Conformance Testing in the presence of Multiple Faults

Viraj Kumar* Mahesh Viswanathan†

Abstract

Conformance testing is the problem of determining if a black-box implementation I is equivalent to a specification S, where both are modeled as finite state Mealy machines. The problem involves constructing a *checking sequence* based on the specification, which is a sequence of inputs that detects all faulty machines. Traditionally, conformance testing algorithms have assumed that the number of states in the implementation does not exceed that in the specification. This is because it is known that, in the absence of this assumption, the length of the checking sequence needs to be at least exponential in the number of extra states in the implementation [41]. However, this has limited the applicability of these techniques in practice where the implementation typically has many more states than the specification.

In this paper we relax the constraints on the size of the implementation and investigate the existence of polynomial length checking sequences for implementations with extra states, under the promise that they either have multiple faults or no faults at all. We present randomized algorithms to construct checking sequences that catch faulty implementations with at most Δ extra states, having at least r faults (where Δ and r are parameters to the algorithm), and pass all correct implementations. We demonstrate the near optimality of our algorithms by presenting lower bounds for this problem. One of the main technical lemmas used in our proof is an estimate of the probability that a random walk on directed graphs will reach a large target set. We believe that this lemma will be of independent interest in the context of verifying safety properties.

1 Introduction

Fault detection is an important problem in the verification of network protocols [22], where one tries to determine if the implementation of a protocol adheres to the specification prescribed in the standard. The protocol specification (or the control portion of the protocol specification) is typically modeled as a deterministic finite state Mealy machine that produces outputs whenever it makes a state transition. The protocol implementation is assumed to be a deterministic "black-box" machine whose internal state transition structure is assumed to be unknown. However, one can test the implementation by applying input symbols and observing the outputs it produces. The objective in fault detection or *conformance testing* is to design a test (called a *checking sequence*) which will help determine if the implementation is indeed equivalent (with respect to language/trace equivalence) to the specification standard.

Since Moore's seminal work [31] that introduced the framework of testing finite state machines, there is an extensive literature on the problem of testing such machines [17, 40, 23, 18, 11, 27, 6], and the fault detection problem [38, 25, 35] in particular. Major results are summarized in [24, 16, 29] (see [28] for a recent survey). A number of algorithms for conformance testing in special cases have been proposed: the D-method based on distinguishing sequences [21] ; the U-method based on UIO sequences [38]; the T-method based on transition tours [33]; checking sequence based on reliable resets [9] are some examples. In an influential paper, Yannakakis and Lee [42] relaxed some of the special constraints imposed by previous algorithms and gave a randomized construction of a polynomially long checking sequence.

All previous algorithms, including the Lee and Yannakakis algorithm, made some common assumptions about the specification and implementation machines. The first is that the specification is a reduced finite state machine and that the specification and implementation are *strongly connected*, i.e., for any pair of states in the specification (respectively, implementation), there is an input sequence that transfers the machine from the first state to the second state[1]. The second more serious assumption is that the implementation does not have more states than the specification. The reason this second strong assumption is usually made is because it is known that in order to detect faulty implementations

*kumar@cs.uiuc.edu University of Illinois at Urbana-Champaign, Urbana, IL 61801. Supported by DARPA/AFOSR MURI award F49620-02-1-0325

†vmahesh@cs.uiuc.edu University of Illinois at Urbana-Champaign, Urbana, IL 61801. Supported by NSF CCF 04-29639

[1] Typically this is not a serious constraint since specifications and implementations often have a reset transition, which guarantees strong connectivity.

that have more states than the specification, the length of the checking sequence needs to be exponential in the number of extra states in the implementation [41]. However, making this assumption severely restricts the applicability of these methods in practice. This is because implementations (being more detailed) typically have many more states than the specification.

In this paper we investigate if this restriction on the size of the implementation can be relaxed, and at the same time obtain polynomially long testing sequences that can provide useful information about the presence of faults in the implementation. More specifically, like in *property testing* [37, 36], we study the conformance testing problem in a promise setting, namely, one where the implementations with additional states either have multiple faults or are completely correct. Once again, like in property testing, the number of faults in an implementation is measured by the Hamming distance to the closest correct implementation [2]. We call a sequence of inputs a (r, Δ)-approximate checking sequence if all implementations with at most Δ extra states and at least r faults give a different output than the specification on this sequence, while correct implementations give the same output as the specification. We present a randomized algorithm that given r and Δ, constructs a (r, Δ)-approximate checking sequence with high probability. It must be noted that the probability of error is over the random choices made by the algorithm and not on a particular distribution on the implementation machines. Secondly the probability of error can be made as small as needed by increasing the length of the checking sequence.

Investigating conformance testing in the promise setting might seem unreasonable in the light of the fact that implementations usually have few subtle errors, rather than many. Nonetheless, we believe that the results presented here are interesting for the following reasons. First, this relaxation of the fault detection problem is a natural mathematical generalization, and one which has been fruitfully studied in the context of property testing ([19, 4, 3, 2, 7, 20, 8, 14, 34, 13, 5]; see [36] for a survey). In addition, apart from its theoretical interest, we believe our constructions will be useful in practice. This is primarily because although we can prove the success (with high probability) of our tests only on machines with at least r faults, the tests have a non-zero probability of detecting every faulty machine (including those with less than r faults). Thus our algorithm's output can be used as a heuristic test that applies in all situations. Second, our algorithm provides the debugging team a hierarchy of test suites of increasing precision to choose from based on practical time constraints imposed on the testing process by product release times.

Our Results. We present two randomized algorithms. The first algorithm constructs an (r, Δ)-approximate checking sequence to detect implementations with a large number of faults; more precisely when $r > d\min(n, \Delta)$, where d is the size of the input alphabet and n is the size of the specification. This algorithm is essentially identical to that of Yannakakis and Lee [42]. Our contribution here lies in showing that this algorithm works in this new setting by observing that the presence of a large number of errors allows one to search for faults in a small neighbourhood, and thus ignore the fact that the implementation has extra states. The second algorithm constructs approximate checking sequences for the case when the number of errors is small, i.e., when $r \leq d\min(n, \Delta)$. This algorithm relies on performing a random walk to reach a target set T among the "unknown" extra states X of the implementation. In general, exploration via random walks takes time exponential in $|X|$ due to the existence of so-called *combination-lock* subgraphs[3]. The hardness of exploring digraphs via random walks has also been characterized in terms of spectral and connectivity properties [12, 26, 30]. For special classes of digraphs, better bounds can be obtained, e.g. polynomial-time upper bounds exist when the digraph is symmetric (i.e. undirected) [32, 39], and when the indegree of each vertex is equal to the outdegree [1]. Good bounds can also be obtained for digraphs with high expansion[4] [30]. However, these results do not directly apply to the digraphs we consider here. To prove the correctness of our second algorithm, we derive bounds on the probability of a random walk on a directed graph eventually reaching a target set T as a function of the *cut-size* of T, i.e. the number of edges that need to be removed to disconnect T from an initial set of vertices. We believe that our results on random walks are of independent interest and may help design provably (time and space) efficient algorithms for verifying safety properties [10] based on random walks. We

[2] There are however important differences between the problem considered here and the property testing setting. First, we are not solving a decision problem. Second, we measure the absolute Hamming distance, as opposed to the distance relative to the size of the input. And finally, we measure complexity in terms of the length of the test suite generated and not the sample complexity.

[3] In particular, T is reachable only via a particular sequence of edges (called the "combination") of length $|X| - |T|$, which can be as large as $\Omega(|X|)$

[4] The *expansion* of a digraph $G = (V, E)$ is the minimum value of $\frac{1}{|S|}\min(|N^+(S)|, |N^-(S)|)$ taken over all subsets S of V for which $0 < |S| \leq |V|/2$, where $N^+(S) = \{v \in V \setminus S \mid (u, v) \in E \text{ and } u \in S\}$ and $N^-(S) = \{v \in V \setminus S \mid (v, u) \in E \text{ and } u \in S\}$

also present lower bounds for the problems considered in this paper. Our lower bounds demonstrate that our algorithms for constructing tests are in fact very close to being optimal.

The rest of the paper is organized as follows. Section 2 introduces some concepts and notation used in the rest of the paper. Our results on random walks are presented in Section 3; this section can skipped by readers only interested in the fault detection problem. Section 4 describes the setting for the fault detection experiment and formally defines approximate checking sequences. Our randomized algorithms for constructing tests for detecting a single faulty machine are presented in Section 5; these algorithms are used in Section 7 to construct approximate checking sequences. Lower bounds are outlined in Section 6. Finally in Section 8 we discuss the upper and lower bounds presented in this paper, and determine bounds on Δ and r when the checking sequence can be polynomially long. Due to space limitation, some of the proofs are omitted.

2 Preliminaries

Directed Graphs and Walks. A *labeled directed graph* (or digraph) $G = (V, E, \Sigma)$ is a directed graph with vertex set V, label set Σ and edge function $E : V \times \Sigma \to V$. We use the notation $u \xrightarrow{\sigma}_G v$ to denote $E(u, \sigma) = v$. When the digraph G is clear from the context, we abbreviate this to $u \xrightarrow{\sigma} v$.

A pair $p = (t_1, \sigma_1\sigma_2 \ldots \sigma_k) \in V \times \Sigma^k$ is called a *path* of length k in $G = (V, E, \Sigma)$ if there is a sequence $((t_1, \sigma_1), (t_2, \sigma_2), \ldots, (t_k, \sigma_k)) \in (V \times \Sigma)^k$ corresponding to p such that $t_i \xrightarrow{\sigma_i} t_{i+1}$ for every $i = 1, \ldots, k-1$, and $t_i \neq t_j$ for every $i \neq j$. We say that t_i is the i-th vertex in p and (t_i, σ_i) is the i-th edge in P. We say that two paths $(t_1, \sigma_1 \ldots \sigma_k)$ and $(t'_1, \sigma'_1 \ldots \sigma'_l)$ are *edge-disjoint* if there is no pair (i, j) such that $(t_i, \sigma_i) = (t'_j, \sigma'_j)$.

A *random walk* in $G = (V, E, \Sigma)$ of length k starting from a set $U \subseteq V$ is a pair $(t_1, \sigma_1 \ldots \sigma_k) \in V \times \Sigma^k$ where t_1 is chosen uniformly at random from U, and for each $i = 1, \ldots, k$, σ_i is chosen uniformly at random from Σ. There is a unique sequence $((t_1, \sigma_1), \ldots, (t_k, \sigma_k)) \in (V \times \Sigma)^k$ corresponding to every random walk $R = (t_1, \sigma_1 \ldots \sigma_k)$, where $t_i \xrightarrow{\sigma_i} t_{i+1}$ for every $i = 1, \ldots, k-1$. We say that t_i is the i-th vertex in R and (t_i, σ_i) is the i-th edge in R.

Let $U \subseteq V$ and $X \subseteq V \times \Sigma$. A set $C \subseteq V \times \Sigma$ is called a (U, X)-*cut* in $G = (V, E, \Sigma)$ if, for every path $(t_1, \sigma_1 \ldots \sigma_k)$ in G such that $t_1 \in U$ and $(t_k, \sigma_k) \in X$, $(t_i, \sigma_i) \in C$ for some $1 \leq i \leq k$. We say that C is a *minimum* (U, X)-cut if C is a (U, X)-cut and there is no (U, X)-cut of cardinality strictly less than $|C|$.

Finite State Machines. A (Σ, Ω)-FSM is a deterministic finite state Mealy machine $M = (V_M, \delta_M, \lambda_M)$ with input alphabet Σ ($|\Sigma| = d$) and finite output alphabet Ω where V_M is a finite set of states, $\delta_M : V_M \times \Sigma \to V_M$ is the transition function and $\lambda_M : V_M \times \Sigma \to \Omega$ is the output function.

REMARK 1. *If $M = (V_M, \delta_M, \lambda_M)$ is a (Σ, Ω)-FSM, then $G_M = (V_M, \delta_M, \Sigma)$ is a labeled digraph.*

REMARK 2. *The transition function δ_M and output function λ_M of a (Σ, Ω)-FSM M can be extended to a mapping from $V_M \times \Sigma^*$ to V_M and to Ω^* respectively in the usual way.*

We say that M is *strongly connected* if G_M is strongly connected, i.e., for every pair of states $s_i, s_j \in V_M$, $\delta_M(s_i, x) = s_j$ for some $x \in \Sigma^*$. We say that M is *reduced* if, for every pair of distinct states $s_i, s_j \in V_M$ ($i \neq j$) there is some $x \in \Sigma^*$ such that $\lambda_M(s_i, x) \neq \lambda_M(s_j, x)$.

A *separating family* for M is a collection of sets $\{Z_u \mid u \in V_M\}$ such that for every pair of distinct states $u, v \in V_M$ there is an input string α such that $\lambda_M(u, \alpha) \neq \lambda_M(v, \alpha)$, and α is a prefix of some sequence in Z_u and some sequence in Z_v.

PROPOSITION 2.1. (YANNAKAKIS-LEE [42]) *Every reduced (Σ, Ω)-FSM M has a separating family $\{Z_u\}$ such that for each $u \in V_M$, $|Z_u| \leq |V_M| - 1$ and each $\alpha \in Z_u$ has length at most $|V_M| - 1$.*

3 Random Walks on Digraphs

In this section, we present our technical results on random walks on directed graphs. Before stating and proving bounds on the probability of a random walk to reach a target set, we observe the existence of edge disjoint paths of a special kind based on graph cuts.

LEMMA 3.1. *Let $G = (V, E, \Sigma)$ be a labeled digraph. If $U \subseteq V$, $X \subseteq V \times \Sigma$ and C is a minimum (U, X)-cut in G, then there are $|C|$ edge-disjoint paths in G such that, for each such path $p = (t_1, \sigma_1 \ldots \sigma_k)$:*

1. *$t_1 \in U$ and $t_i \notin U \ \forall\ 2 \leq i \leq k$;*
2. *$(t_k, \sigma_k) \in X$ and $(t_i, \sigma_i) \notin X \ \forall\ 1 \leq i \leq k-1$.*

The proof is a straightforward consequence of the Maxflow-Mincut theorem [15] and is omitted. If C is a minimum (U, X)-cut in G, then the $|C|$ edge-disjoint paths in G guaranteed by Lemma 3.1 are called *cut-paths* in G from U to X. We are now ready to prove our main lemma about random walks in directed graphs.

LEMMA 3.2. *Let $G = (V, E, \Sigma)$ be a labeled digraph such that $|V| \leq n + \Delta$ and $|\Sigma| = d$. Further, suppose*

$U \subseteq V$, $|U| = n$ and $X \subseteq (V \setminus U) \times \Sigma$ such that every (U, X)-cut in G has size at least $r > 0$, where $r \leq d\min(n, \Delta)$. We say that r is "small" if r is $o(d)$. Otherwise, there is some constant $c > 1$ such that $r \geq \frac{d}{c}$. Let

$$\rho = \begin{cases} r & \text{if } r \text{ is small} \\ r(1 - \frac{1}{c}) & \text{otherwise} \end{cases}$$
$$\lambda = \begin{cases} 1 + \Delta & \text{if } r \text{ is small} \\ 1 + \frac{cd\Delta}{r} & \text{otherwise} \end{cases}$$

Let P be the probability that the last edge of a random walk R in G beginning from U of length l (where l is chosen uniformly at random from $\{2, \ldots, \lambda\}$) is in X. Then $P \geq \frac{1}{\lambda - 1} \frac{\rho}{nd} \left(\frac{\rho}{4d\Delta^2}\right)^{\lambda - 1}$.

Proof. We claim that there are at least ρ cut-paths each of length at most λ. This is clearly true when r is small, since the size of every (U, X)-cut in G is at least r and hence there are at least r $[= \rho]$ cut-paths, each of length at most $1 + \Delta$ $[= \lambda]$. Consider the case when r is not small. Let N be the number of cut-paths of length at least $2 + \frac{cd\Delta}{r}$. Since all cut-paths are edge disjoint, the total number of edges *other than the first* in all such cut-paths is at least $N(1 + \frac{cd\Delta}{r})$. Now $N(1 + \frac{cd\Delta}{r}) \leq d\Delta$ since every edge other than the first in a cut-path is of the form (t, σ) where $t \notin U$. Hence, there are at most $\frac{r}{c}$ cut-paths of length at least $2 + \frac{cd\Delta}{r}$. Therefore, in this case too, there are at least $(1 - \frac{1}{c})r$ $[= \rho]$ cut-paths each of length at most $1 + \frac{cd\Delta}{r}$ $[= \lambda]$. Suppose these cut-paths are $p_1, \ldots, p_\rho$.

Consider first the case when the length of each cut-path is exactly λ. For each $i = 0, 1, \ldots, \lambda - 1$, let L_i be the set of $(i+1)$-th vertices of these cut-paths (note that $L_0 \subseteq U$) and let P_i be the probability that each of the first $(i + 1)$ edges of the random walk R is an edge in some cut-path. By the definition of R, it follows that $P_0 \geq \frac{\rho}{nd}$. Further, it is possible to show that

$$P_1 \geq \left(\frac{\rho}{d}\right) \frac{P_0}{|L_1|}$$
$$P_i \geq \left(\frac{\rho}{d}\right) \frac{P_{i-1}}{|L_i||L_{i-1}|} \quad \forall\, 2 \leq i < \lambda$$

Due to space limitations, we omit the proof of this claim. Thus, we have

$$P_{\lambda - 1} \geq \frac{\rho}{nd} \left(\frac{\rho}{d}\right)^{\lambda - 1} \frac{1}{\prod_{i=1}^{\lambda - 1} |L_i|^2} \tag{3.1}$$

Now consider the general case. For every $i = 1, \ldots, \lambda - 1$, let L'_i be the set of $(i + 1)$-th vertices of $p_1, \ldots, p_\rho$. Note that $|L'_i| \leq \Delta$ for every i. For every $i = 1, 2, \ldots, \lambda - 1$, let n_i be the number of cut-paths among $p_1, \ldots, p_\rho$ of length at most i (note that $n_1 = 0$ since every cut-path has length at least 2). Let L''_i be a set of $\max(1, \frac{n_i}{d})$ new vertices. Note that $|L''_i| \leq \max(1, \frac{\rho}{d}) \leq \Delta$ for every i.

Let $L_0 = U$ and for every $i = 1, \ldots \lambda - 1$, let $L_i = L'_i \cup L''_i$. Let $L = \bigcup_{i=1}^{\lambda - 1} L_i$. We claim that there is a digraph $G' = (V', E', \Sigma)$ such that

1. $V' = V \cup L \cup \{s\}$, where s is a new vertex not in $V \cup L$
2. for every $p = p_1, \ldots, p_\rho$, if $p = (t_0, \sigma_0 \ldots \sigma_k)$ then there is a path $p' = (t_0, \sigma_0 \ldots \sigma_k \sigma_{k+1} \ldots \sigma_\lambda)$ in G' such that for every $i > k$, $t_i \in L''_i$, and for each $i \neq j$, p'_i and p'_j are edge-disjoint.
3. For every $(t, \sigma) \in (V \times \Sigma) \setminus X$, $E'(t, \sigma) = E(t, \sigma)$.

This is easy to see, since G' can be obtained from G by adding the new vertices (in the sets L''_i) and edges by "extending" the cut-paths $p_1, \ldots, p_\rho$ in the appropriate manner as shown in Figure 1. We call the paths $p'_1, \ldots, p'_\rho$ *extended cut-paths* in G'. Let $X' \subseteq V' \times \Sigma$ be the set of last edges of all extended cut-paths. Thus, there are ρ (extended) cut-paths from U to X' in G', each of length λ. Every extended cut-path p' in G' has a prefix p of length k $(2 \leq k \leq \lambda)$ that is a cut-path in G.

Let P' be the probability that each edge of a random walk in G' beginning from U of length λ is an edge of some extended cut-path in G'. Using the above inequality 3.1 and noting that $|L_i| \leq 2\Delta$ for every i, we have

$$P' = P_{\lambda - 1} \geq \frac{\rho}{nd} \left(\frac{\rho}{4d\Delta^2}\right)^{\lambda - 1}$$

The probability that the last edge of a prefix of length l (where l is chosen uniformly at random from $\{2, \ldots, \lambda\}$) of such a random walk is the last edge of some cut-path is $\frac{P'}{\lambda - 1}$. Clearly, $P \geq \frac{P'}{\lambda - 1}$, and hence the lemma holds. □

4 Conformance Testing of FSMs

We now describe the setting for conformance testing. We are given a specification machine S and a "black-box" implementation I. Both I and S are assumed to be FSMs with input alphabet Σ and output alphabet Ω. In order to test if I is equivalent to S (defined later) we make some assumptions:

1. S is a reduced, strongly connected (Σ, Ω)-FSM;
2. $V_S = \{s_1, s_2, \ldots, s_n\}$;
3. I does not change during the testing experiment;
4. I is a strongly connected (Σ, Ω)-FSM;
5. $|V_I| \leq n + \Delta$ for some $\Delta > 0$.

The assumption of strong connectivity of S is needed because otherwise the test may not be able to visit all the states of S and thereby guarantee the equivalence of I to S; typically this is not a serious constraint since specifications often have a reset transition. The requirement that S be reduced can be easily met by minimizing the

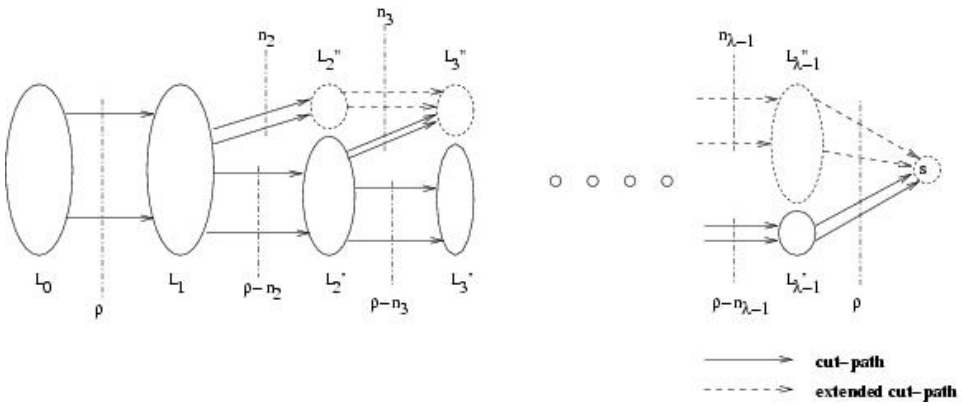

Figure 1: Construction of Extended Cut Paths

specification, if it is not already reduced. The need for assumption (3) is obvious. The requirement that I be strongly connected is a technical requirement based on ensuring that the homing sequence applied before the test leads I to the right state (see discussion below). If we are guaranteed that I starts in a known state, we can drop this requirement. Finally, in order to construct the checking sequence we need to know an upper bound on the number of states in I; without this knowledge constructing the checking sequence is impossible.

Under the assumptions discussed above, we would like to test if the implementation is equivalent to the specification or has multiple faults. We first define the notion of equivalence (trace) that we use.

DEFINITION 4.1. *Let $S = (V_S, \delta_S, \lambda_S)$ and $I = (V_I, \delta_I, \lambda_I)$ be two (Σ, Ω)-FSMs. A state $s \in V_S$ is said to be equivalent to $t \in V_I$ (denoted by $s \simeq t$) iff for every $x \in \Sigma^*$, $\lambda_S(s, x) = \lambda_I(t, x)$. S is said to be equivalent to I (denoted as $S \simeq I$) if and only if some state $s \in V_S$ is equivalent to some state $t \in V_I$.*

REMARK 3. *Observe that if S is strongly connected and $S \simeq I$ then for every $s \in V_S$, there is a $t \in V_I$ such that $s \simeq t$. A similar statement holds if I is strongly connected.*

Note that our notion of a specification does not include an initial state. If the specification includes a designated initial state then the notion of equivalence will require, in addition, that the initial state of the implementation be equivalent. Thus, in such a situation, the problem consists of two parts: (1) verify that the black box I is equivalent to S, and (2) verify that I starts in a state that is equivalent to the initial state of S, (say) s_1. The second problem, which is called the state verification problem, requires the existence of a sequence of inputs (called *unique input-output* (UIO) sequence) that distinguishes the starting state s_1 from all other states based on the outputs produced on the UIO sequence. UIO sequences need not exist for all reduced strongly connected FSMs, and so a test may not exist if our specification includes an initial state.

Thus, I may be equivalent to S but start in an unknown state. The first part of a fault detection experiment, therefore, applies what is called a *homing sequence* (see [24]) that brings I to a known state s_1, which is then the start state of the second part of the experiment. We require the implementation to be strongly connected to ensure that if I is equivalent to S then the homing sequence will work correctly. If I is known to start in a state equivalent to some state in the specification, then we do not require I to be strongly connected. Since homing sequences have been well understood for decades, the key challenge in designing a fault detection experiment is therefore to check if I is equivalent to S, and this is the problem we address in this paper.

DEFINITION 4.2. *We say that I is at* distance r *from S if $D \subseteq V_I \times \Sigma$ with $|D| = r$, is a set of smallest size for which there is a (Σ, Ω)-FSM $J = (V_J, \delta_J, \lambda_J)$ such that (1) $V_J = V_I$, (2) (δ_J, λ_J) and (δ_I, λ_I) differ only on the set D, and (3) S is equivalent to J.*

Note, that J need not be strongly connected in the above definition. Also observe that if I is at distance $r \geq 1$ from S, then S is not equivalent to I.

DEFINITION 4.3. *An (r, Δ)-approximate checking sequence for an n state (Σ, Ω)-FSM S is an input sequence w such that every (Σ, Ω)-FSM I with at most $n + \Delta$ states that is at distance r or more from S produces on input w a different output than that produced by S starting from state s_1, and every (Σ, Ω)-FSM I with at most $n + \Delta$ states that is equivalent to S produces on input w the same output that S produces starting from state s_1.*

Observe, that $(1,0)$-approximate checking sequence is usually simply called a *checking sequence* in the literature; such a sequence checks whether the implementation (with no additional states) is indeed isomorphic to the specification.

A *configuration* is a pair of states $(s,t) \in V_S \times V_I$; it is used to denote the states of S and I during some stage in the conformance testing experiment. Recall, that Proposition 2.1 says that every reduced FSM has a separating family. Let $Z = \{Z_i \mid i = 1,2,\ldots,n\}$ be a separating family for S (we use the shorthand notation Z_i to denote Z_{s_i}). We say that state $t \in V_I$ is *similar* to state $s_i \in V_S$ (which we denote by $t \sim_I s_i$) if for every $\alpha \in Z_i$, $\lambda_S(s_i,\alpha) = \lambda_I(t,\alpha)$. We define the *similarity function* $\pi_I : V_I \to V_S \cup \{\perp\}$ as $\pi_I(t) = s$ if $t \sim_I s$, and $\pi_I(t) = \perp$ otherwise. Note that $\pi_I(t)$ is well defined since a state $t \in V_I$ can be similar to at most one state $s \in V_S$. We define the *error set* X w.r.t. S and I as $X = X_1 \cup X_2 \cup X_3 \subseteq (V_I \times \Sigma)$ where X_1 and X_2 are the "transfer-errors":

$$X_1 = \{(t,\sigma) \mid \pi_I(\delta_I(t,\sigma)) = \perp\}$$
$$X_2 = \{(t,\sigma) \mid \pi_I(t) \neq \perp,\ \pi_I(\delta_I(t,\sigma)) \neq \delta_S(\pi_I(t),\sigma)\}$$

and X_3 is the set of "output-errors":

$$X_3 = \{(t,\sigma) \mid \pi_I(t) \neq \perp \text{ and } \lambda_I(t,\sigma) \neq \lambda_S(\pi_I(t),\sigma)\}$$

If $U \subseteq V_I$, a set $C \subseteq V_I \times \Sigma$ is called a (U,X)-*cut* in I if C is a (U,X)-cut in the labeled digraph $G_I = (V_I, \delta_I, \Sigma)$.

For every $i = 1,\ldots,n$ let T_i be some fixed (directed) breadth-first search (BFS) tree in $G_S = (V_S, \delta_S, \Sigma)$ with root s_i. For every $j = 1,\ldots,n$, let w_{ij} be the (unique) string such that (s_i, w_{ij}) is the unique path from s_i to s_j in T_i. Thus $\delta_S(s_i, w_{ij}) = s_j$ for every $1 \leq i,j, \leq n$. Note that w_{ii} is the empty string, for every i. We call the set $V(s_i,t) = \{t_j \mid t_j = \delta_I(t, w_{ij}), j = 1,\ldots,n\}$ the *vicinity* of $(s_i,t) \in V_S \times V_I$. We say that $V(s_i,t)$ is *safe* if $(V(s_i,t) \times \Sigma) \cap X = \emptyset$. Otherwise, $V(s_i,t)$ is called *unsafe*.

Remark 4. *For every prefix w of w_{ij}, there is some $j' \in \{1,\ldots,n\}$ such that $w = w_{ij'}$ and hence $\delta_I(t,w) = t_{j'} \in V(s_i,t)$.*

We now observe a crucial fact about safe vicinities, namely that, all the states in a safe vicinity are similar to corresponding states in the specification. This is the content of the next lemma which is proved before the main result of this section.

Lemma 4.1. *Let $(s_i,t) \in V_S \times V_I$ such that $\pi_I(t) = s_i$ and $V(s_i,t) = \{t_j \mid j = 1,\ldots,n\}$ is safe. Then for every $j = 1,\ldots,n$, $\pi_I(t_j) = s_j$.*

Proof. The proof is by induction on $|w_{ij}|$, the length of w_{ij} (recall that $\delta_I(t, w_{ij}) = t_j$). The base case, when $|w_{ij}| = 0$, is trivially true since in this case $j = i$ and it is given that $\pi_I(t_i) = s_i$.

Suppose $w_{ij} = w\sigma$ for some $w \in \Sigma^*$ and $\sigma \in \Sigma$. By Remark 4, there is some $j' \in \{1,\ldots,n\}$ such that $w = w_{ij'}$ and $\delta_I(t,w) = t_{j'}$. By the inductive hypothesis, $\pi_I(t_{j'}) = s_{j'}$. Since $V(s_i,t)$ is safe, it follows that $\pi_I(\delta_I(t_{j'},\sigma)) = \delta_S(s_{j'},\sigma)$, i.e. $\pi_I(t_j) = s_j$. The proof is now complete by induction. □

Cuts in the graph G_I and the distance of I from S can be related, and cuts play a crucial role in our algorithms. This relationship is formally stated in the next lemma, which is the main result of this section.

Lemma 4.2. *Suppose I is at distance r from S and $(s_i,t_i) \in V_S \times V_I$ is a configuration such that $\pi_I(t_i) = s_i$ and $V(s_i,t_i)$ is safe. Then, for every $(V(s_i,t_i),X)$-cut C in I, $|C| \geq r$.*

Proof. $V(s_i,t_i) = \{t_j \mid t_j = \delta_I(t_i, w_{ij}), j = 1,\ldots,n\}$. Suppose that there is a $(V(s_i,t_i),X)$-cut C in I such that $|C| < r$. We obtain a contradiction by showing that there is a (Σ,Ω)-FSM J at distance at most $|C|$ from I that is equivalent to S. Let $V_J = V_I$. Define $\delta_J : V_J \times \Sigma \to V_J$ and $\lambda_J : V_J \times \Sigma \to \Omega$ as follows:

$$\delta_J(t,\sigma) = \begin{cases} \delta_I(t,\sigma) \text{ if } (t,\sigma) \notin C \text{ or } \pi_I(t) = \perp \\ t_j \text{ otherwise, where } \delta_S(\pi_I(t),\sigma) = s_j \end{cases}$$
$$\lambda_J(t,\sigma) = \begin{cases} \lambda_I(t,\sigma) \text{ if } (t,\sigma) \notin C \text{ or } \pi_I(t) = \perp \\ \lambda_S(\pi_I(t),\sigma) \text{ otherwise} \end{cases}$$

Let $J = (V_J, \delta_J, \lambda_J)$. We prove that J is equivalent to S by showing that $t_i \simeq s_i$. In particular, we claim that for every $x \in \Sigma^*$, $\pi_I(\delta_J(t_i,x)) = \delta_S(s_i,x)$, and $\lambda_J(t_i,x) = \lambda_S(s_i,x)$. The proof is by induction on $|x|$, the length of x.

The base case, when $|x| = 0$, is trivially true since $\pi_I(t_i) = s_i$. Suppose $x = w\sigma$ for some $w \in \Sigma^*$ and $\sigma \in \Sigma$. Let $t = \delta_J(t_i,w)$ and $s = \delta_S(s_i,w)$. By the inductive hypothesis, we have $\pi_I(t) = s$ and $\lambda_J(t_i,w) = \lambda_S(s_i,w)$. Thus, to prove the inductive step, we need to show that $\pi_I(\delta_J(t,\sigma)) = \delta_S(s,\sigma)$ and $\lambda_J(t,\sigma) = \lambda_S(s,\sigma)$. There are two cases:

Case $(t,\sigma) \in C$**:** Since $\pi_I(t) = s \neq \perp$, we have $\delta_J(t,\sigma) = t_j$, where j is such that $\delta_S(s,\sigma) = s_j$ and $\lambda_J(t,\sigma) = \lambda_S(s,\sigma)$. By Lemma 4.1, $\pi_I(t_j) = s_j$ and hence $\pi_I(\delta_J(t,\sigma)) = \delta_S(s,\sigma)$ and $\lambda_J(t,\sigma) = \lambda_S(s,\sigma)$.

Case $(t,\sigma) \notin C$**:** We first show that $(t,\sigma) \notin X$. It suffices to show that for every $t_j \in V(s_i,t_i)$ and every $y \in \Sigma^*$, the path (t_j,y) in J does not contain an edge in X. Suppose, on the contrary, that there is a path $p = (t_j, \sigma_1 \ldots \sigma_k)$ in J which corresponds to the sequence $((u_1,\sigma_1),\ldots,(u_k,\sigma_k)) \in (V_J \times \Sigma)^k$ such that:

1. $u_1 = t_j \in V(s_i, t_i)$ and $u_m \notin V(s_i, t_i)\ \forall\ 2 \leq m \leq k$

2. $(u_k, \sigma_k) \in X$ and $(u_m, \sigma_m) \notin X\ \forall\ 1 \leq m < k$

We show by induction that for all $1 \leq m \leq k$, $\pi_I(u_m) \neq \perp$ and $(u_{m-1}, \sigma_{m-1}) \notin C$ (when $m > 1$). The base case ($m = 1$) is true since $\pi_I(u_1) = \pi_I(t_j) = s_j$ by Lemma 4.1. For the inductive step, suppose that $\pi_I(u_m) \neq \perp$. Since $u_{m+1} \notin V(s_i, t_i)$, it follows from the definition of δ_J that $(u_m, \sigma_m) \notin C$ and hence $\delta_J(u_m, \sigma_m) = \delta_I(u_m, \sigma_m)$. Since $(u_m, \sigma_m) \notin X$, $\delta_I(u_m, \sigma_m) \neq \perp$. So, $u_{m+1} = \delta_J(u_m, \sigma_m) \neq \perp$.

Thus, the sequence $((u_1, \sigma_1), \ldots, (u_k, \sigma_k))$ in fact corresponds to a path $(u_1, \sigma_1 \ldots \sigma_k)$ in I such that $u_1 \in V(s_i, t_i)$, $(u_m, \sigma_m) \notin C$ for every $1 \leq m < k$ and $(u_k, \sigma_k) \in X$. But this is impossible, since C is a $(V(s_i, t_i), X)$-cut in I. Hence, $(t, \sigma) \notin X$.

Now, since $(t, \sigma) \notin C$, we have $\delta_J(t, \sigma) = \delta_I(t, \sigma)$. Since $(t, \sigma) \notin X$, we have $\pi_I(\delta_I(t, \sigma)) = \delta_S(s, \sigma)$ and $\lambda_I(t, \sigma) = \lambda_S(s, \sigma)$. This proves the inductive step.

Hence J is equivalent to S. But this implies that I is at distance at most $|C| < r$ from S, a contradiction. Hence, $|C| \geq r$. □

COROLLARY 4.1. *Suppose I is at distance r from S and $(s_i, t_i) \in V_S \times V_I$ is a configuration such that $\pi_I(t_i) = s_i$ and $V(s_i, t_i)$ is safe. Then $|X| \geq r$ and $|V(s_i, t_i) \times \Sigma| \geq r$*

Proof. Clearly, the sets X and $(V(s_i, t_i) \times \Sigma)$ are both $(V(s_i, t_i), X)$-cuts in I, so Lemma 4.2 applies. □

5 Testing One Black Box

Suppose we are given a specific black box implementation machine I which we want test for conformance with specification S. If I is subjected to a deterministic experiment, then unless the test sequence x is a (r, Δ)-approximate checking sequence, we cannot say I has less than r errors if I passes the test. This is because if x misses even a single FSM with more than r faults, I could potentially be this FSM. Thus, for deterministic algorithms, testing a single implementation I does not differ from testing all possible machines. However, if we allow the fault detection experiment to be randomized then testing a single machine to achieve a certain level of confidence may require a shorter test. The algorithms we present in this section produce tests for a single black box. These constructions will be later used in Section 7 to obtain (r, Δ)-approximate checking sequences. In this section we first obtain an algorithm for the case when r is large, i.e., $r > d\min(n, \Delta)$, where $d = |\Sigma|$, and then consider the case when $r \leq d\min(n, \Delta)$.

5.1 Algorithm for $r > d\min(n, \Delta)$

The algorithm for constructing the (r, Δ)-approximate checking sequence is given in Figure 2. The algorithm is essentially similar to the one proposed by Yannakakis and Lee [42] and simply tests transitions in the specification at random.

LEMMA 5.1. *If I has at most $n + \Delta$ states and is at distance $r > d\min(n, \Delta)$ from S then, in each iteration of algorithm in Figure 2, the probability that the outputs of S and I differ is at least $\frac{1}{2dnz}$, where $z = \max_{1 \leq i \leq n} |Z_i|$*

Proof. Let the configuration at the start of any iteration be (s_i, t). There are two possible cases:
Case $\pi_I(t) \neq s_i$: In this case test 1 can detect an error, and the probability that an error is discovered in the current iteration is at least $\frac{1}{2}\frac{1}{|Z_i|} \geq \frac{1}{2z}$.
Case $\pi_I(t) = s_i$: We claim that $V(s_i, t)$ is unsafe. Suppose, for the sake of contradiction, that $V(s_i, t)$ is safe. We have $|V(s_i, t)| = n$. By Corollary 4.1, $|V(s_i, t) \times \Sigma| = dn \geq r$. Now if $n \leq \Delta$, we obtain a contradiction since $r > d\min(n, \Delta)$. Therefore suppose that $\Delta < n$ and hence $r > d\Delta$. By Corollary 4.1, we have $|X| \geq r > d\Delta$. Also, the sets $(V(s_i, t) \times \Sigma)$ and X are disjoint, since $V(s_i, t)$ is safe by assumption. Hence, $|V_I \times \Sigma| \geq |V(s_i, t) \times \Sigma| + |X| > nd + d\Delta = (n + \Delta)d$. Hence, $|V_I| > n + \Delta$, which contradicts the fact that I has at most $n + \Delta$ states.

Hence, $V(s_i, t)$ is unsafe. In this case test 2 can detect an error, and the probability that an error is discovered in the current iteration is at least $\frac{1}{2}\frac{1}{ndz}$. Hence, the lemma holds. □

THEOREM 5.1. *Let I be a FSM with at most $n + \Delta$ states and which is at distance at least r from S. For any $\epsilon > 0$, after $k = 2\log(\frac{1}{\epsilon})dnz$ iterations, the probability that the algorithm in Figure 2 detects that I is faulty is at least $1 - \epsilon$.*

The theorem follows from Lemma 5.1 and a standard application of Chernoff bounds [32].

5.2 Algorithm for $r \leq d\min(n, \Delta)$

We now consider the case when $r \leq d\min(n, \Delta)$, where $d = |\Sigma|$. When r is $o(d)$, let $\rho = r$ and $\lambda = 1 + \Delta$, and when there is a constant $c > 1$ such that $r \geq \frac{d}{c}$, let $\rho = r(1 - \frac{1}{c})$ and $\lambda = 1 + \frac{cd\Delta}{r}$. The algorithm for testing a single black-box is given in Figure 3.

LEMMA 5.2. *If I has at most $n + \Delta$ states and is at distance r from S then, in each iteration of the algorithm in Figure 3, the probability that the outputs of S and I differ is at least $\frac{1}{3dnz}\min\left\{1, \frac{\rho}{\lambda - 1}\left(\frac{\rho}{4d\Delta^2}\right)^{\lambda - 1}\right\}$, where $z = \max_{1 \leq i \leq n} |Z_i|$*

Input: A (Σ, Ω)-FSM S initially in state s_1, a separating family $\{Z_i\}$ of sequences for the states of S, and a black box (Σ, Ω)-FSM I.
Output: A random test sequence to be applied to S and I.
Method: Repeat k times: Apply one of the following 2 tests uniformly at random (u.a.r.)
Test 1: If S is currently in state s_i, choose a separating sequence x from Z_i u.a.r. and apply it to S and I.
Test 2: If S is currently in state s_i, choose $s_j \in V_S$ u.a.r. and $\sigma \in \Sigma$ u.a.r. If $\delta_S(s_j, \sigma) = s_{j'}$, choose a separating sequence x from $Z_{j'}$ u.a.r. Apply the sequence $w_{ij}\sigma x$ to S and I.

Figure 2: Algorithm for testing a specific Black-Box when $r > d\min(n, \Delta)$

Input: A (Σ, Ω)-FSM S initially in state s_1, a separating family $\{Z_i\}$ of sequences for the states of S, and a black box (Σ, Ω)-FSM I.
Output: A random test sequence to be applied to S and I.
Method: Repeat k times: Apply one of the following 3 tests uniformly at random (u.a.r.)
Test 1: If S is currently in state s_i, choose a separating sequence x from Z_i u.a.r. and apply it to S and I.
Test 2: If S is currently in state s_i, choose $s_j \in V_S$ u.a.r. and $\sigma \in \Sigma$ u.a.r. If $\delta_S(s_j, \sigma) = s_{j'}$, choose a separating sequence x from $Z_{j'}$ u.a.r. Apply the sequence $w_{ij}\sigma x$ to S and I.
Test 3: If S is currently in state s_i, choose $s_j \in V_S$ u.a.r., $l \in \{2, \ldots, \lambda\}$ u.a.r. and $w \in \Sigma^l$ u.a.r. If $\delta_S(s_j, w) = s_{j'}$, choose a separating sequence x from $Z_{j'}$ u.a.r. Apply the sequence $w_{ij}wx$ to S and I.

Figure 3: Algorithm for testing a specific black-box when $r \leq d\min(n, \Delta)$

Proof. Let the configuration at the start of any iteration be (s_i, t). There are three possible cases:
Case $\pi_I(t) \neq s_i$: In this case test 1 can detect an error, and the probability that an error is discovered in the current iteration is at least $\frac{1}{3}\frac{1}{|Z_i|} \geq \frac{1}{3z}$
Case $V(s_i, t)$ is not safe: Then, there is a state s_j (possibly $s_j = s_i$) such that for some $\sigma \in \Sigma$, $\pi_I(\delta_I(t, w_{ij}\sigma)) \neq \delta_S(s_i, w_{ij}\sigma)$ or $\lambda_I(t, w_{ij}\sigma) \neq \lambda_S(s_i, w_{ij}\sigma)$. In this case test 2 can detect an error, and the probability that an error is discovered in the current iteration is at least $\frac{1}{3}\frac{1}{ndz}$
Case $\pi_I(t) = s_i$ and $V(s_i, t)$ is safe: Note that choosing w_{ij} and w as described corresponds to a random walk of length $|w|$ in G_I starting from $V(s_i, t)$. Since $|V(s_i, t)| = n$ and the size of the $(V(s_i, t), X)$-cut in I (and hence in G_I) is at least r, it follows from Lemma 3.2, the probability that test 3 discovers an error is at least $\frac{1}{3}\frac{1}{z}\frac{\rho}{nd(\lambda-1)}\left(\frac{\rho}{4d\Delta^2}\right)^{\lambda-1}$.
Hence, the lemma holds. □

THEOREM 5.2. *Let I be a FSM with at most $n + \Delta$ states and which is at distance at least r from S. For any $\epsilon > 0$, the algorithm in Figure 3 detects that I is faulty with probability at least $1 - \epsilon$ after $k = 3dnz\log(\frac{1}{\epsilon})\max\left\{1, \frac{\lambda-1}{\rho}\left(\frac{4d\Delta^2}{\rho}\right)^{\lambda-1}\right\}$ iterations.*

The theorem follows from the observations in Lemma 5.2. The length of the test sequence output by the algorithm is $O\left(dnz(n+\Delta)\max\left\{1, \frac{\Delta}{r}2^{O(\Delta\log\frac{d\Delta^2}{r})}\right\}\right)$, when r is $o(d)$ and $O\left(dnz(n+\frac{d\Delta}{r})\max\left\{1, \frac{1}{r\Delta}2^{O(\frac{d\Delta}{r}\log\frac{d\Delta^2}{r})}\right\}\right)$, otherwise.

6 Lower Bounds

In this section, we present lower bounds for the problem of checking if a single black box conforms to the specification or has multiple faults.

THEOREM 6.1. *For every n, $d > 5$, there is a specification FSM S with d inputs and n states such that for every randomized test of length $L(n, \Delta, r)$ there is FSM I with at most $n + \Delta$ states at distance at least r from S that will pass the test with probability at least $1 - \epsilon$. The value of $L(n, \Delta, r)$ for different values of n, Δ and r are given as follows.*

1. *If $\Delta = 1$ and $r > d\Delta$, $L(n, \Delta, r) = \epsilon\frac{(d-5)(n+4)n^2}{32}$*
2. *If there is a constant $c > 1$ such that $\frac{d-3}{c} \leq r \leq \frac{(d-3)\Delta}{c}$, then $L(n, \Delta, r) = \epsilon(n-1)\frac{(d-3)\Delta}{cr}c^{1+\frac{(d-3)\Delta}{cr}}$*
3. *If $r = o(d)$ then $L(n, \Delta, r) = \epsilon(n-1)\Delta\left(\frac{d-3}{r}\right)^{1+\Delta}$*

The lower bounds are proved by generalizing the "combinatorial lock" construction used by Vasilevskii [41] to construct the specification machine and a family of implementation machines that defeat short tests. The formal constructions and their proofs of correctness are omitted due to space limitations.

7 Checking sequences

By running our algorithms for more iterations, we can obtain a randomized construction of an (r, Δ)-approximate checking sequence. Let N be the number of faulty machines, i.e. the number of (Σ, Ω)-FSMs with at most $n+\Delta$ states that are not equivalent to S. Clearly N is at most $\Delta\left(|\Omega|(n+\Delta)\right)^{d(n+\Delta)}$, since this is the total number of (Σ, Ω)-FSMs with at most $n+\Delta$ states. Note that $|\Omega| \leq d(n+\Delta)$ since this is the maximum number of transitions possible in any (Σ, Ω)-FSM with at most $n+\Delta$ states.

For any $\theta > 0$, if we let $\epsilon = \theta/N$ in Theorem 5.1 and Theorem 5.2, the probability that the resulting sequence detects *all* faulty machines is at least $1-\theta$, yielding an (r, Δ)-approximate checking sequence. We thus obtain the following theorems:

THEOREM 7.1. *When $r > d\min(n, \Delta)$, for any $\theta > 0$, taking $k = O\left(d^2 nz(n+\Delta)\log(\frac{|\Omega|(n+\Delta)}{\theta})\right)$, the algorithm in Figure 2 generates a sequence of length $O\left(d^2 n^2 z(n+\Delta)\log(\frac{|\Omega|(n+\Delta)}{\theta})\right)$ that is a (r, Δ)-approximate checking sequence with probability at least $1-\theta$.*

THEOREM 7.2. *When $r \leq d\min(n, \Delta)$, for any $\theta > 0$ the algorithm in Figure 3 generates a sequence of length $O\left(d^2 n(n+\Delta)^2 z \log\frac{|\Omega|(n+\Delta)}{\theta}\max\{1, m(d, \Delta, r)\}\right)$ that is a (r, Δ)-approximate checking sequence with probability at least $1-\theta$, where*

$$m(d, \Delta, r) = \begin{cases} \frac{\Delta}{r} 2^{O(\Delta \log \frac{d\Delta^2}{r})} & , \text{ if } r = o(d) \\ \frac{1}{r\Delta} 2^{O(\frac{d\Delta}{r} \log \frac{d\Delta^2}{r})} & , \text{ otherwise} \end{cases}$$

8 Discussion

In this section we determine bounds on the number of extra states in the implementation that can be tolerated, and still obtain polynomially long approximate checking sequences. Based on the lower bounds derived in Section 6, we determine the size of Δ that will force any checking sequence to be super-polynomial. The results from Section 7, help us obtain upper bounds on Δ for which our algorithms can construct polynomially long approximate checking sequence. In what follows, we consider the upper and lower bounds on Δ in three cases, depending on the value of r; $L(n, \Delta)$ is used to denote the length of a checking sequence.

Case $r > d\min(n, \Delta)$: Both the lower bound and the upper bound on the length of checking sequences are polynomial in $n+\Delta$ for every Δ.

Case r is $o(d)$: If Δ is $\omega\left(\frac{\log n}{\log(d/r)}\right)$, then the lower bound on $L(n, \Delta)$ is super-polynomial. We can guarantee a polynomial upper bound on $L(n, \Delta)$ when Δ is $O\left(\frac{\log n}{\log\log n + \log(d/r)}\right)$. Thus, our guarantee is best-possible when r is $O\left(\frac{d}{\log n}\right)$, and worse by a factor of at most $\log\log n$ otherwise.

Case $\frac{d}{c} \leq r \leq d\min(n, \Delta)$ for some constant $c > 1$: If Δ is $\omega\left(\frac{r}{d}\log n\right)$, then the lower bound on $L(n, \Delta)$ is super-polynomial. We can guarantee a polynomial upper bound on $L(n, \Delta)$ when Δ is $O\left(\frac{r}{d}\frac{\log n}{\log\log n + \log(r/d)}\right)$. Thus, our guarantee is worse than best-possible by at least a factor of $\log\log n$ and at most a factor of $\log n$ (since $r \leq d\min(n, \Delta)$).

These observations demonstrate that our upper bounds and lower bounds are only separated in the case when r is small, and even in this case the separation is at most by a factor of $\log n$. Trying to close this gap is an interesting open problem.

Acknowledgement

We would like to thank Sariel Har-Peled for his key idea of extended cut-paths in the proof of Lemma 3.2.

References

[1] R. Aleliunas, R. Karp, R. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of the maze problems. *Proc. of the Twentieth Annual Symp. on Foundations of Computer Science*, pages 218–223, 1979.

[2] N. Alon. Testing Subgraphs in Large Graphs. In *Proceedings of the IEEE Sysmposium on the Foundations of Computer Science*, 2001.

[3] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. In *Proceedings of the IEEE Sysmposium on the Foundations of Computer Science*, pages 656–666, 1999.

[4] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. In *Proceedings of the IEEE Sysmposium on the Foundations of Computer Science*, pages 645–655, 1999.

[5] N. Alon and A. Shapira. Testing Subgraphs in Directed Graphs. In *Proceedings of the ACM Symposium on Theory of Computing*, 2003.

[6] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic

machines. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 363–372, 1995.

[7] M. Bender and D. Ron. Testing acyclicity of directed graphs in sublinear time. In *Proceedings of the International Colloquim on Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 809–820. Springer, 2000.

[8] H. Chockler and O. Kupferman. ω-Regular Languages are testable with a constant number of queries. In *Proceedings of the Workshop on Randomization and Approximation Techniques (RANDOM)*, volume 2483 of *Lecture Notes in Computer Science*, pages 26–28. Springer, 2002.

[9] T. Chow. Testing software design modeled by finite-state machines. *IEEE Trans. Softw. Eng.*, 4(3):178–187, 1978.

[10] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[11] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal of Computing*, 19(3):500–510, 1990.

[12] J. A. Fill. Eigenvalue bounds for nonreversible Markov chains. *Annals of Applied Probability*, 1:62–87, 1991.

[13] E. Fischer, E. Lehman, I. Newman, S. R. R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the ACM Symposium on Theory of Computing*, 2002.

[14] E. Fischer and I. Newman. Testing of matrix properties. In *Proceedings of the ACM Symposium on Theory of Computing*, 2001.

[15] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[16] A. Friedman and P. Menon. *Fault Detection in Digital Circuits*. PrenticeHall, Inc., Englewood Cliffs, New Jersey, 1971.

[17] A. Gill. *Introduction to the Theory of Finite-state Machines*. McGraw-Hill, New York, 1962.

[18] S. M. Gobershtein. Check words for the states of a finite automaton. *Kibernetika*, 1:46–49, 1974.

[19] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998.

[20] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 406–415, 1997.

[21] F. Hennie. Fault detection experiments for sequential circuits. In *Proc. 5th Ann. Symp. Switch. Theory and Logical Design*. Princeton, NJ, 1964.

[22] G. J. Holzmann. *Design and Validation of Protocols*. Prentice-Hall Englewood Cliffs, NJ, 1990.

[23] E. P. Hsieh. Checking experiments for sequential machines. *IEEE Trans. Comput.*, C-20(10):1152–1166, 1971.

[24] Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, 1978.

[25] I. Koufareva, A. Petrenko, and N. Yevtushenko. Test generation driven by user-defined fault models, 1999.

[26] G. F. Lawler and A. D. Sokal. Bounds on the l^2 spectrum of Markov chains. *Transactions of the American Mathematical Society*, 309:557–580, 1988.

[27] D. Lee and M. Yannakakis. Testing finite state machines: State identification and verification. *IEEE Transactions on Computers*, 43(5), 1994.

[28] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - A survey. In *Proceedings of the IEEE*, volume 84, pages 1090–1126, 1996.

[29] R. Linn and M. Üyar. *Conformance testing methodologies and architectures for OSI protocols*. IEEE Computer Society Press, 1995.

[30] M. Mihail. Conductance and convergence of markov chains - a combinatorial treatment of expanders. In *Proc. of 30th Symp. on Foundations of Computer Science*, pages 526–531, 1989.

[31] E. F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, pages 129–153, Princeton University Press, Princeton, NJ, 1956.

[32] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[33] S. Naito and M. Tsunoyama. Fault detection for sequential machines by transition tours. In *Proc. IEEE Fault Tolerance Comput. Symp.*, pages 238–243. IEEE Comput. Soc. Press, Washington D.C, 1981.

[34] I. Newman. Testing functions that have small width branching programs. In *Proceedings of the IEEE Sysmposium on the Foundations of Computer Science*, 2000.

[35] A. Petrenko and N. Yevtushenko. On test derivation from partial specifications. In *FORTE*, pages 85–102, 2000.

[36] D. Ron. Property Testing. In S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, editors, *Handbook of Randomized Computing*. Kluwer Press, 2001.

[37] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing*, 25(2):252–271, 1996.

[38] K. K. Sabnani and A. T. Dahbura. A protocol test generation procedure. *Comput. Networks ISDN Syst.*, 15(4):285–297, 1988.

[39] A. Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.

[40] M. N. Sokolovskii. Diagnostic experiments with automata. *Kibernetika*, 6:44–49, 1971.

[41] M. P. Vasilevskii. Failure diagnosis of automata. *Kibernetika*, 4:98–108, 1973.

[42] M. Yannakakis and D. Lee. Testing finite state machines: Fault detection. *Journal of Computer and System Sciences*, 50:209–227, 1995.