from γ, and how closed sequences over T are constructed.

3.1 Disjunctive Partitioning

To alleviate the problem of large intermediate results during one iteration of the fixpoint computation, Cabodi *et al.* [5,4] have proposed the disjunctive partitioning of γ. In the simplest form of partitioning, the set $T = \{\tau_1, \ldots, \tau_k\}$ is obtained by identifying a set of state variables such that cofactoring (or restricting) the transition relation according to all their possible assignments leads to τ_i's with small BDDs. The functions are upward, and are applied in round-robin fashion. This gives fair sequences for which Theorems 2.11 and 2.8 guarantee convergence to the desired fixpoint. Cabodi *et al.* allow the partitioning to change at each image computation. Their approach has been extended by Narayan *et al.* [16] to allow, among other things, more flexible sequencing of the τ_i's, while still guaranteeing fairness.

3.2 High Density Reachability Analysis

The states that are expanded in one iteration of fixpoint computation form the *frontier set*. Breadth-first search always tries to expand all unexpanded states (and possibly more). High Density (HD) Reachability Analysis [19,20], on the other hand, may use a subset of the unexpanded states as frontier set. BDD approximation functions [19,18] are used to that effect. Because of subsetting, the function applied at the i-th iteration, τ_i, satisfies the condition $\tau_i \leq \gamma$. Furthermore, all τ_i are upward. However, there is no guarantee that $\gamma = \sum_{i \geq 0} \tau_i$. Hence, HD Reachability Analysis resorts to *dead-end computations* to guarantee convergence. Let $\tau_{\delta j}$ be the function applied when the j-th dead-end computations is performed; $\tau_{\delta j}$ is upward and guarantees that $\tau_{\delta j}(Z) = Z$

only if $\tau_{\delta j}(Z) = \gamma(Z)$. Hence, if a sequence over $T = \{\tau_i\} \cup \{\tau_{\delta j}\}$ is closed, then $\gamma = \sum_{\tau \in T} \tau$. By Theorems 2.11 and 2.8 convergence to $\mu Z.\gamma$ is guaranteed. An approach related to HD Reachability Analysis is Saturated Simulation [23], in which, however, there is no provision to guarantee convergence.

3.3 Symbolic Guided Search

Symbolic guided search [21] applies *hints* to the transition relation of the system to be verified that are meant to enable only some modes of operation, or restrict addresses to some values or sequences. The hints are conjoined to the transition relation to yield the set $T = \{\tau_1, \ldots, \tau_k\}$. All the functions in T are upward. Each function is applied until it yields no further states. Repeated application of the same function enables the use of the frontier set. To guarantee convergence, [21] always includes as last hint the hint that gives $\tau_k = \gamma$. This choice tends to keep the number of iterations needed to reach convergence low. When switching from τ_{j-1} to τ_j, the guided search algorithm cannot use the frontier set. This my lead to an expensive computation, especially when switching to $\tau_k = \gamma$. For this, the dead-end computation algorithm of HD Reachability Analysis is used. An alternative approach, which may be useful to avoid some of the most expensive dead-end computations, is to iterate the application of $\{\tau_1, \ldots, \tau_{k-1}\}$. This alternative approach requires $\sum_{1 \le i \le k-1} \tau_i = \gamma$. Symbolic guided search has been applied to model checking of some LTL properties [1] and also to CTL.

3.4 Approximate Reachability Analysis

Approximate Reachability Analysis [6] can be used to compute a superset of reachable states of a system. This finds applications in sequential optimization, as well as in model checking [14]. The computation of an upper bound to the reachable states is accomplished by decomposing the system in submachines [7]. The decomposition produces a set $T = \{\tau_1, \ldots, \tau_k\}$ such that $\bigcup_{\tau \in T} \tau \ge \gamma$. One can then use Corollary 2.5 to prove that any fair sequence over T produces the same approximation of the reachable states. (As long as the approximation algorithm is otherwise the same for all sequences.) The interesting fact is that some sequences are much more efficiently computed than others [15].

4 Conclusions

Closed sequences of monotonic functions provide a natural framework to express search strategies used in symbolic state exploration. These strategies address the limitations of pure breadth-first search that cause the BDDs produced by the model checking algorithms to grow too large. In this paper we have shown in particular how High Density Reachability Analysis, Disjunctive

Partitioning, Symbolic Guided Search, and Approximate Reachability Analysis can be formulated in a uniform way as computations of closed sequences.

Acknowledgements

I thank Roderick Bloem, Jim Kukula, In-Ho Moon, Kavita Ravi, and Tom Shiple for providing inspiration for this work and comments on early versions of the manuscript.

References

[1] R. Bloem, K. Ravi, and F. Somenzi. Efficient decision procedures for model checking of linear time logic properties. In N. Halbwachs and D. Peled, editors, *Eleventh Conference on Computer Aided Verification (CAV'99)*, pages 222–235. Springer-Verlag, Berlin, 1999. LNCS 1633.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science*, June 1990.

[4] G. Cabodi, P. Camurati, L. Lavagno, and S. Quer. Disjunctive partitionining and partial iterative squaring: An effective approach for symbolic traversal of large circuits. In *Proceedings of the Design Automation Conference*, pages 728–733, Anaheim, CA, June 1997.

[5] G. Cabodi, P. Camurati, and S. Quer. Improved reachability analysis of large finite state machines. In *Proceedings of the International Conference on Computer-Aided Design*, pages 354–360, Santa Clara, CA, November 1996.

[6] H. Cho, G. D. Hachtel, E. Macii, B. Plessier, and F. Somenzi. Algorithms for approximate FSM traversal based on state space decomposition. *IEEE Transactions on Computer-Aided Design*, 15(12):1465–1478, December 1996.

[7] H. Cho, G. D. Hachtel, E. Macii, M. Poncino, and F. Somenzi. Automatic state space decomposition for approximate FSM traversal based on circuit analysis. *IEEE Transactions on Computer-Aided Design*, 15(12):1451–1464, December 1996.

[8] O. Coudert, C. Berthet, and J. C. Madre. Verification of sequential machines based on symbolic execution. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, pages 365–373. Springer-Verlag, 1989. LNCS 407.

[9] O. Coudert, C. Berthet, and J. C. Madre. Verification of sequential machines using boolean functional vectors. In L. Claesen, editor, *Proceedings IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pages 111–128, Leuven, Belgium, November 1989.

[10] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *Theoretical Computer Science*, 126:77–97, 1994.

[11] E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proceedings of the First Annual Symposium of Logic in Computer Science*, pages 267–278, June 1986.

[12] M. Fujita, Y. Matsunaga, and T. Kakuda. On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In *Proceedings of the European Conference on Design Automation*, pages 50–54, Amsterdam, February 1991.

[13] K. L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, Boston, MA, 1994.

[14] I.-H. Moon, J.-Y. Jang, G. D. Hachtel, F. Somenzi, C. Pixley, and J. Yuan. Approximate reachability don't cares for CTL model checking. In *Proceedings of the International Conference on Computer-Aided Design*, pages 351–358, San Jose, CA, November 1998.

[15] I.-H. Moon, J. Kukula, T. Shiple, and F. Somenzi. Least fixpoint approximations for reachability analysis. In *Proceedings of the International Conference on Computer-Aided Design*, San Jose, CA, November 1999. To appear.

[16] A. Narayan, A. J. Isles, J. Jain, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Reachability analysis using partitioned ROBDDs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 388–393, November 1997.

[17] C. Pixley. A computational theory and implementation of sequential hardware equivalence. In E. M. Clarke and R. P. Kurshan, editors, *Computer-Aided Verification '90*, pages 293–320. American Mathematical Society – Association for Computing Machinery, 1991.

[18] K. Ravi, K. L. McMillan, T. R. Shiple, and F. Somenzi. Approximation and decomposition of decision diagrams. In *Proceedings of the Design Automation Conference*, pages 445–450, San Francisco, CA, June 1998.

[19] K. Ravi and F. Somenzi. High-density reachability analysis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 154–158, San Jose, CA, November 1995.

[20] K. Ravi and F. Somenzi. Efficient fixpoint computation for invariant checking. In *Proceedings of the International Conference on Computer Design*, Austin, TX, October 1999. To appear.

[21] K. Ravi and F. Somenzi. Hints to accelerate symbolic traversal. In *CHARME'99*, September 1999. To appear.

[22] A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[23] J. Yuan, J. Shen, J. Abraham, and A. Aziz. On combining formal and informal verification. In O. Grumberg, editor, *Ninth Conference on Computer Aided Verification (CAV'97)*, pages 376–387. Springer-Verlag, Berlin, 1997. LNCS 1254.