A Unified Approach to the Evaluation of a Class of Replacement Algorithms

EROL GELENBE

Abstract—The replacement problem arises in computer system management whenever the executable memory space available is insufficient to contain all data and code that may be accessed during the execution of an ensemble of programs. An example of this is the page replacement problem in virtual memory computers. The problem is solved by using a replacement algorithm that selects code or data items that are to be removed from executable memory whenever new items must be brought in and no more free storage space remains. An automaton theoretic model of replacement algorithms is introduced for the class of "random partially preloaded" replacement algorithms, which contain certain algorithms of practical and theoretical interest. An analysis of this class is provided in order to evaluate their performance, using the assumption that the references to the items to be stored are identically distributed independent random variables. With this model, it is shown that the well-known page replacement algorithms FIFO and RAND yield the same long-run page-fault rates.

Index Terms—Markov chains, memory hierarchies, page-fault rates, page replacement algorithms, paging, storage allocation.

I. Preface

THE *replacement problem* is basic to certain situations in which a limited number of resources are multiplexed among a larger number of users. The problem may be stated as follows. Consider a set of users $U = \{u_1, u_2, \cdots, u_n\}$ and a set of identical resources $R = \{r_1, r_2, \cdots, r_m\}$ with $m \leqslant n$. At each instant of time

$$t_1, t_2, \cdots, t_k, \cdots$$

exactly one of the users requests and utilizes a resource without consuming it. At that time, one may discover that all of the resources have already been allocated, so that if our policy is to systematically satisfy each request we shall have to deallocate a resource from one of the users. A *replacement algorithm* is applied at such times to select the user who is going to lose his resource, and the replacement problem is to select an algorithm based upon certain criteria and to evaluate the performance of the algorithm. Various generalizations of this problem may be imagined; one may consider several types of resources, for instance.

A typical application of this problem is to computations carried out with limited memory space. Suppose that the set of resources corresponds to a set of memory units of equal size from which instructions can be executed, and let the set of users be an ensemble of data items (all of equal size) that are stored normally on some peripheral memory unit and brought into a memory unit (one data item per memory unit) at execution time. In such a situation, the replacement algorithm is selected so as to reduce the number of times data items have to be transferred to and from the peripheral memory since this is usually a time consuming operation.

A class of replacement algorithms of interest are the *random partially preloaded* algorithms we shall study here. One may imagine that in many cases it is imperative that certain users never lose their resources, or that certain data items never be removed from their memory units. A random partially preloaded algorithm maintains the resources allocated to these users, and selects the user who will lose his resource at random from the remaining ones with equal probability. This class of algorithms is also of particular interest on theoretical grounds, as will be seen in what follows.

The mathematical model we shall use to describe and evaluate this class of algorithms is a *stochastic automaton* [6]. These automata have appeared in the literature primarily as models of communication channels and sequential machines with random failures, as well as to represent adaptive or learning automata.

In the sequel, we shall discuss the problem using the terminology of storage allocation problems [1] both to simplify the discussion and to provide motivation for the reader. It should be stressed, however, that we consider the problem as being of broader interest.

In some virtual memory computer systems [1], a program's address space is divided into equal size blocks called *pages*. Similarly, primary memory space is divided into *page frames* each of which may contain a page of some program. At a given instant of time, not all of a program's pages need reside in memory so that when the program references a page not in memory, a *page fault* occurs suspending computation until the referenced page can be brought in.

Suppose that a program's set of pages is $N = (p_1, p_2, \cdots, p_n)$ and that exactly m of these can be kept in memory. Then, if $m < n$, each time a page fault occurs one of the pages currently in memory must lose its page frame to accommodate the incoming page. An algorithm that selects the page to be removed from memory when a page fault occurs is called a *page replacement algorithm (PRA)*.

Since a high page-fault rate will cause a deterioration of system performance, it is of great interest to determine the page-fault rate caused by various PRA [2].

The class of random partially preloaded (RPPL) PRA contains both RAND [2] which selects the page to be replaced at random with equal probability among the pages in memory, and A_0 which was shown in [4] to be the "optimal" algorithm if program references are represented as a sequence of

Manuscript received July 10, 1972; revised January 9, 1973. This work was supported in part by the Office of Naval Research (Information Systems Branch) under Contract N00014-67-A-0181-0036, NR 049-311 and in part by a RACKHAM Research Grant.

The author is with the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, Mich., on leave at the Institut de Recherche d'Informatique et d'Automatique (IRIA), Rocquencourt, France.

independent identically distributed random variables. RAND is used as a benchmark [2] since no PRA used in practice should have a worse performance than RAND. The class of RPPL PRA contains

$$\sum_{i=0}^{m-1} \binom{n}{i}$$

algorithms, and the evaluation presented in this paper gives us a single expression for the *long-run page-fault rate* for any algorithm in this class, based on a model of the page reference string.

Any theoretical evaluation of a PRA requires a mathematical model of the reference string. The model we shall use here is the same as the one used by King [3], namely the independent identically distributed model. This model captures the fact that programs do not refer to their pages with the same frequency; i.e., some pages are referred to more frequently than others. It does not capture the correlations that exist between successive page references or any time-varying behavior that may exist. Thus this model is of interest over relatively short lengths of program activity; a detailed discussion of its region of validity is given by Denning and Schwartz [7]. Since replacement algorithms are of more general interest than in their applications to virtual memory machines, the independent reference model yields an evaluation that is more universal than a more realistic but more restricted representation of the reference process.

II. Introduction

In this section we shall define some concepts of use to us. Subsequent sections will contain the results of this paper.

Definition 1: A page reference string is a sequence of symbols from N, the set of pages. It represents the sequence of pages referenced by the program during execution.

Definition 2: A memory state s is an m-element subset of N. There are $\binom{n}{m}$ distinct memory states, where m is fixed and $1 \leqslant m \leqslant n$. S_m is the set of all memory states.

We will now define formally a PRA. Before doing that, however, let us describe informally what it does. A PRA is a control mechanism that examines the page reference string and the memory state, and with this information (and other information that it may store) changes the memory state with the following constraints.

Constraint 1: If the last page referenced is in memory, the new memory state will still contain it.

Constraint 2: If the last page referenced is not in memory, the new memory state will contain it.

This notion of a PRA may be generalized to randomized algorithms in which memory and control-state transitions are probabilistic.

A *probability distribution* on the finite set $V = (v_1, v_2, \cdots, v_f)$ is the row vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_f)$

where

$$\alpha_i \geqslant 0, \quad 1 \leqslant i \leqslant f$$

and

$$\sum_{i=0}^{f} \alpha_i = 1.$$

This is interpreted as follows. Let v be a random variable taking values in V. Then

$$\alpha_i = \Pr\,[v = v_i].$$

We say that α is *degenerate* if for some j, $1 \leqslant j \leqslant f$, $\alpha_j = 1$.

Definition 3: A PRA is the system

$$B = (S, Q, N, s_0, q_0, \{M(r)\})$$

where

S	Nonempty $\lvert S\rvert$[1] element subset of S_m.
Q	Finite $\lvert Q\rvert$ element set of control states.
N	n-element ($n \geqslant m$) set of pages.
s_0	Initial memory state, $s_0 \in S$.
q_0	Initial control state, $q_0 \in Q$.
$\{M(r)\}$	n-element set of stochastic[2] matrices each of which is $\lvert S\rvert \cdot \lvert Q\rvert$ by $\lvert S\rvert \cdot \lvert Q\rvert$, and contains a matrix $M(r)$ for each $p_r \in N$.

Each pair $(s, q) \in S \cdot Q$ is called a *configuration*. To each distinct configuration (s, q), we assign a distinct integer $i = g(s, q)$, $1 \leqslant i \leqslant \lvert S\rvert \cdot \lvert Q\rvert$, with the restriction that $g(s_0, q_0) = 1$. The set $\{M(r)\}$ is interpreted as follows. For any $p_r \in N$, let $m_{ij}(r)$ be the ith row jth column entry of $M(r)$, where $1 \leqslant i \leqslant \lvert S\rvert \cdot \lvert Q\rvert$, $1 \leqslant j \leqslant \lvert S\rvert \cdot \lvert Q\rvert$. Then $m_{ij}(r)$ is the probability that when the program references page p_r and B is in configuration (s, q), the new configuration will be (s', q'), where $i = g(s, q)$ and $j = g(s', q')$.

Let

$$x = p_{r_1} p_{r_2} \cdots p_{r_k} p_{r_{k+1}} \cdots$$

be a page reference string. The PRA B responds to x by passing through a sequence of configurations

$$Y_1, Y_2, \cdots, Y_k, Y_{k+1}, \cdots$$

where

$$\Pr\{g(Y_1) = j \mid (s_0, q_0), p_{r_1}\} = m_{1j}(r_1) \tag{1}$$

and

$$\Pr\{g(Y_{k+1}) = j \mid g(Y_k) = i, p_{r_{k+1}}\} = m_{ij}(r_{k+1}) \tag{2}$$

for any $1 \leqslant i, j \leqslant \lvert S\rvert \cdot \lvert Q\rvert$, and $k > 1$.

We say that a PRA is *deterministic* if for each $p_r \in N$, $m_{ij}(r)$ is either 0 or 1 for all $1 \leqslant i, j \leqslant \lvert S\rvert \cdot \lvert Q\rvert$.

In Definition 3, S is a subset of S_m to indicate that certain PRA (such as A_0 [2]) keep the memory state in a proper subset of S_m.

The model of program behavior under which we will evaluate a PRA is identical to the one defined by King [3]. Let $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ be a probability distribution on N, the set of pages. We shall assume that for any page reference string

[1] $\lvert X\rvert$ is the cardinality of the finite set X.
[2] A matrix is stochastic if each of its entries is nonnegative, and the sum of the entries along each row is 1.

$$p_{r_1} p_{r_2} \cdots p_{r_k} p_{r_{k+1}} \cdots$$

the following properties hold.

Property 1: Pr $[r_k = i] = \beta_i$ for any $k \geq 1$ and $1 \leq i \leq n$.

Property 2: For any $k \neq l, l \geq 1, k \geq 1$, the event $[r_k = i]$ is independent of the event $[r_l = j]$ for any $1 \leq i, j \leq n$.

Property 3: $\beta_i \neq 0$, for each $1 \leq i \leq n$.

This simple model is known as the *independent reference model of program behavior*. Henceforth, it will be understood that the page reference string constitutes a random process governed by Properties 1, 2, and 3. Let

$$Y_1, Y_2 \cdots, Y_k, Y_{k+1}, \cdots$$

be the sequence of configurations the PRA B passes through in response to a page reference string. Due to Property 2 of the page reference string and (1) and (2) we have that

$$\Pr\{g(Y_{k+1}) = j \mid g(Y_k) = i, Y_{k-1}, \cdots, Y_2, Y_1, (s_0, q_0)\} = \Pr\{g(Y_{k+1}) = j \mid g(Y_k) = i\}$$

so that the sequence (1) is a Markov chain [5]. The states of this chain are the configurations of B and the transition probabilities are easily obtained as follows; let c_{ij} be the probability of transition from the configuration numbered i to that numbered j, $1 \leq i, j \leq |S| \cdot |Q|$. Then

$$c_{ij} = \Pr\{g(Y_{k+1}) = j \mid g(Y_k) = i\} = \sum_{r=1}^{n} \beta_r \cdot m_{ij}(r) \tag{3}$$

and we denote by C the matrix whose ith row jth column entry is c_{ij}. Evidently C is a stochastic $|S| \cdot |Q|$ by $|S| \cdot |Q|$ matrix.

Definition 4: A PRA is said to be a *demand paging* algorithm if for any $i = g(s, q)$, $j = g(s', q')$ such that $m_{ij}(r) \neq 0$ and $p_r \notin s$ we have that p_r is the only element in s' that is not in s. That is, only the page that has been demanded is loaded into memory.

In a demand paging PRA, a transition from configuration (s, q) to (s', q') is called a *page-fault transition* if $s \neq s'$.

All PRA studied in this paper are demand paging algorithms.

Now let us turn to the performance measure for a PRA which we will use in this work.

Again, consider the sequence of configurations

$$Y_1, Y_2, \cdots, Y_k, Y_{k+1}, \cdots, Y_w, \cdots$$

that the PRA produces in response to a page reference string. Let $Y_0 = (s_0, q_0)$ and define $f_k(s, q)$, $k \geq 1$, as follows:

$$f_k(s, q) = \begin{cases} 1, & \text{if the transition from } Y_{k-1} \text{ to } Y_k \text{ is a page-fault transition, and } Y_k = (s, q) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

for any configuration (s, q). Let

$$N_w(s, q) = \sum_{k=1}^{w} f_k(s, q). \tag{5}$$

Defintion 5: The expected long-run page-fault rate for the PRA B is

$$F(B) = \sum_{\substack{\text{all} \\ (s,q) \in S \cdot Q}} \left[\lim_{w \to \infty} E\left\{ \frac{N_w(s, q)}{w} \right\} \right]$$

if the limit exists.

For the class of algorithms studied in this paper, the limit in Definition 5 always exists.

Let C^t be the matrix obtained by multiplying C by itself t times. We say that the Markov chain is *irreducible and aperiodic* if there exists a natural number t_0 such that each entry in C^t is nonzero for all $t \geq t_0$. This is equivalent to stating that the probability of transition from any one configuration to any other one (including itself) in t steps is nonzero for all $t \geq t_0$. A chain with this property is also called *regular* [5]. Regular chains have useful properties, some of which will be applied here since we will be dealing with PRA for which the chain with transition matrix C is regular. The following is a well-known theorem.

Theorem 1 [5]: Let the Markov chain with transition matrix C be regular. An $|S| \cdot |Q|$ element stochastic row vector ξ exists such that

$$\xi \cdot C = \xi.$$

The ith entry of ξ, denoted by $\xi(i)$, $i = g(s, q)$, is the long-run probability of finding the chain in configuration (s, q), for any $(s, q) \in S \times Q$. ξ is unique.

A form of $F(B)$ which is more convenient for our purposes is given in the following lemma whose proof can be found in the Appendix.

Lemma 1: Let $T(i)$ be the set of integers

$$T(i) = \{j \mid i = g(s, q), j = g(s', q') \text{ and the transition} \} \text{ from } (s, q) \text{ to } (s', q') \text{ is a page-fault transition.}$$

Then if the chain with matrix C is regular, for any initial configuration

$$F(B) = \sum_{i=1}^{|S| \cdot |Q|} \xi(i) \sum_{\substack{\text{all} \\ j \in T(i)}} c_{ij}$$

for any demand paging PRA B and the independent reference model of program behavior.

This lemma simply states that if C is the transition matrix of a regular Markov chain, then the expected long-run page-fault rate is merely the probability of a page fault occurring at steady state.

III. Random, Partially Preloaded Algorithms

In this section we introduce the class of PRA that are the subject of our study. A theorem giving the expected long-run page-fault rate for this class is stated.

Definition 6: For $0 \leq k \leq m - 1$, let ψ_k be any k-element subset of N; ψ_0 is the empty subset. An RPPL PRA is the system B of Definition 3 with the following restrictions.

Restriction 1: $S = \{s \mid s \in S_m \text{ and } \psi_k \subseteq s\}$ so that $|S| = \binom{n-k}{m-k}$.

Restriction 2: $Q = \{q_0\}$, hence $|Q| = 1$ and each $M(r)$ is $|S|$ by $|S|$.

Restriction 3: For any $s, s' \in S$; any $p_r \in N$, $i = g(s, q_0)$, $j = g(s', q_0)$

$$m_{ij}(r) = \begin{cases} 1, & \text{if } i = j \text{ and } p_r \in s \\ \dfrac{1}{m-k}, & \text{if } p_r \in s',\ p_r \notin s, \text{ and } p_r \text{ is the only page in } s' \text{ that is not in } s \\ 0, & \text{otherwise.} \end{cases}$$

Note that for each k there are $\binom{n}{k}$ distinct RPPL algorithms.

An RPPL algorithm always maintains ψ_k in memory, where ψ_k has been chosen on the basis of some decision external to the algorithm. Each time a page fault occurs the page to be replaced is chosen with equal probability among those pages in memory that are not in ψ_k. Clearly, an RPPL algorithm is a demand paging algorithm.

With the following theorem we obtain an expression for the expected long-run page-fault rate of any RPPL PRA.

Theorem 2: Let B be an RPPL PRA with $\psi_k = (p_1, p_2, \cdots, p_k)$ such that $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ is the probability distribution on the set $N = (p_1, p_2, \cdots, p_n)$. (No special relation among the β_i, $1 \leqslant i \leqslant n$, is implied by this choice of ψ_k.) Then

$$F(B) = \frac{\sum\limits_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m} \sum\limits_{p_l \notin s} \beta_l}{\sum\limits_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}$$

where $s = (p_1, p_2, \cdots, p_k, p_{i_{k+1}}, p_{i_{k+2}}, \cdots, p_{i_m})$ if $1 \leqslant k \leqslant m-1$, and $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ if $k = 0$.

Theorem 2 is proved in Section IV. The rest of this section is devoted to an application.

The PRA A_0 [4] maintains the $m-1$ pages whose probability of being referenced is highest, constantly in memory. King [3] obtained $F(A_0)$; we obtain the same result here as a corollary to Theorem 2.

Definition 7: Let the relationship $\beta_1 \geqslant \beta_2 \geqslant \cdots \geqslant \beta_n$ hold. Then A_0 is the RPPL PRA with $k = m-1$, $\psi_{m-1} = (p_1, p_2, \cdots, p_{m-1})$ and $s_0 = (p_1, p_2, \cdots, p_{m-1}, p_m)$.

Theorem 3:

$$F(A_0) = \frac{\sum\limits_{\substack{\text{all} \\ s \in S}} \beta_{i_m} \sum\limits_{p_l \notin s} \beta_l}{\sum\limits_{\substack{\text{all} \\ s \in S}} \beta_{i_m}}$$

where $S = \{s \mid s \in S_m \text{ and } \psi_{m-1} \subset s\}$, and any $s \in S$ is of the form $s = (p_1, p_2, \cdots, p_{m-1}, p_{i_m})$, $m \leqslant i_m \leqslant n$.

$F(A_0)$ in Theorem 3 may be rewritten as

$$F(A_0) = \frac{\sum\limits_{j=m}^{n} \beta_j \sum\limits_{\substack{i=m \\ i \neq j}}^{n} \beta_i}{\sum\limits_{j=m}^{n} \beta_j}$$

since $m \leqslant i_m \leqslant n$, and because $p_l \notin s$ implies that $m \leqslant l \leqslant n$. But then

$$F(A_0) = \frac{\left[\sum\limits_{j=m}^{n} \beta_j\right]^2 - \sum\limits_{j=m}^{n} \beta_j^2}{\sum\limits_{j=m}^{n} \beta_j}$$

which is equal to the expression obtained by King [3].

IV. Proof of Theorem 2

Let C be the transition matrix defined by (3) of Section II. The proof of Theorem 2 consists of the following parts.

Part 1: The entries of C will be obtained.

Part 2: It will be shown that C is the transition matrix of a regular Markov chain.

Part 3: The stochastic row vector ξ satisfying

$$\xi C = \xi$$

will be obtained.

Part 4: F (RPPL) will be derived using Lemma 1.

Part 1: By Definition 6, C is $|S|$ by $|S|$. Let $u = g(s, q_0)$, $v = g(s', q_0)$; $s, s' \in S$. Then by (3) and Restriction 3 of Definition 6 we have

$$c_{uu} = \sum_{r=1}^{n} \beta_r m_{uu}(r) = \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r \tag{6}$$

$$c_{uv} = \frac{\beta_r}{m-k}, \quad \text{if } p_r \notin s,\ p_r \in s', \text{ and } p_r \text{ is the only page in } s' \text{ that is not in } s \tag{7}$$

$$c_{uv} = 0, \quad \text{in all cases not covered by (6) and (7).} \tag{8}$$

Part 2: To show that C is regular, it must be shown that the probability of reaching any configuration $w = g(s'', q_0)$ from any $u = g(s, q_0)$ in t steps is nonzero for any $t \geqslant t_0$, and t_0 fixed. This is easily proved. Let s'' contain y pages not in s; let these pages be

$$p_{z_1}, p_{z_2}, \cdots, p_{z_y}.$$

Evidently $y \leqslant m-k$. Then $w = g(s'', q_0)$ is reached from $u = g(s, q_0)$ with no more than $m-k$ transitions of the type whose probability is given in (7). Since $\beta_i \neq 0$, $1 \leqslant i \leqslant n$, by Property 3 of the independent reference model of program behavior it follows that the probability of this y-step transition is nonzero since it can be no smaller than

$$\frac{\beta_{z_1} \cdot \beta_{z_2} \cdots \beta_{z_y}}{(m-k)^y}. \tag{9}$$

The probability of reaching w from u in more than y steps is also nonzero since an arbitrary finite-number transitions of the type whose probability is given in (6) can be used as well. Therefore C is regular and $t_0 \leqslant m-k$.

Part 3: For any $s \in S$, recall that without loss of generality we write

$$s = (p_1, p_2, \cdots, p_k, p_{i_{k+1}}, p_{i_{k+2}}, \cdots, p_{i_m}) \tag{10}$$

if $k \geq 1$ and $\psi_k = \{p_1, p_2, \cdots, p_k\}$, and

$$s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m}) \tag{11}$$

if $k = 0$ (since ψ_0 is empty). Further, note that for $0 \leq k \leq m - 1$

$$p_{i_{k+j}} \notin \psi_k \tag{12}$$

for any $1 \leq j \leq m - k$. Let $(s, q_0), (s', q_0)$ be two configurations of the RPPL algorithms so that $u = g(s, q_0)$ and $v = (s', q_0)$. By Theorem 1 and Part 2 of this proof, there exists an $|S|$-dimensional row vector ξ whose uth entry $\xi(u)$, $1 \leq u \leq |S|$, is the long-run probability of finding the Markov chain C whose entries are given by (6), (7), and (8), in configuration (s, q_0) where $u = g(s, q_0)$. Furthermore, ξ satisfies

$$\xi \cdot C = \xi \tag{13}$$

and

$$\sum_{u=1}^{|S|} \xi(u) = 1. \tag{14}$$

Equation (13) may be written as $|S|$ equations of the form (for each $1 \leq u \leq |S|$, $u = g(s, q_0)$, $s \in S$)

$$\xi(u) = \sum_{v=1}^{|S|} \xi(v)\, c_{vu}$$

which, using (6), (7), and (8) becomes

$$\xi(u) = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \sum_{\substack{\text{all} \\ v \in R(u,j)}} \xi(v) \cdot \frac{\beta_{i_{k+j}}}{m - k} \tag{15}$$

where $R(u, j)$ is the set of all $v = g(s', q_0)$ such that if s' is the memory state when the program references page $p_{i_{k+j}}$, for some j, $1 \leq j \leq m - k$, then a page fault will occur and the new memory state will be s with probability [from (7)]

$$c_{vu} = \frac{\beta_{i_{k+j}}}{m - k}. \tag{16}$$

Thus, each s' may be expressed as

$$s' = s \cup \{p_a\} - \{p_{i_{k+j}}\} \tag{17}$$

where $p_a \notin s$, if $v = g(s', q_0) \in R(u, j)$. We shall show that for each u, $1 \leq u \leq |S|$,

$$\xi(u) = \frac{\beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}} \tag{18}$$

satisfies (15), where $u = g(s, q_0)$ and s is given in (10). By (17) and (18)

$$\xi(v) = \frac{\beta_a}{\beta_{i_{k+j}}}\, \xi(u). \tag{19}$$

Let the right-hand side of (15) be called A. To show that $\xi(u)$ in (18) satisfies (15), it suffices to prove that by substituting (19) in A we still obtain

$$A = \xi(u).$$

Substituting (19) in A we get that

$$A = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \sum_{\substack{\text{all} \\ v \in R(u,j)}} \beta_a \frac{\xi(u)}{m - k}.$$

But, for each $p_a \notin s$, $R(u, j)$ contains exactly one v and $|R(u, j)| = m - n$, independently of j. Therefore,

$$A = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \sum_{j=1}^{m-k} \frac{\xi(u)}{m - k} \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r. \tag{20}$$

But $\xi(u)$ [in (18)] is independent of j, therefore,

$$A = \xi(u) \sum_{\substack{\text{all} \\ p_r \in s}} \beta_r + \xi(u) \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r = \xi(u) \tag{21}$$

so that the proof that $\xi(u)$ in (18) satisfies (15) and (13) is complete. The vector ξ defined by (18) is stochastic since its entries sum to unity. This completes Part 3 of the proof.

Part 4: By Lemma 1, and Part 2 of the proof, for any RPPL PRA, independently of the initial configuration

$$F(\text{RPPL}) = \sum_{u=1}^{|S|} \xi(u) \sum_{\substack{\text{all} \\ v \in T(u)}} c_{uv}. \tag{22}$$

Let $u = g(s, q_0)$ and let $p_r \notin s$. If the RPPL PRA is in configuration (s, q_0) when the program references page p_r, the PRA may enter any one of $(m - k)$ configurations of the form (s', q_0), $v = g(s', q_0)$, where $s' = s \cup \{p_r\} - \{p_{i_{k+j}}\}$ for any $1 \leq j \leq m - k$, with probability [see (7)] $\beta_r/(m - k)$. Evidently, each such v is in $T(u)$. Since this is true for each $p_r \notin s$, we have

$$\sum_{\substack{\text{all} \\ v \in T(u)}} c_{uv} = \sum_{\substack{\text{all} \\ p_r \notin s}} (m - k) \cdot \frac{\beta_r}{m - k} = \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r. \tag{23}$$

Turning to (22), we see that it may be rewritten as

$$F(\text{RPPL}) = \sum_{\substack{\text{all} \\ s \in S}} \xi(u) \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r \tag{24}$$

using (23) and the fact that for each $1 \leq u \leq |S|$ there is a unique $s \in S$ such that $u = g(s, q_0)$, and vice versa. When (20) is used in (24) we obtain

$$F(\text{RPPL}) = \frac{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m} \sum_{\substack{\text{all} \\ p_r \notin s}} \beta_r}{\sum_{\substack{\text{all} \\ s \in S}} \beta_{i_{k+1}} \beta_{i_{k+2}} \cdots \beta_{i_m}}$$

which completes the proof of Theorem 2.

V. The Algorithms RAND and FIFO

It is the purpose of this section to define and examine the algorithm RAND and to show that the expected long-run page-fault rate or RAND and of the well-known PRA FIFO [3] are equal for the independent reference model of program behavior.

RAND is, in some sense, the most trivial PRA. Whenever a page fault occurs, RAND chooses the page to be replaced at random among the pages in memory so that the probability of being removed is equal for all pages in memory. Thus any PRA being used should have a page-fault rate that is at least no worse than RAND's, and RAND is useful as a basis for comparison.

FIFO (first-in first-out), on the other hand, always replaces the page that was first to enter memory among the pages currently in memory. FIFO is briefly discussed in [2]. King [3] has obtained $F(\text{FIFO})$ under the independent reference model of program behavior.

Definition 8: RAND is the RPPL PRA with $k = 0$ (so that ψ_0 is the empty subset of N, and $S = S_m$).

As an immediate consequence of Theorem 2 and Definition 8 we have the following theorem.

Theorem 4: The expected long-run page-fault rate of RAND is

$$F(\text{RAND}) = \frac{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m} \sum\limits_{\substack{\text{all}\\ p_r\notin s}} \beta_r}{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}}$$

where $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ for any $s \in S$. From King's paper [3] we have an expression $F(\text{FIFO})$ that we will presently show to be identical to $F(\text{RAND})$. But first let us describe the PRA FIFO. Informally, FIFO keeps a marker on that page that was first to enter memory among all pages currently in memory. When a page fault occurs, the marked page is removed from memory and the marker is updated. Thus, FIFO is a deterministic PRA with $S = S_m$; whenever $s = (p_{i_1}, p_{i_2}, \cdots, p_{i_m})$ is the memory state, the control state is $q = (p_{j_1}, p_{j_2}, \cdots p_{j_m})$ where $(j_1, j_2, \cdots, j_m)$ is a permutation of $(i_1, i_2, \cdots, i_m)$ so that $p_{j_1}, p_{j_2}, \cdots, p_{j_m}$ is the ordering of the members of s from left to right in order of first entry into memory (i.e., p_{j_1} is the first-in-first-out). If now the program references $p_a \notin s$ causing a page fault, the new memory state is

$$s' = s \cup \{p_a\} - \{p_{j_1}\}$$

and the new control state is

$$q' = (p_{j_2}, p_{j_3}, \cdots, p_{j_m}, p_a).$$

Thus, for each memory state there are $m!$ possible control states. For a formal treatment of FIFO the reader is referred to [3].

Theorem 5–(King [3]): The expected long-run page-fault rate of FIFO under the independent reference model of program behavior is

$$F(\text{FIFO}) = \frac{\sum\limits_{\substack{\text{all}\\ q\in Q}} \beta_{j_1}\beta_{j_2}\cdots\beta_{j_m}(1-\beta_{j_1}-\beta_{j_2}-\cdots-\beta_{j_m})}{\sum\limits_{\substack{\text{all}\\ q\in Q}} \beta_{j_1}\beta_{j_2}\cdots\beta_{j_m}}$$

where

$$q = (p_{j_1}, p_{j_2}, \cdots, p_{j_m}).$$

When the control state is q of Theorem 4, the memory state is $s = (p_{i_1}, p_{i_2}, \cdots p_{i_m})$ where $(j_1, j_2, \cdots, j_m)$ is a permutation of $(i_1, i_2, \cdots, i_m)$ so that

$$\sum_{\substack{\text{all}\\ q\in Q}} \beta_{j_1}\beta_{j_2}\cdots\beta_{j_m} = (m!) \sum_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}.$$

Furthermore,

$$\sum_{\substack{\text{all}\\ p_r\notin s}} \beta_r = (1-\beta_{j_1}-\beta_{j_2}-\cdots-\beta_{j_m})$$

so that

$$F(\text{FIFO}) = \frac{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m} \sum\limits_{\substack{\text{all}\\ p_r\notin s}} \beta_r}{\sum\limits_{\substack{\text{all}\\ s\in S}} \beta_{i_1}\beta_{i_2}\cdots\beta_{i_m}}$$

and since $S = S_m$ for both FIFO and RAND, we have the following theorem.

Theorem 6: $F(\text{FIFO}) = F(\text{RAND})$.

Appendix

Proof of Lemma 1

It is to be shown that if the Markov chain with transition matrix C, given by (3), is regular for the PRA B under the independent reference mode of program behavior, then the expected long-run page-fault rate $F(B)$ defined in Definition 5 is given by

$$F(B) = \sum_{i=1}^{|S|\cdot|Q|} \xi(i) \sum_{\substack{\text{all}\\ j\in T(i)}} c_{ij} \tag{1.1}$$

where $\xi(i)$ is the ith entry of the vector ξ of Theorem 1, and c_{ij} is an entry of the matrix C defined by (3).

Let $u = g(s, q)$, $v = g(s', q')$, for any two configurations (s, q), (s', q'). Let

$$Y_1, Y_2, \cdots, Y_k, \cdots$$

be the sequence of configurations the PRA B passes through in response to a page reference string; B is started in $Y_0 = (s_0, q_0)$. Define the function

$$h_k(u) = \begin{cases} 1, & \text{if } g(Y_k) = u \\ 0, & \text{otherwise} \end{cases} \tag{1.2}$$

for all $k \geq 0$, and

$$M_w(u) = \sum_{k=0}^{w-1} h_k(u). \tag{1.3}$$

We will need the following theorem known as the law of large numbers for regular Markov chains.

THE LAW OF LARGE NUMBERS ([5, THEOREM 4.2.1])

Let C be the transition matrix of a regular chain with stochastic vector ξ satisfying

$$\xi C = \xi.$$

Then, for any initial state (in this case "configuration") it follows that

$$\lim_{w\to\infty} E\left\{\frac{M_w(u)}{w}\right\} = \xi(u)$$

and for any $\epsilon > 0$

$$\lim_{w\to\infty} \Pr\left\{\left|\frac{M_w(u)}{w} - \xi(u)\right| > \epsilon\right\} = 0.$$

For any $v = g(s', q')$, $k \geq 1$, let

$$X_k(u, v) = \begin{cases} 1, & \text{if } g(Y_{k-1}) = u \text{ and } g(Y_k) = v \\ 0, & \text{otherwise} \end{cases} \tag{1.4}$$

and

$$e_w(u, v) = E\left\{\frac{\sum_{k=1}^{w} X_k(u, v)}{w}\right\} = \frac{1}{w}\sum_{k=1}^{w} E\{X_k(u, v)\}. \tag{1.5}$$

But quite simply, by (1.4)

$$E\{X_k(u, v)\} = \Pr\{X_k(u, v) = 1\} \tag{1.6}$$

and by Bayes' rule and (1.2)

$$\Pr\{X_k(u, v) = 1\} = \Pr\{v = g(Y_k) \mid u = g(Y_{k-1})\} \cdot \Pr\{h_{k-1}(u) = 1\}.$$

But since

$$Y_1, Y_2, \cdots Y_k, \cdots$$

is a Markov chain, we have that

$$c_{uv} = \Pr\{v = g(Y_k) \mid u = g(Y_{k-1})\}$$

independently of k, so that by (1.5) and (1.6) we obtain

$$e_w(u, v) = \frac{1}{w}\sum_{k=1}^{w} c_{uv} \Pr\{h_{k-1}(u) = 1\}. \tag{1.7}$$

But then

$$e_w(u, v) = \frac{1}{w} c_{uv} \sum_{k=0}^{w-1} \Pr\{h_k(u) = 1\}$$

and by (1.2)

$$\Pr\{h_k(u) = 1\} = E\{h_k(u)\}.$$

Therefore, using (1.3) we obtain

$$e_w(u, v) = c_{uv} E\left\{\frac{M_w(u)}{w}\right\}$$

so that by the law of large numbers

$$\lim_{w\to\infty} e_w(u, v) = \xi(u)\, c_{uv} \tag{1.8}$$

for any u, v. In particular, this is true if the transition from (s, q) to (s', q') is a page-fault transition, $u = g(s, q)$, $v = g(s', q')$. With reference to (4) of Section II and (1.4), notice that $f_k(s', q') = 1$ if and only if $X_k(u, v) = 1$ for any (s, q) such that the transition from (s, q) to (s', q') is a page-fault transition. Therefore, from (5)

$$N_w(s', q') = \sum_{k=1}^{w} \sum_{\substack{\text{all } u \\ \text{such that} \\ v\in T(u)}} X_k(u, v) \tag{1.9}$$

so that by (1.5)

$$E\left\{\frac{N_w(s', q')}{w}\right\} = \sum_{\substack{\text{all } u \\ \text{such that} \\ v\in T(u)}} e_w(u, v). \tag{1.10}$$

By (1.8) and (1.10)

$$\lim_{w\to\infty} E\left\{\frac{N_w(s', q')}{w}\right\} = \sum_{\substack{\text{all } u \\ \text{such that} \\ v\in T(u)}} \xi(u)\, c_{uv}. \tag{1.11}$$

Finally from (1.11) and Definition 5 we obtain

$$F(B) = \sum_{\substack{\text{all} \\ (s',q')\in S\cdot Q}} \lim_{w\to\infty} E\left\{\frac{N_w(s', q')}{w}\right\} = \sum_{v=1}^{|S|\cdot|Q|} \sum_{\substack{\text{all } u \\ \text{such that} \\ v\in T(u)}} \xi(u)\, c_{uv}.$$

The two summations in the above expression are used to sum $\xi(u)\, c_{uv}$ over all pairs (u, v) such that $v \in T(u)$. Therefore, we may rewrite this as

$$F(B) = \sum_{u=1}^{|S|\cdot|Q|} \sum_{\substack{\text{all} \\ v\in T(u)}} \xi(u)\, c_{uv}$$

completing the proof.

REFERENCES

[1] B. Randell and C. J. Kuehner, "Dynamic storage allocation system," *Commun. Ass. Comput. Mach.*, vol. 11, pp. 297–305, May 1968.

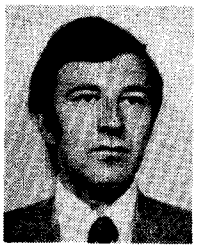
[2] L. A. Belady, "A study of replacement algorithms for a virtual storage computer," *IBM Syst. J.*, vol. 5, pp. 78–101, July 1966.

[3] W. F. King, III, "Analysis of paging algorithms," in *Proc. IFIP Congr.* (Ljubjana, Yugoslavia), 1971; also IBM Rep. RC 3288.

[4] A. V. Aho, P. Denning, and J. D. Ullman, "Principles of optimal page replacement," *J. Ass. Comput. Mach.*, vol. 18, pp. 80–93, Jan. 1971.

[5] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. Princeton, N.J.: Van Nostrand, 1960.

[6] T. L. Booth, *Sequential Machines and Automata Theory*. New York: Wiley, 1967.
[7] P. J. Denning and S. C. Schwartz, "Properties of the working set model," in *Ass. Comput. Mach. 3rd Symp. Operating Syst. Principles*, Oct. 1971, pp. 130–140.
[8] A. Paz, *Introduction to Probabilistic Automata*. New York: Academic, 1971.



Erol Gelenbe (S'67–M'70) was born in Istanbul, Turkey, on August 22, 1945. He received the B.S. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, in 1966 and the Ph.D. degree in computer science from the Polytechnic Institute of Brooklyn, N.Y., in 1969.

After obtaining his degree he was appointed to an Assistant Professorship at the Polytechnic Institute of Brooklyn and taught courses in programming and in switching and automata theory. During the summers of 1970 and 1972 he was a Visiting Scientist and Lecturer at the Philips Research Laboratories, Eindhoven, The Netherlands, where he did research in the modeling of computer operating systems. Since the fall of 1970 he has been on the faculty of the Department of Electrical and Computer Engineering and of the Graduate Program in Computer, Information and Control Engineering, University of Michigan, Ann Arbor, where he introduced new courses in compiler writing, programming languages, and operating systems theory. On leave of absence from the University of Michigan since January 1972, he has been at the research laboratory (LABORIA) of the Institut de Recherche d'Informatique et d'Automatique (IRIA), Rocquencourt, France, first participating in the design of an operating system, and then initiating and heading a research project in models of computer systems. His primary research interest at the present time is the quantitative evaluation and modeling of operating systems and of other information processing systems. He has published several papers in journals and at symposia on that subject, as well as on automata and formal language theory. He is a referee for several journals in computer science, and is a European Lecturer of the Association for Computing Machinery. He is active in ACM activities in Europe, is a member of the program committee of the 1973 ACM International Computing Symposium, and was co-chairman of the Symposium on Computers and Automata held in New York City in 1971.

Circuit Structure and Switching Function Verification

MIN-WEN DU AND C. DENNIS WEISS

Abstract—**A new approach is presented for the design of multiple fault detection tests in which the structure of a combinational circuit is used to reduce the number of input combinations required. The structure is defined by the interconnection of the basic elements, each of arbitrary complexity. The fault model assumes that the functions realized by the basic elements may undergo any deviation whatsoever, but that the circuit structure is fault free. Thus, arbitrary combinations of multiple faults within one or more basic elements are included in the model. Decomposition theory can be used to verify that a set of input combinations is a multiple fault detection test set under this model. A process called *expansion* will be introduced to simplify this task. A well-defined procedure is given for deriving a suitable test set which for some circuits is minimal or near minimal. It will yield a multiple fault detection test of length less than 2^n for any circuit with a nontrivial nondisjoint decomposition, defined by a *basic-element partition*. Higher order basic-element partitions are introduced as a generalization. An upper bound is given on the length of a multiple fault detection test for any circuit with a given structure, independent of the function realized on the structure. The bound is tighter when function information is also used.**

Index Terms—**Combinational logic networks, fault detection, functional decomposition, multiple faults.**

Manuscript received October 13, 1971; revised November 27, 1972. A preliminary version of this paper was presented at the 9th Allerton Conference on Circuit and System Theory, October 1971.

M.-W. Du was with the Department of Computer Science, The Johns Hopkins University, Baltimore, Md. 21218. He is now with the College of Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

C. D. Weiss was with the Departments of Computer Science and Electrical Engineering, The Johns Hopkins University, Baltimore, Md. He is now with Bell Laboratories, Holmdel, N.J. 07733.

I. Introduction

DESIGNING tests for multiple faults in combinational circuits has proven to be extremely difficult, primarily because most approaches require the consideration of each combination of possible signal faults, usually assumed to be of the stuck-at-1 or stuck-at-0 variety. If there are n lines in a circuit, there are $3^n - 1$ possible multiple stuck faults. Although these may be considered in equivalence classes [1], [2], the number of such classes may still be quite large.

The most familiar approaches to the design of fault detection tests involve either simulation [3] or employ the basic path sensitization idea [4]–[6], sometimes with the use of the Boolean difference [7]–[10]. In the present study, a quite different approach is possible, in which knowledge of the structure of a combinational circuit is used to reduce the number of input combinations required to test the circuit. The fault free approach employed in [11] also exploits circuit structure, and the results reported here can be incorporated into the procedures in [11].

A circuit's structure is defined by the interconnection of the basic elements, each of arbitrary complexity. The idea is to exploit the fact that the structure of a circuit implies constraints among certain sets of subfunctions of f_C, the circuit output function. These constraints enable one to draw inferences of the following form.

If a circuit C with known structure produces outputs $f_C(S)$