On the Random Walk Method for Protocol Testing

Milena Mihail
Bellcore

*Christos H. Papadimitriou**
University of California, San Diego

Abstract

An important method for testing large and complex protocols repeatedly generates and tests a part of the reachable state space by following a random walk; the main advantage of this method is that it has minimal memory requirements. We use the coupling technique from Markov chain theory to show that short trajectories of the random walk sample accurately the reachable state space of a nontrivial family of protocols, namely, the symmetric dyadic flip-flops. This is the first evidence that the random walk method is amenable to rigorous treatment.

Following West's original reasoning, efficient sampling of the reachable state space by random walk suffices to ensure effectiveness of testing. Is, however, efficient sampling of the random walk necessary for the effectiveness of the random walk method? In the context of Markov chain theory, "small cover time" can be thought of as a simpler justification for the effectiveness of testing by random walk; all symmetric (reversible) protocols possess the small cover time property.

Thus the conclusions of our work are that (i)the random walk method can be understood in the context of known Markov chain theory, and (ii)symmetry (reversibility) is a general protocol style that supports testing by random simulation.

1 Introduction

Testing complex protocols for conformance with their specifications is an ever-increasing technological challenge —see Holzmann's book for a detailed exposition [13]. According to Holzmann, traditional testing techniques based on full or controlled partial state exploration become very complex (perhaps prohibitively so) when the reachable state space becomes larger than, say, 10^8 [13] [4] [17]. For larger protocols, an alternative testing method is what could be called the *random walk method* (a.k.a. *random partial state exploration*, or *random simulation*) [13] [19] [18]: Beginning at the starting protocol state, a random applicable action of the protocol is chosen and applied, another random step is repeated at the resulting state, and so on, up to some sufficiently large and yet computationally feasible number of steps. At each step the next action is chosen among the applicable actions either according to a distribution that reflects the protocol's operation, or uniformly at random —the latter technique is good for exploring the less traveled paths in the state space. The main advantage of this random simulation method is that it has minimal space (memory) requirements: to simulate a step of the random walk we need to store only the current state and a description of the actions (which is a natural lower bound on memory for any state exploration process).

When and why is the random walk testing technique sound? Engineers argue that testing is effective if the random walk looses memory of its starting state after a relatively short trajectory, thus the final point of the walk simulates accurate observation of the system in steady state, or, preferably, fair sampling of the protocol's reachable state space. Under such *rapid mixing* conditions, errors are recovered in time proportional to their probability of occurrence in stationarity, and the whole reachable state space can be potentially visited in time roughly proportional to its size (which is again a natural lower bound on time for any exhaustive search). In his influential paper introducing and advocating the random walk technique [19], West argued that this is indeed the behavior of certain validation experiments over the OSI Session Layer, and, in the formal sense, in the case of a very simple hypothetical *totally decoupled* system (a system without communication between its entities).

*Research supported by the National Science Foundation.

There are several mathematical tools at the disposal of the full and controlled partial state exploration approach to protocol testing, such as temporal logic algorithms for model verification (eg. see [20] [8] [9] for early references), symbolic methods, compression methods, projection methods, and techniques for testing and minimizing finite state machines (eg. see [15] [21] for recent references). In contrast, there is very little rigorous work on the random walk method —besides West's observation concerning the decoupled protocol. In Markov chain theory, it is now well understood that many common random walks *do not* converge in any satisfactory way —and a theory of those that do is emerging [10] [1] [2] [14] [16] [6] [11] [12]. In this context, West's example of a decoupled system is easily identified as the *hypercube,* the most simple case of a rapidly mixing random walk.

The aim therefore is to use Markov chain theory to separate those protocols which can be effectively tested by random walk from those which cannot. Naturally, circular characterizations such as "protocols whose reachable state space is an expander" are completely useless. For a proposed family of protocols to be interesting, it should be definable in terms of "local," "syntactic" properties of the protocols. Ideally, one would like syntactic guidelines, perhaps "styles" of protocol writing, that guarantee convergence of the associated random walk; for example, "reversibility of actions" could be such a property. A possible analogy here is the "structured programming" paradigm in software engineering, which guarantees positive debugging and maintainability properties of the resulting code. The question therefore is, *what are examples of broad families of protocols whose state space can betested by random walks?*

In this paper we identify a family of protocols, namely *systems of symmetric dyadic flip-flops* (SSDF's), and prove that the random walk always converges to the uniform distribution over the state space of such protocols in time polylogarithmic in the size of the state space (Theorem 3.1). Thus, by West's statistical arguments, this class can be tested by random walk. SSDF's are systems of communicating finite automata (see also Figure 1) with two states per entity (hence the name flip-flop); each action involves only two entities (hence the attribute "dyadic"); and *symmetry (reversibility),* that is, whenever an action that affects two coordinated transitions of two entities is present, the reverse action is also present. Although the expressibility of SSDF's is rather restricted, in the sense that their reachable state spaces have fairly regular characterizations, the transition graph over the reachable state space is highly irregular, and consequently our rapid mixing argument demonstrates nontrivial technique (and is a giant step beyond the mixing argument for the hypercube decoupled protocol, the only one heretofore analyzed in this way [19]). Therefore our result provides the first substantial evidence that the random walk method is amenable to formal Markov chain analysis and understanding.

The technique employed to prove our main rapid mixing result uses the *coupling* argument from Markov chain theory [1] [10] [5] [6]. To show that the random walk from some starting state converges within polynomial time to its stationary distribution, we imagine side-by-side two such walks, one from the starting state and one from an arbitrary state of the reachable state space, and coordinate their transitions so that each walk is simulated fairly, but in small expected time the two states coincide and thus the two walks are indistinguishable; since the second random walk can be assumed to start from the stationary distribution over all reachable states, fast convergence is established.

Can arguments for the effectiveness of the random walk method extend beyond the class of SSDF's? This question is twofold. Firstly, can rapid mixing arguments for the justification of the random walk method extend beyond SDFF's? Secondly, are rapid mixing conditions necessary for the random walk testing method to be effective?

For the first question, there are counter-examples showing that, if any of the conditions defining SDFF's fails, rapid mixing also fails [7]. But perhaps characterizations somewhat different to the features of SDFF's may reveal further rapid mixing cases (experiments suggest that "random" symmetric protocols mix rapidly, thus, capturing features of randomness that ensure rapid mixing might be a way to proceed).

For the second question, we point out that *small cover time* may be a simpler justification for the effectiveness of the random walk method; small cover time is a strictly weaker property than rapid mixing. The cover time of a graph is the expected time for a random walk to visit all the vertices of the graph, and it fits naturally in the context of state space exploration. All symmetric systems posses the small cover time property. Thus a possible heuristic for testing by random

simulation could be to design systems as reversible as possible, and/or impose artificial reversibility for simulation purposes.

2 Preliminaries

A *protocol* is a system of communicating finite automata, the *entities*. Each entity is driven by a set of *actions*, and for a ⟨state, action⟩ pair, the *transition function* determines (i)whether the action is applicable on the particular state (not all actions are applicable on all states), and if so, (ii)which is the new state after the action is applied. The system has *communication* in that an action may simultaneously involve several entities. When the system evolves as a whole, an action is applicable iff it is applicable for all entities that it involves. This determines the *transition function* of the *combined automaton*, with combined states consisting of vectors of size n (storing the state of each entity) and actions consisting of the union of actions over all entities. Finally there is a specified *starting combined state*, and a *reachable state space* consisting of combined states to which the initial state can be driven by a sequence of applicable actions; realize that the size N of the combined state space is exponential in n. The aim of *testing* is to explore the reachable state space of the reachable state space of the combined automaton. The aim of *efficient testing* is to explore the reachable state space using resources that match the natural lower bounds (up to polynomial factors, that is, time poly$N$ and space poly n, and for practical efficiency as close to linear as possible).

We shall also consider a protocol as a *Markov chain* (or *random walk on the underlying graph of the combined automaton*). Without loss of generality, we consider a slightly modified version of the natural random walk: Beginning at the starting combined state, at each step, some action is chosen uniformly at random among all actions of the system, and if the action is applicable to the current combined state, then, with probability 1/2, it is applied; in all other cases the combined state does not change. How can such a scheme support efficient testing? Firstly realize that it satisfies the memory requirement for efficiency: to simulate each step, we need to store only the current state and the set of actions. To argue about time requirements we need additional conditions.

A protocol is *symmetric* if it is it is symmetric as a combined automaton and as a Markov chain. This amounts to the protocol being *reversible* as a system: for each action a that drives

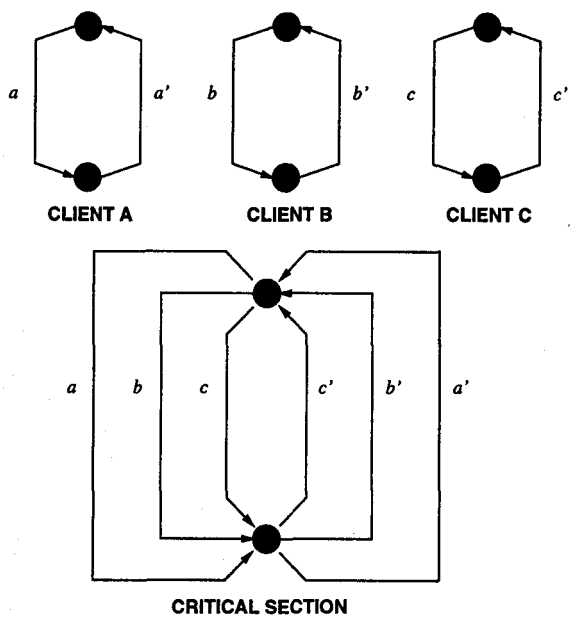


Figure 1: A system of Symmetric Dyadic Flip-Flops implementing Critical Section with Three Clients.

combined state $\vec{q}$ to combined state $\vec{q'}$, there exists a reverse action a^{-1} that drives $\vec{q'}$ to $\vec{q}$. Well known Markov chain theory suggests that the random walk on symmetric protocols *converges to the uniform distribution* over the whole reachable state space. Now for testing purposes, convergence to uniformity is quite favorable. We can use a point of the random walk *after convergence has (nearly) occurred* as a uniformly random point of the state space. For a reachable state space of size N, $O(N \log N)$ uniform samples, in expectation, generate the whole space (in addition, for any fraction ϵ of the reachable state space, a representative state is generated by ϵ^{-1} samples in expectation). But can we ensure *efficient convergence* to uniformity? We focus on a class of symmetric protocols that we call *systems of symmetric dyadic flip-flops* (SSDF's), and show that their convergence to uniformity is efficient, in the sense that it is ensured after poly $\log N$ steps (Theorem 3.1).

Three features determine symmetric dyadic flip-flops: They are symmetric in the way discussed above. They are flip-flops, meaning that all entities have two states; we can assume that these are 0 and 1. They are dyadic, meaning that each action involves at most two automata; without loss of generality we assume that each action involves exactly two automata. We impose the additional technical condition that there are no self-loops, that is actions that leave the state of some involved automaton unchanged (self loops are treated in [7]). Thus, there are two kinds of actions: the so called *bar* actions that change a pair of 0's to 1's and their inverses changing a pair of 1's to 0's, and the so called *cross* actions changing a pair of a 1 and a 0 to a pair of a 0 and a 1. Given a system of symmetric dyadic flip-flops, we define its *communication graph*, with one vertex for each automaton, and with an edge $[u, v]$ present if and only if there is an action (and its inverse) involving both u and v; again, without loss of generality, we assume that if there is a bar (resp. cross) action involving u and v, then there is no cross (resp. bar) action involving u and v. Now realize that the transitions of the protocol can be completely specified by its communication graph, once its edges have been labeled as bar or cross; we call this the *labeled communication graph*.

For each combined state $\vec{q}$ we also consider its *action communication graph*: if $[u, v]$ is a bar edge and the bits u and v are similar, or if $[u, v]$ is a cross edge and the bits u and v are opposite, then one of actions a and a^{-1} that correspond to $[u, v]$ is applicable on $\vec{q}$, thus we mark $[u, v]$ as *active*; otherwise we mark $[u, v]$ as *inactive*. Realize that if a state $\vec{q'}$ differs from a state $\vec{q}$ on exactly one bit, say v, then the action communication graphs of $\vec{q}$ and $\vec{q'}$ are identical on edges not incident to v, and exactly opposite on edges incident to u. In the next section, we shall repeatedly think of combined states in terms of their action communication graphs and furthermore, of transitions as follows (see also Figure 2): for some active edge $[u, v]$ with corresponding actions a and a^{-1} exactly one of which, say a, is applicable to $\vec{q}$, the effect of applying a on $\vec{q}$ can be thought of as activating the edge $[u, v]$ on the action communication graph of $\vec{q}$ thus changing bits u and v, and, in the resulting action communication graph, changing all edges incident to u or v from active to inactive and vice versa, except for $[u, v]$ itself which remains unchanged.

Clearly the labeled communication graph is an invariant of the protocol, while the action communication graph changes from state to state. However, the action graph inherits certain invariants of the labeled graph. We shall need several such invariants in the next section; here we start by demonstrating the simplest one in Lemma 2.1 below. In particular, we say that a *cycle of the labeled graph* has *odd parity* iff it contains an odd number of bar edges, and we say that a *cycle of the action graph of a specific state* has *odd parity* iff it contains an odd number of active edges (here we mean generalized cycle: a closed trail with arbitrary partial overlaps).

Lemma 2.1 (Cycle invariant) (See also Figure 2). *For any cycle of the communication graph, the parity of the cycle in action graphs is invariant over all combined states, and it is the same as the parity of the cycle in the labeled graph.*

PROOF. The parity of cycles in labeled and action graphs are trivially identical for the combined state $\vec{q} = 00\ldots0$, and changing the bits of $\vec{q}$ one at a time preserves the parity of cycles on action communication graphs.

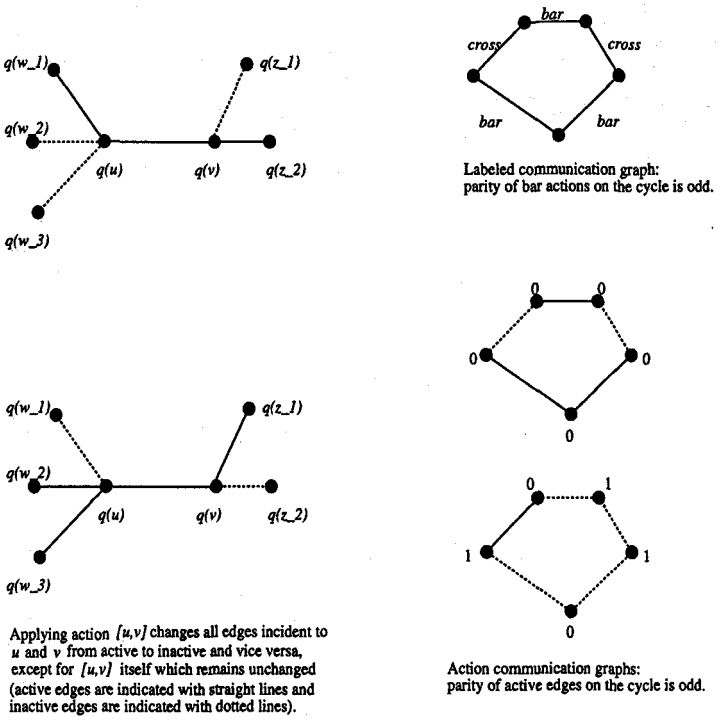


Figure 2: Transitions and cycle parities on action and labeled communication graphs.

3 SDFF's, Coupling, and Rapid Mixing

In this section we prove that SDFF's, viewed as Markov chains, posses the rapid mixing property, that is, the random walk is arbitrarily close to uniformity over N reachable states after poly $\log N$ number of steps. To establish rapid mixing we use the *coupling* method from Markov chain theory [1] [10] [5] [6]. Consider two random walks on the protocol, random walk $\vec{q}(t)$ starting from some starting state $\vec{q}(0)$, and random walk $\vec{q'}(t)$ starting from some arbitrary state $\vec{q'}(0)$ reachable from $\vec{q}(0)$; $\vec{q'}(t)$ can be thought of as representing the uniform stationary distribution. Intuitively, we must show that $\vec{q}(t)$ and $\vec{q'}(t)$ will soon be indistinguishable. To do this, we run $\vec{q}(t)$ and $\vec{q'}(t)$ side-by-side, by applying subtle coordination in their transitions. The coordination is such that both random walks still obey the appropriate transition probabilities, but within expected time poly n = poly $\log N$ the two walks converge to the same state. More formally, where the *coupling time* T is the first time that $\vec{q}(t)$ and $\vec{q'}(t)$ match: $\vec{q}(T) = \vec{q'}(T)$, and $E[T]$ is its worst case expectation (over all initial configurations $\vec{q}(0)$ and $\vec{q'}(0)$ that are reachable from each other), the *variation distance* of $\vec{q}(t)$ from stationarity can be bounded by (see [1] [10] [5] for a proof):

$$d(t) = \max_{Q} \left(\Pr[\vec{q}(t) \in Q] - \frac{|Q|}{N} \right) \leq 1/2^{\frac{t}{2E[T]}}$$

For the coupling, we keep pebbles on the vertices of the communication graph where $\vec{q'}(t)$ and $\vec{q'}(t)$ differ and measure progress in terms of the number of pebbles left. In particular, where $[u, v]$ is an edge of the communication graph and there are pebbles on vertices here $\vec{q'}(t)$ and $\vec{q'}(t)$ differ, realize that the following hold: (i)If there are pebbles on both u and v, then either $[u, v]$ is active for both $\vec{q'}(t)$ and $\vec{q'}(t)$, or $[u, v]$ is inactive for both $\vec{q'}(t)$ and $\vec{q'}(t)$. The case where $[u, v]$ is active for both processes is particularly favorable: if $[u, v]$ is activated on exactly one of the processes,

then this process will change bits u and v thus matching them with the other process, and the pebbles can be removed from vertices u and v. (ii)If there is a pebble on exactly one of u and v, say u, then $[u,v]$ is active on exactly one of $\vec{q}'(t)$ and $\vec{q}'(t)$ and inactive on the other. This case is indifferent: if $[u,v]$ is activated for the process for which it is active, then the two processes will match the bit u but will unmatch the bit v, thus the pebble should be moved from u to v. (iii)If there are pebbles on neither u nor v, then the two processes match on these two bits and, as far as $[u,v]$ is concerned, they can proceed in full coordination. The above three observations lead us to the following coupling (which is convenient to think of as it evolves on action communication graphs, in the way explained in Section 2):

Let $\vec{q}(t)$ and $\vec{q}'(t)$ be the states at time t;
Pick $[u,v]$ uniformly at random among all edges of the communication graph;
Toss a fair coin;

Case ia IF there are pebbles on both u and v and $[u,v]$ is active in both processes

THEN
- if heads then $[u,v]$ is activated on $\vec{q}(t)$ only;
- if tails then $[u,v]$ is activated on $\vec{q}'(t)$ only;
- in either case the pebbles are removed from both u and v;

Case ib IF there are pebbles on both u and v and $[u,v]$ is inactive in both processes
THEN nothing happens;

Case ii IF there is a pebble on u but not on v, thus $[u,v]$ is active for exactly one of the processes
THEN, if heads, $[u,v]$ is activated on the process for which it is active, and the pebble is moved from u to v;
ELSE nothing happens;

Case iii IF there are no pebbles on either u or v, thus $[u,v]$ is either active or inactive for both processes,
THEN, if heads and $[u,v]$ is active, $[u,v]$ is activated on both processes;
ELSE nothing happens;

Theorem 3.1 (Main Theorem) *For any SDFF with n entities and m actions, for any initial configurations $\vec{q}(0)$ and $\vec{q}'(0)$ that belong to the reachable state space, the expectation of the coupling time T (when all pebbles are removed and $\vec{q}(T) = \vec{q}'(T)$) is $E[T] = O(n^3m^2)$, and hence the variation distance decreases as $2^{-\Omega(t/n^3m^2)}$.*

PROOF (outline). We wish to establish that the favorable *Case ia* with two pebbles placed at the endpoints of an active edge occurs often enough. What causes difficulty in the pebble game is that evolution is determined by the action graphs, and these graphs change. We therefore go through a sequence of reductions to reformulate a pebble game on the time invariant labeled communication graph. By the end of Lemma 3.4 the pebble game is played only on the labeled communication graph.

Firstly, the pebble game can be reduced to one which starts with only two pebbles, where the first time that *Case ia* occurs the game is finished. Let τ be the time for *Case ia* to occur when we start with two pebbles, and let $E[\tau]$ be its worst case expectation. It can be argued (and is intuitive to understand) that

$$E[T] \leq nE[\tau] \tag{1}$$

Now for the two-pebble game, say a red and a blue, there may be many possibilities by which *Case ia* occurs; we focus on a single one. In Lemma 3.2 we isolate a "target" trail on the action communication graphs with the property that this trail always contains an odd number of active edges in both processes. In Lemma 3.3(i) we reduce removal of the pebbles to traversal of the target trail: when this odd trail is of length one, the pebbles are neighboring by an active edge and thus *Case ia* has occurred. However, straightforward traversal of the target trail is probabilistically very unlikely, thus in Lemmas 3.3(ii) and 3.4 we account for the likely case of repeated "circular" efforts until the target trail is finally traversed.

Lemma 3.2 (Target trail). *If $\vec{q}'$ is reachable from $\vec{q}$ and they differ in exactly bits u and v (the red and blue pebbles), then there is a "target" trail from u to v on the communication graph such*

that the trail contains an odd number of active edges in the action communication graphs of both $\vec{q}$ and $\vec{q'}$.

PROOF (outline). Any trail from u to v has the same number of active edges in both $\vec{q}$ and $\vec{q'}$ (all edges incident to neither u nor v are similar —and so is $[u,v]$, if present, and all edges incident to exactly one of u and v are opposite). Thus we need to find an odd active edge trail (*odd trail*, for short) only for $\vec{q}$.

If the action communication graph of $\vec{q}$ contains a cycle with an odd number of active edges (and by Lemma 2.1 the labeled graph contains and odd cycle), then an odd trail can be found trivially: start with a path from u to some vertex of the cycle, end with a path from some vertex of the cycle to v, and use the odd parity of the cycle to insert a trail that ensures correct parity of active edges. If all cycles in the action communication graph of $\vec{q}$ contain an even number of active edges, then it can be shown that all trails from u to v are odd (this reasoning is somewhat more detailed and is omitted here).

Note that the specific target trail constructed in this proof is of length at most m.

Remark: As a byproduct of Lemma 3.2 we obtain regular characterizations of the reachable state spaces. For protocols whose labeled graph contains an odd cycle, all vectors with the same parity of bits can be reached, and a similar (but somewhat finer) characterization holds for protocols whose labeled graph contains no odd cycle. Such observation draw limitations on the expressibility of SDDF's in terms of their reachable state spaces, however, it should be obvious that the transitions over these state spaces are highly irregular.

Lemma 3.3 *Let $u = v_0, v_1, \ldots, v$ be a target trail of Lemma 3.2. (i)If edge $[u, v_1]$, which is active in exactly one of $\vec{q}$ and $\vec{q'}$, is applied, thus moving the red pebble to v_1, then the suffix $v_1, \ldots, v$ is a target trail for the new configurations. (ii)If edge $[u, u']$, $u' \neq v_1$, which is also active in exactly one of $\vec{q}$ and $\vec{q'}$, is applied, thus moving the red pebble to u', then the concatenation $u', u, v_1, \ldots, v$ is a target trail for the new configurations.*

Lemma 3.4 *Let $u = v_0, v_1, \ldots, v$ be a target trail of Lemma 3.2. Let $v_i, \ldots, v_j = v_i$ be a cycle that has even number of bar edges in the labeled graph, and by Lemma 2.1, the cycle contains an even number of active edges in the action graphs of both $\vec{q}$ and $\vec{q'}$. Then $u = v_0, v_1 \ldots v_i, v_j, \ldots, v$ is also a target trail for $\vec{q}$ and $\vec{q'}$.*

The proofs of Lemmas 3.3 and 3.4 are straightforward and omitted.

We proceed to finish the proof of Main Theorem 3.1. In particular, we are now ready to formulate a time invariant pebble game.

Let $u = v_0, v_1, \ldots, v_l = v$ be a target trail for $\vec{q}(0)$ and $\vec{q'}(0)$. Lemmas 3.3 and 3.4 suggest that when, for some (so called "return") time τ_r, the red pebble returns on u' after having traveled a cycle $u = y_0, y_1 \neq v_1, \ldots, u$ that has an even number of bar edges in the labeled graph, and, for the same τ_r, the blue pebble returns on v after having traversed a cycle $v = z_0, z_1 \neq v_{l-1}, \ldots, v$, then $u = v_0, v_1, \ldots, v_l = v$ is still a target trail for $\vec{q}(\tau_r)$ and $\vec{q'}(\tau_r)$.

And if the first time that a pebble moves after τ_r, it is either the red pebble moving along $[u, v_1]$, or the blue pebble moving along $[v, v_{l-1}]$, then the target trail shortens by one. The probability that this happens is at least $1/n$, thus in expected n attempts it will indeed occur.

Now it follows that the expected time for a target trail of initial length l to become a single edge is

$$E[\tau] \leq l \cdot n \cdot E[\tau_r] \leq n^2 E[\tau_r] \tag{2}$$

For the return time τ_r, combining arguments of [3] and averaging principles (about odd and even cycles), it can be shown that

$$E[\tau_r] \leq O(m^2) \tag{3}$$

Now (1), (2), and (3) complete the proof of the Main Theorem 3.1.

4 More General Protocols and Cover Times

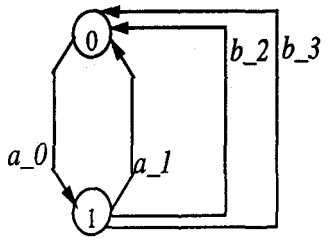
Three main restrictions define the class of SDFF's: symmetry, two states per enity, and two entities per action. In Figures 3 and 4 we outline examples where if any of the above conditions fail, rapid mixing fails (more extensive discussions are in [7]; the counter-like counter-examples were pointed out to us by Ernie Cohen). Thus, in the strict sense, rapid mixing cannot extend beyond SDFF's, unless we obtain some characterization somewhat different to the features involved in SDFF's.

On the positive side, experiments with random graphs indicate that random symmetric protocols mix rapidly [7]. Therefore, perhaps there are features of randomness that can be captured to establish rapid mixing beyond SDFF's (in graph theory, the process of isolating features of random graphs, and constructing explicit graphs that posses these features thus inheriting certain behaviors of random graphs, is well explored). For example, the combined automata and the communication graphs (hypergraphs) of random symmetric protocols have small diameter —a feature strongly violated by counter-like counter-examples (where not only wost case points are in long distances, but also average case points are in long distances).

However, is rapid mixing essential for the effectiveness of testing by random walk?

If the requirement of testing is that, for any fraction ϵ of the reachable state space of size N, we visit a representative element from this set in expected time roughly $\epsilon^{-1}\text{poly}\log N$ (which is the natural lower bound), then rapid mixing (which achieves efficient testing almost by definition) appears necessary.

If the requirement of testing is weakened to exhaustive state space exploration, then perhaps "small cover time" for the combined automaton is a simpler justification for effectiveness of testing.



A system of assymetric dyadic flip–flops that does not mix rapidly: Starting from the 00...0 combined state, it takes 2^n attempts in expectation for a 1 to propagate to the n–th bit. This is because, at each step, with probability 1/2, some "b" action will bring the system to its starting combined state.

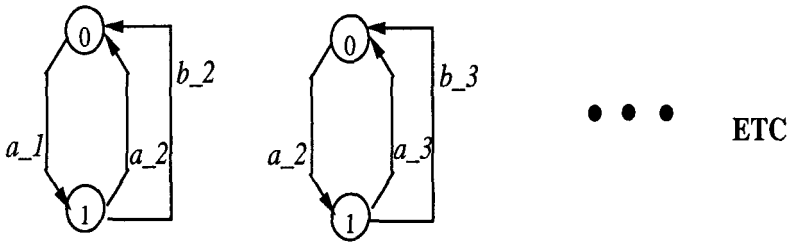


Figure 3: Assymmetric counter-example to rapid mixing.

For a random walk on a graph, the *cover time* is the time by which all vertices have been visited at least once. All symmetric (undirected) graphs are known to have cover times at most cubic in their sizes [3], and most graphs have cover times slightly bigger than linear. Thus all symmetric protocols posses the small cover-time property, and, in the sense of exhaustive search of the reachable state space, are amenable to effective testing by random walk.

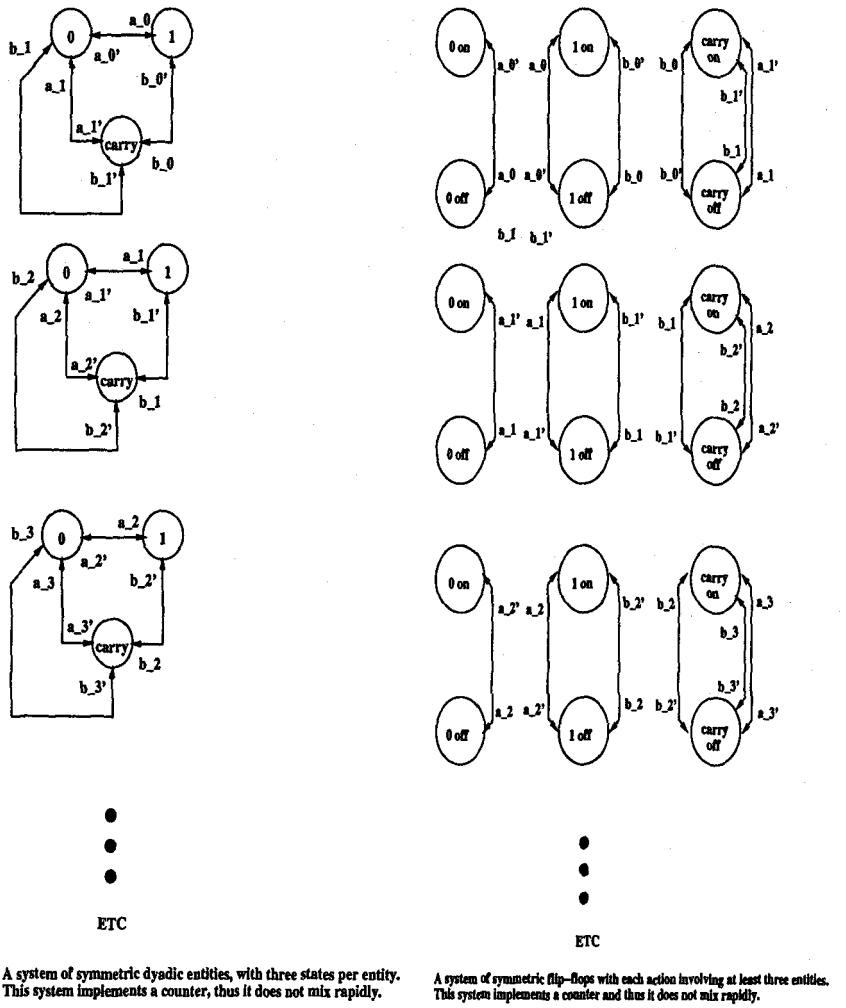


Figure 4: Counter-like counter-examples to rapid mixing.

For general assymmetric protocols (and graphs) little is known about bounds on their cover times —besides the fact that, in the worst case, these bounds are exponential in the size of the graphs. However, experiments suggest more refined behavior (in particular, that under suitable adaptations of the random walk, only a small fraction of the states remains unvisited after what seems to be superpolynomial effort).

References

[1] Aldous, D., "Random Walks on finite groups and rapidly mixing Markov chains", *Seminaire de Probabilites XVII*, Lecture Notes in Mathematics, Vol. 986, Springer Ferlag, Berlin, 1983.

[2] Aldous, D., "On the Markov Chain Simulation Method for uniform combinatorial distributions and simulated annealing", *Probability in Eng. and Inf. Sci. 1*, 1987, pp 33-46.

[3] Aleliunas, R., Karp, R.M., Lipton, J.R., Lovasz, L., and Rackoff, C., "Random walks, universal traversal sequences, and the complexity of maze problems", *Proc. 20th IEEE Symp. on Foundations of Computer Science*, 1979, 218-233.

[4] Apt, K.R., and Kozen, D.Z., "Limits for automatic verification of finite state concurrent systems", *Inf. Processing Letters*, Vol. 22. No. 6, 1986, pp. 307-309.

[5] Broder, A.Z., "How hard is it to marry at random? Approximating the Permanent", *Proc. 18th ACM Symp. on the Theory of Computing*, 1986, pp. 55-58.

[6] Broder, A.Z., "Generating random Spanning Trees", *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989, pp 4422-447.

[7] Cohen, E., Mihail, M., Papadimitriou, C.H., and Tsantilas, T., "Testing Protocols by Random Walk: Mixing and Cover Times", Bellcore TM ARA-4-94, 1994.

[8] Browne, M.C., Clarke, E.M., Dill, D.L., and Mishra, B., "Automatic verification of sequential circuits using temporal logic", *IEEE Trans. on Computers*, Vol. C-35, No. 12, 1986, pp. 1035-1043.

[9] Clarke, E.M., Emerson, E.A., and Sisla, A.P., "Automatic verification of finite state concurrent systems using temporal logic specifications: a practical approach", *Proc. 10th ACM Symposium on Principles of Programming Languages*, 1983.

[10] Diaconis, P., "Group Theory and Statistics", Lecture Notes from a course taught at Harvard, Spring 1982.

[11] Dyer, M., Frieze, A., and Kannan, R., "A random polynomial time algorithm for estimating volumes of convex bodies", *Proc. 21st ACM Symp. on the Theory of Computing*, 1989, pp 375-381.

[12] Feder, T., and Mihail, M., "Balanced Matroids", *Proc. 24th ACM Symp. on the Theory of Computing*, 1992, pp. 26-37.

[13] Holzmann, G.J., *Design and Validation of Computer Protocols*, Prentice Hall Software Series, 1991.

[14] Jerrum, M.R., and Sinclair, A., "Approximating the Permanent", *Proc. 20th ACM Symp. on the Theory of Computing*, 1988, pp 235-243.

[15] Lee, D., and Yannakakis, M., "Online minimization of transition systems", *Proc. 24th ACM Symp. on the Theory of Computing*, 1992, pp. 264-274.

[16] Mihail, M., "Conductance and Convergence of Markov Chains, A Combinatorial Treatment of Expanders", *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989.

[17] Reif, J.H., and Smolka, S.A., "The complexity of reachability in distributed communicating processes", *Acta Informatica*, Vol. 25, 1988, pp. 333-354.

[18] Rudin, H., "Protocol development success stories: Part 1", *Protocol Specification, Testing, and Verification*, XII, Elsevier Science Publishers, (North-Holland), 1992.

[19] West, C.H., "Protocol Validation in Complex Systems", *Proc.. 8th ACM Symposium on Principles of Distributed Computing*, 1989, pp. 303-312.

[20] Wolper, P., "Specifying interesting properties of programs in propositional temporal logic", *Proc. 13th ACM Symposium on Principles of Programming Languages*, 1986, pp. 148-193.

[21] Yannakakis, M., and Lee, D., "Testing Finite State Machines", *Proc. of the 23rd ACM Symp. on the Theory of Computing*, 1991, pp. 476-485.