Modeling Transient Trace Data

Anup Mathur Marc Abrams

Technical Report TR 96-14

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106
{mathur,abrams}@cs.vt.edu

October 25, 1996

Abstract

This paper introduces a novel technique to construct an empirical workload model fitting time-varying (transient) trace data. The trace can be a categorical or numerical time-series. We model the trace as a Piecewise Independent stochastic process. To estimate the parameters for our model we first build a Rate Evolution Graph from the trace data. Piecewise linear regression is then used to construct a joint time-dependent probability mass function for the trace data. Two methods are proposed to build a parsimonious model. The modeling approach is demonstrated by the application of our model to twelve traces from the performance analysis domain.

1 Introduction

Creation of accurate non executable, artificial workload models [FSZ83] of computer systems is a key step in evaluating, debugging, and tuning these systems. Workload models for a system can be built from traces collected by observing a system. Traces (or log files) from several domains, including program behavior traces, contain transient behavior, however, most modeling methods in the literature are based on *stationary* stochastic process models. Self-similar stochastic processes [Ber94, LTWW94] have been successfully used to model trace data previously thought to be non-stationary because they exhibited *long range dependence*. However, these models do not address modeling traces of transient data. Most trace modeling methods apply only to traces containing numerical data only. However traces sometimes contain *categorical data*, that is identified by a category rather than an ordinal value. For example, in a log file from a World Wide Web server the document type (e.g., text, image) is a category and not a number. This paper contains a novel approach for constructing an empirical model for transient trace data containing either numerical or categorical traces.

1.1 Related Work

There are many methods in the literature for deriving models from trace, including models of traces of program behavior [GH93, ADM92], modeling network traffic packet-traces [PF94, LTWW94], and modeling traces of compressed video [Gar93, HTL92]. The state of the art is illustrated by Heyman and Lakshman [HL96], who model twelve video traffic traces with a variety of stationary distributions. The *Gamma*, *Paretto*, and *Weibull* distributions are used for traffic sequences that not autocorrelated, and the Markov chain and discrete auto regressive models are used with autocorrelated traffic sequences.

1.2 Our Approach

A *state* is a discrete value or a vector of discrete values that uniquely describes the system that a trace represents. We assume here that a state is a discrete scalar value; an extension for vector states is discussed in [Mat96]. A *trace* is a sequence of states that records the behavior of the system over a period of time. Generally a trace can have time stamps associated with every occurrence of a state. Our analysis is limited to traces which can be considered as *time series* [BJ76] with an index parameter space that is discrete, equally spaced, and monotonically increasing. The term *event number* refers to the index parameter space for a trace.

The key idea underlying our method is to estimate the probability that the state x occurs at event number t in the trace by the rate of occurrence of state x at t. We fit a path to the function representing the rate of occurrence of a state versus time, for each state, then use the derivative of the paths to function that estimates the time-dependent probability mass function of each state. Finally, the collection of individual mass functions forms the joint probability mass function.

For example, consider a two-dimensional Cartesian graph with the number of occurrences of states as the ordinate and event numbers as abscissa. We call this a *Rate Evolution Graph* (REG). The set of points plotted for each unique state forms a path. The slope of the REG path for a state x at t

has units of *number-of-occurrences/event-duration*, or the rate of occurrence of x at t. The REG of the first 150 events for a trace of what document type is retrieved from a World-Wide Web server is shown in Figure 1. The trace contains three states, representing three document types: directory contents, a graphics document, and a text or HTML document. Thus the figure contains three paths. We can fit a polynomial to each path:

$$\begin{aligned}
\text{Rate of occurrence of directory docs} &= 1.15 + 0.03\,t + 1.18 \times 10^{-7}\,t^4 \\
\text{Rate of occurrence of graphics docs} &= -0.84 + 0.97\,t - 1.55 \times 10^{-7}\,t^4 \\
\text{Rate of occurrence of text or HTML docs} &= 1.92 - 0.04\,t + 9.17 \times 10^{-8}\,t^4
\end{aligned}$$

These three equations capture the time dependent trace behavior, from which a time dependent joint probability mass function can be computed. However, we use an alternative to curve fitting in this paper, for two reasons:

1. Fitting a polynomial is an iterative process that requires human intervention. In Figure 1 polynomials of various degrees were tried before selecting a fourth degree polynomial.

2. Polynomial functions that exactly fit a path may need the estimation of a large number of parameters — up to $k - 1$ degrees for k data points.

Our alternative to fitting polynomials is based on a property that we have observed in examining REGs of a few dozen traces: Paths *usually* can be decomposed into a sequence of straight line segments (e.g., Figure 2). Thus we examine in this paper a method to partition the time domain of a trace (hence the x-axis of a REG) into intervals such that all paths in the REG are nearly straight lines. The number of such intervals is typically a small fraction of the length of the trace. We then use regression to fit line segments to sub-paths within intervals, and the slope of the segments is then used to estimate the probability mass function of each state.

Fitting a collection of straight lines per REG path has several advantages over fitting polynomial functions:

1. Fitting a collection of straight lines to a REG path is a mechanical process which can be easily automated.

2. A straight line has a trivial derivative (e.g., its slope). Thus the estimation of the rate of occurrence of a state is a simple computation.

3. If the number of intervals per REG path is small then we have few parameters to estimate (a slope per interval).

The joint probability mass function for the entire trace is consequently expressed as a collection of probability mass functions such that each probability mass function holds over an interval.

Figure 3 shows that our solution approach, the *Piecewise Independent Stochastic Process* (PISP), and its general form, the *Independent Stochastic Process* (ISP), form a subset of REG based stochastic processes that can be used to model transient trace data. The independent stochastic process and the piecewise independent stochastic process are defined in Sections 2 and 3 respectively.

1.3 Problem Statement

The objective of this paper is to define the PISP model, develop a method to estimate the model parameters from sample traces, and evaluate the model quality on a set of twelve test cases drawn from various systems. (See [Mat96] for more details on the traces.)

2 The Independent Stochastic Process

Consider a stochastic process $\{X_t \mid t \in \mathbf{T}\}$ with a discrete and finite index parameter space $\mathbf{T} = \{t_1, t_2, t_3, \ldots, t_n\}$ and discrete state space $\mathbf{S} = \{x_1, x_2, x_3, \ldots, x_m\}$. For X_i in $\{X_t \mid t \in \mathbf{T}\}$ and x_i in $\mathbf{S}$, denote the probability $P\{X_i = x_i\}$ as $p_{x_i}^{X_i}$. For all $x_1, x_2, \ldots . x_n$ in $\mathbf{S}$, denote the set $\{x_1, \ldots, x_n\}$ as $\mathbf{x}$. Also, for all $X_1, X_2, \ldots . X_n$ in $\{X_t \,|\, t \in \mathbf{T}\}$, denote the set $\{X_1, \ldots, X_n\}$ as $\mathbf{X}$. A stochastic process $\{X_t\}$ is usually described using the joint distribution of its member random variables X_t. The joint probability mass function of the stochastic process $\{X_t\}$ denoted $p_{\mathbf{x}}^{\mathbf{X}}$, is given by

$$p_{\mathbf{x}}^{\mathbf{X}} = p_{\{x_1, x_2, \ldots, x_n\}}^{\{X_1, X_2, \ldots, X_n\}} = P\{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\}. \tag{1}$$

In the absence of simplifying assumptions the computation of $p_{\mathbf{x}}^{\mathbf{X}}$ as described in equation (1) would require us to determine $p_{\mathbf{x}}^{\mathbf{X}}$ for all values in the index parameter space $\mathbf{T}$.

The *independent stochastic process* is derived from equation (1) by making the simplifying assumption that all member random variables of the stochastic process $\{X_t\}$ are independent of each other. Thus the joint probability mass function, $p_{\mathbf{x}}^{\mathbf{X}}$, factors into the product

$$p_{\mathbf{x}}^{\mathbf{X}} = p_{x_1}^{X_1}.p_{x_2}^{X_2}\cdots p_{x_n}^{X_n} = \prod_{1\leq t\leq n} p_{x_t}^{X_t}. \tag{2}$$

To model a trace as an independent stochastic process we must obtain all marginal distributions $p_{x_t}^{X_t}$, $1 \leq t \leq n$. Obtaining all the marginal distributions is a formidable task. Also, the storage for such a model will exceed the storage for the trace.

3 The Piecewise Independent Stochastic Process

We introduce the piecewise independent stochastic process as a variant of the independent stochastic process. Consider partitioning $\mathbf{T}$ into r non-overlapping *intervals* $[t_{a_1}, t_{a_2})$, $[t_{a_2}, t_{a_3}), \ldots, [t_{a_r}, t_{a_{r+1}})$, where $0 = t_{a_1} < t_{a_2} < \ldots < t_{a_r} < t_{a_{r+1}} = \infty, 1 \leq r \leq n$, and $\{t_{a_1}, t_{a_2}, \ldots, t_{a_r}\} \in \mathbf{T}$. We shall refer to the set $\{t_{a_1}, t_{a_2}, \ldots, t_{a_r}\}$ as the set of *partitioning events* and denote this set as $\mathbf{R}$. The set $\{X_t \mid t \in \mathbf{R}\}$ is a proper subset of the stochastic process $\{X_t \mid t \in \mathbf{T}\}$ and consists of the family of random variables $\{X_{a_1}, \ldots, X_{a_r}\}$.

<u>**Definition**</u> **1** *A piecewise independent stochastic process is a discrete state and discrete index parameter stochastic process $\{X_t\}$ whose joint probability mass function $p_{\mathbf{x}}^{\mathbf{X}}$ is given by*

$$\forall x_{a_j} \in \mathbf{S},\ p_{\mathbf{x}}^{\mathbf{X}} = \prod_{1\leq j\leq r} p_{x_{a_j}}^{X_{a_j}}. \tag{3}$$

Consequently, the joint probability mass functions of $\{X_t\}$ within two distinct intervals $[t_{a_i}, t_{a_{i+1}})$ and $[t_{a_j}, t_{a_{j+1}})$, denoted $p_{x_{a_i}}^{X_{a_i}}$ and $p_{x_{a_j}}^{X_{a_j}}$ respectively, are independent of each other. Denote the joint probability mass function of X_{a_i} and X_{a_j} as $p_{\{x_{a_i}, x_{a_j}\}}^{\{X_{a_i}, X_{a_j}\}}$. Then,

$$\forall\, x_{a_i}, x_{a_j} \in \mathbf{S} \,\wedge\, \forall\, t_{a_i}, t_{a_{i+1}}, t_{a_j}, t_{a_{j+1}} \in \mathbf{R} \,\wedge\, i \neq j, \quad p_{\{x_{a_i}, x_{a_j}\}}^{\{X_{a_i}, X_{a_j}\}} = p_{x_{a_j}}^{X_{a_i}}.p_{x_{a_j}}^{X_{a_j}}.$$

Also, the joint probability mass function of $\{X_t\}$ within an interval $[t_{a_i}, t_{a_{i+1}})$, denoted $p_{x_{a_i}}^{X_{a_i}}$, is invariant to t. Formally,

$$\forall\, x_{a_i}, x_u, x_v \in \mathbf{S} \,\wedge\, \forall\, t_u, t_v \in [t_{a_i}, t_{a_{i+1}}) \,\wedge\, \forall\, t_{a_i}, t_{a_{i+1}} \in \mathbf{R}, \quad p_{x_u}^{X_u} = p_{x_v}^{X_v} = p_{x_{a_i}}^{X_{a_i}}.$$

To model a trace as a piecewise independent stochastic process we need only to estimate the following parameters:

1. the set of partitioning events $\mathbf{R}$, and
2. the joint probability mass function $p_{\mathbf{x}}^{\mathbf{X}}$ expressed as a set of probability mass functions $\{p_{x_t}^{X_t} \,|\, t \in \mathbf{R}\}$.

4 Estimating Model Parameters

4.1 Overview

In this section we describe the steps to estimate the parameters of the piecewise independent stochastic process model described in section 3. To completely describe this stochastic process we must estimate the set of partitioning events $\mathbf{R}$ and the joint probability mass function $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$.

The first step in our solution approach is to estimate $\mathbf{R}$. In section 1.2 we introduced the REG as a tool that plots the rate of occurrence of the states in a trace. A formal definition of the REG is discussed now. Let the function $g(x_i, t)$ be the number of times that the state x_i occurs up until and including the event number t. Formally, $g(x_i, t) = \|\{t' \,|\, X_{t'} = x_i \,\wedge\, 1 \le t' \le t\}\|$. Also, we define $\mathbf{T_i}$ as a proper subset of $\mathbf{T}$ such that the state x_i occurs at each event number t in $\mathbf{T_i}$. Formally, $\mathbf{T_i} = \{t \,|\, X_t = x_i \,\wedge\, x_i \in \mathbf{S}\}$. The REG path for state x_i, denoted Ψ_i, consists of the points $(t, g(x_i, t))$ such that t is in $\mathbf{T_i}$.

Definition 2 *The* REG path *for state x_i in $\mathbf{S}$ is given by*

$$\Psi_i = \{(t, g(x_i, t)) \mid x_i \in \mathbf{S} \,\wedge\, t \in \mathbf{T_i}\}.$$

A REG is a collection of all REG paths. A REG $\mathbf{\Psi}$ is the set $\{\Psi_i \mid \forall x_i \in \mathbf{S}\}$.

Example 1 *Consider the trace of hits (H) and misses (M) from a cache in Table 2. The state space* $\mathbf{S}$ *for this trace is given by $\{H, T\}$. The integer i indexes the state space. The domain of i is given by the* *set* $\{0,1\}$. *Therefore, for* x_i *in* $\mathbf{S}$, $x_0 = H$ *and* $x_1 = M$. *The set* $\mathbf{T_0} = \{0,1,2,3,8,9,10,11,13,15\}$ *and the set* $\mathbf{T_1} = \{4,5,6,7,12,14,16\}$. *The REG for the states* H *and* M *is displayed as the two paths* Hits *and* Misses, *respectively, in Figure 4.*

Recall from Section 1.2 that within an interval the rate of occurrence of all states remains nearly constant. Also, recall from Section 1.2 that we define a partitioning event as that event number where the rate of occurrence of any state changes (by more than some constant value ϵ) from its previous value. The rate of occurrence of a state x_i at event number t is the slope of the REG path, Ψ_i, at t. Consequently the event number 0 along with any event number at which the slope of any REG path changes significantly from its previous value is a partitioning event.

Example 2 *Figure 4 shows the set of partitioning events for the example trace in Table 2 (see details in Section 5). The rate of occurrence of the state* Hits *changes at event numbers* $0,4,8,12,13,16$. *Similarly, the rate of occurrence of the state* Misses *changes at event numbers* $4,8,12,17$. *Therefore, the set* $\mathbf{R}$ $= \{0,4,8,12,13,16,17\}$ *for this trace.*

The computation of $\mathbf{R}$ is detailed in Section 5. From $\mathbf{R}$ we can determine the joint probability mass functions $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$. Recall from Section 1.2 that the rate of occurrence of state x_i at event number t can be used as an estimate of the probability of occurrence of x_i at t. Denote by $\phi_{i,t}$ the rate of occurrence of x_i at t. Consider the segment of the REG path Ψ_i within an interval $[t_{a_1}, t_{a_2})$. Recall from Section 1.2 that $\phi_{i,t}$ is nearly constant within an interval. Consequently, we can estimate $\phi_{i,t}$ ($t_{a_1} \le t < t_{a_2}$) by the *constant* slope of the straight line fitting the REG path Ψ_i within the interval $[t_{a_1}, t_{a_2})$. The *rate vector*, Φ_t, defined at partitioning event t, is a collection of rates $\phi_{i,t}$ for all states x_i in $\mathbf{S}$.

<u>Definition</u> 3 *Consider an event* $t \in \mathbf{R}$. *The* rate vector Φ_t *is defined as the set*

$$\Phi_t = \{\phi_{i,t} \mid x_i \in \mathbf{R}\}.$$

Example 3 *For the example trace in Table 2 the rate vectors* Φ_t, *for all* t *in* $\mathbf{R}$, *are displayed in Table 3.*

We normalize the rate vector such that all its member rates, $\phi_{i,t}$, satisfy the property $\sum_{x_i \in \mathbf{S}} \phi_{i,t} = 1$. Denote the normalized $\phi_{i,t}$ as $\phi'_{i,t}$. The normalized rate vector, $\phi'_{i,t}$, is the estimate of the probability mass function of the interval starting at the partitioning event t, $p_{x_t}^{X_t}$. Finally, because $\mathbf{R}$ has been already determined, we can compute the set $\mathbf{\Phi} = \{\Phi'_t \mid t \in \mathbf{R}\}$. The set $\mathbf{\Phi}$ estimates the joint probability mass function $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$.

Example 4 *For the example trace in Table 2, the estimated joint probability mass function* $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$ *is displayed in Table 4.*

5 Estimation of Partitioning Events

In this section we describe the computation of the set $\mathbf{R}$. In order to explain the computational details we define some terms next. The *point slope* $s_i(t_u, t_v)$, of the REG path Ψ_i is given by the expression

$$\forall\, (t_u, g(x_i, t_u)), (t_v, g(x_i, t_v)) \in \Psi_i \ \wedge\ t_u < t_v, \quad s_i(t_u, t_v) = \frac{g(x_i, t_v) - g(x_i, t_u)}{t_v - t_u}.$$

A *partition beginning event* is the event number t ($t \in \mathbf{T_i}$) immediately following the event number where the point slope of the state x_i changes more than a tolerance from its previous value. Consider the consecutive members $(x_i, g(x_i, t_u))$, $(x_i, g(x_i, t_{u+1}))$, and $(x_i, g(x_i, t_{u+2}))$ of the REG path Ψ_i. If the absolute difference of the REG path Ψ_i. If the absolute difference between the point slopes $s_i(t_u, t_{u+1})$ and $s_i(t_{u+1}, t_{u+2})$ exceeds a constant ϵ then t_{u+2} is a partition beginning event for x_i.

Definition **4** *The set of* partition beginning events *for state x_i, denoted $\mathbf{B_i}$, is defined as follows:*

$$\mathbf{B_i} = \{t_u \ \mid\ t_{u-1} \notin \mathbf{B_i} \ \wedge\ |s_i(t_{u-2}, t_{u-1}) - s_i(t_{u-1}, t_u)| > \epsilon \ \wedge\ t_{u-2}, t_{u-1}, t_u \in \mathbf{T_i}\}.$$

For the special case where $u = 0$, t_u is in $\mathbf{B_i}$. Similarly, for the special case where $u = 1$, t_u is not in $\mathbf{B_i}$.

A *partition ending event* is the event number t ($t \in \mathbf{T_i}$) that immediately precedes the next partition beginning event (with the exception of the first partition beginning event for a state).

Definition **5** *The set of* partition ending events *for a state x_i, denoted $\mathbf{E_i}$, is defined as follows:*

$$\mathbf{E_i} = \{t_u \ \mid\ t_{u+1} \in \mathbf{B_i} \ \wedge\ u > 0 \ \wedge\ t_u, t_{u+1} \in \mathbf{T_i}\}.$$

An *intervening partition beginning event* is the event number t ($t \in \mathbf{T}$) that begins an interval where the state x_i does not occur.

<u>Definition</u> 6 *The set of* intervening partition beginning events *for a state x_i, denoted $\mathbf{A_i}$, is defined as follows:*

$$\mathbf{A_i} = \{t \mid t-1 \in \mathbf{E_i} \wedge t \in \mathbf{T} - \mathbf{T_i}\}.$$

A *state partitioning event* is the event number t ($t \in \mathbf{T}$) that marks the start of each interval during which the rate of occurrence of the state x_i remains nearly constant. Let us denote the collection of all state partitioning events for x_i as $\mathbf{R_i}$.

<u>Definition</u> 7 *For a state x_i, and its corresponding REG path Ψ_i, the set of state partitioning events, $\mathbf{R_i}$, is given by:*

$$\mathbf{R_i} = \mathbf{B_i} \cup \mathbf{A_i}.$$

Example 5 *For the example trace in Table 2, the sets $\mathbf{B_0}$ and $\mathbf{B_1}$ are given by $\{0, 8, 13\}$ and $\{4, 12\}$ respectively. The sets $\mathbf{E_0}$ and $\mathbf{E_1}$ are given by $\{3, 11, 15\}$ and $\{7, 16\}$ respectively. The sets $\mathbf{A_0}$ and $\mathbf{A_1}$ are given by $\{4, 12, 16\}$ and $\{8, 17\}$ respectively. Finally, the sets $\mathbf{R_0}$ and $\mathbf{R_1}$ are given by $\{0, 4, 8, 12, 13, 16\}$ and $\{4, 8, 12, 17\}$ respectively.*

We call this method of finding state partitioning events by the name *slope-based partitioning*. Another method for finding state partitioning events, *incremental-regression partitioning* is presented later in this document. We have found that *slope-based partitioning* is a more sensitive method because it can find consecutive partitioning events even when the partitioning events are only one event apart.

Once we have the set $\mathbf{R_i}$ for all unique states in the trace, the set of partitioning events $\mathbf{R}$ is determined by computing a union of all $\mathbf{R_i}$. Formally, $\mathbf{R} = \bigcup_{x_i \in \mathbf{S}} \mathbf{R_i}$.

6 Estimation of the Joint Probability Mass Function

To estimate the joint probability mass function $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$ we need the rate vectors Φ_t, for all t in $\mathbf{R}$. Therefore we describe a method to estimate Φ_t first. Recall from Section 1.2 that Φ_t is a set of rates

$\phi_{i,t}$ (x_i in $\mathbf{S}$ and t in $\mathbf{R}$) for the interval that starts at partitioning event t. Consequently, to determine Φ_t we must compute its members $\phi_{i,t}$, for all $1 \leq i \leq m$. Consider an interval $[t_{a_1}, t_{a_2})$. The REG path Ψ_i corresponding to this interval is the set of points $(t_{a_1}, g(x_i, t_{a_1})), \ldots, (t_{a_2} - 1, g(x_i, t_{a_2} - 1))$. We compute the rate $\phi_{i,t_{a_1}}$ as the slope parameter of the linear regression function [Ott88] that fits the points $(t_{a_1}, g(x_i, t_{a_1})), \ldots, (t_{a_2} - 1, g(x_i, t_{a_2} - 1))$.

Example 6 *For the example trace in Table 2, the rates $\phi_{0,t}$ ($t \in \mathbf{R_0}$) are shown in Table 5. Similarly, the rates $\phi_{1,t}$ ($t \in \mathbf{R_1}$) are shown in Table 6.*

We can compute the rate vector $\Phi_{t_{a_1}}$ by computing $\phi_{i,t_{a_1}}$, for all $1 \leq i \leq m$. Let Φ^i denote the subset of $\mathbf{\Phi}$ such that the members of Φ^i are the rate vectors Φ_t corresponding to the state partitioning events in $\mathbf{R_i}$. Formally, $\Phi^i = \{\Phi_t \mid \forall t \in \mathbf{R_i}\}$.

Example 7 *For the example trace in Table 2 the sets Φ^0 and Φ^1 are displayed in Table 7 and Table 8 respectively.*

The set $\mathbf{\Phi}$ can now be computed as a union of all Φ^i. Formally, $\mathbf{\Phi} = \bigcup_{x_i \in \mathbf{S}} \Phi^i$.

Example 8 *For the example trace in Table 2 the set $\mathbf{\Phi}$ is displayed in Table 3.*

Finally, each rate vector Φ_t in the set $\mathbf{\Phi}$ is normalized as described below. Recall that $\Phi_t = \{\phi_{i,t} \mid t \in \mathbf{R}\}$. Each $\phi_{i,t}$ in Φ_t is replaced by the value $\frac{\phi_{i,t}}{\sum_{1 \leq i \leq m} \phi_{i,t}}$. The set of rate vectors $\mathbf{\Phi}$ can now be used as an estimate of the joint probability mass function $\{p_{x_t}^{X_t} \mid t \in \mathbf{R}\}$.

Example 9 *For the example in Table 2 the estimated joint probability mass function is displayed in Table 4.*

7 Model Storage

We store the model as the sets $\mathbf{R_i}$ and $\{\Phi_{i,t} \mid t \in \mathbf{R_i}\}$ for each state x_i in the trace. The expression for the size of the piecewise independent stochastic process model is a function of two factors, namely the number of states in the trace and the number of partitioning events to be stored for each state. As defined earlier the cardinality, m, of the state space $\mathbf{S}$ is the number of states to be stored for the model. Denote as k_i the number of partitioning events to be stored for the state x_i (the cardinality of the set $\mathbf{R_i}$). The storage for the model is therefore given by:

$$C \sum_{i=1}^{m} k_i \tag{4}$$

Constant C denotes the constant storage needed for each $\langle$*partitioning event, rate*$\rangle$ pair.

Example 10 *For the example trace in Table 2 our model is stored as Tables 5 and 6. According to equation (4) above the storage requirement for our model is* $(5+6)C = 11C$ *units.*

8 Parsimonious Model

One of the desirable qualities of a model is parsimony. Specifically, we want the model to be as compact (in terms of storage) as possible without excessive loss of accuracy. Recall from Section 7, that the size of the piecewise independent stochastic process model is given by the expression $C \sum_{i=1}^{m} k_i$. The only variables in this expression are k_i and m. Consequently, two approaches to reduce the model storage are to (1) *reduce the number of partitioning events* that need to be stored for each state and to *reduce the number of states* in the model. In method (1), the key idea is to iteratively reduce the size of $\mathbf{R_i}$ for each state x_i. The iterations converge when $\mathbf{R_i}$ is a singleton set. However, with only one partitioning event per state stored, the model obtained may be extremely inaccurate. Therefore, each iteration to reduce $\mathbf{R_i}$ must be followed by an evaluation of the quality of the model obtained. This insures that we have a compact model without significantly compromising the quality of the model. Approach (2), which only applies to numerical traces, consists of selecting the state space size desired, and then mapping the trace states to this reduced state space size. For numerical traces, the two methods may be combined. The details of both methods is explained in [Mat96].

9 Model Validation

To evaluate the quality of models produced by our method, we select twelve test traces (Table 1) representing a variety of features, then generate models of each trace, then synthetically generate a set of traces (using different seeds) from the models, then compare the synthesized traces to the original traces with a set of validation tests. The accuracy of the models for the twelve traces is then summarized. Finally, we compare the size of the original traces to the size of the models to gauge model parsimony.

As with any empirical study, our conclusions on the quality of our model hinges on how representative our test cases are of traces that practitioners would actually encounter. To achieve some degree of representativeness, our twelve test traces represent both real measured systems and artificial traces, represent both a small and a large number of events, represent both a small and a large number of states, and represent a variety of popular modeling problems (e.g., network and Web traffic, compressed video, and program behavior).

We first list the twelve traces, then explain our validation tests, and finally state the results of validation.

9.1 Test Case Traces (Table 1)

In the three video traces (i.e., Ingalls, ACM, and Star Wars), the number of frame sizes modeled was reduced using a method discussed in [Mat96].

GE Message Trace: This trace was obtained from an instrumented implementation of a Gaussian elimination algorithm, which solves a 64×64 system of equations using 32 processors on an NCUBE multicomputer [RW93]. The trace data obtained by executing this algorithm was filtered to retain time-stamped events which represent the receiving of a message from a processor. The state of the system recorded in the trace is the number of the processor that last sent a message.

Dining Philosophers Trace: This is a trace of a parallel implementation of Dijksta's dining philosophers problem for five philosophers on a symmetrical multiprocessor. Each state is a five tuple (one tuple component per philosopher). Each tuple component records whether the corresponding philosopher is thinking, acquiring resources, eating, or releasing resources.

UoS Ethernet Trace: This is a trace of Ethernet traffic from the University of Saskatchewan. The trace records the addresses of machines generating traffic to a certain host on the Ethernet.

IBM CICS Trace: This trace was obtained from a CICS transaction system running on an IBM 3090. The trace records calls made to various CICS functions during a 19 minute interval of system operation.

GE Memory Trace: This trace was obtained by executing a sequential implementation of the Gaussian elimination algorithm. The trace records the memory addresses referenced during the execution.

WWW Trace: This trace of WWW traffic records the sequence of document types (i.e., graphics, directory, audio, text or HTML, script, video, or other) accessed from an undergraduate student lab at Virginia Tech.

Ingalls Trace: This trace contains frame sizes from the MPEG movie of an ACM videotape presentation titled "Object-Oriented Programming" by Dan Ingalls. The trace was recorded at 30 frames per second using a display window size of 320 lines $\times$ 240 pixels.

ACM Trace: This is a trace of frame sizes from the MPEG movie of a videotape presentation titled "An Introduction to Interactive Digital Video." The trace was recorded at 10 frames per second using a display window size of 320 lines $\times$ 240 pixels.

Star Wars Trace: This is a trace of frame sizes of the Star Wars movie [Gar93] [1]. The trace is a sequence of frame sizes from the movie.

Periodic Trace: This trace is generated from a known synthetic distribution. The two unique states in this trace occur with identical probability (0.5).

[1] The trace is available from ftp://thumper.bellcore.com/pub/vbr.video.traces

Time-trend Trace: This trace is generated from a known synthetic distribution. The trace has two unique states (0 and 1). The probability of the first state increases proportional to the length of the trace. The probability of the second state decreases proportional to the length of the trace. Denote the trace as a time series $\{X_t\}$ where $1 \leq t \leq N$ and the random variables X_t are IID. The probability mass for the random variables X_t, denoted $P\{X_t\}$ is given by:

$$P\{X_t = x\} = \begin{cases} \frac{t}{N} & \text{if } x = 0 \\ 1 - \frac{t}{N} & \text{if } x = 1. \end{cases}$$

Mean-change Trace: This trace is generated from a known synthetic distribution. The trace has four unique states. The main behavioral characteristic of this trace is that it shows three intervals of distinct behavior. Within each interval only two out of the four states occur, each with probability 0.5.

9.2 Validation Tests

We use three validation tests, explained below.

Kruskall-Wallis Rank Sum Test: The KW test [Ott88] is non-parametric; its advantage is that it does not make any assumptions about the shape of the distribution of the trace data. The following null hypothesis is tested: *A trace synthesized from our model is generated from the same distribution that generated the original trace.*

We compare the original trace to five synthetically generated traces with the KW test as follows. To appropriately test the time dependent nature of the trace data we divide the synthesized and the original traces into ten time segments and then apply the KW test to corresponding partitions, and report the percentage of partitions that passed the test. (Hence 100% is complete agreement.)

Correlation Analysis: For numerical traces, we compare the autocorrelation of the original to the synthetic traces. For categorical traces, we use the categorical correlation plot [ARM96], which measures the degree of correlation between two traces. To validate our model we measure the categorical correlation between a test case trace and a synthetic trace generated from our model of the test case trace. The categorical correlation plots are constructed using a moving window, denoted W, of size 32 events. The categorical correlation is plotted as the fraction of states (x_i), within W, that are common to the test case trace as well as the synthetic trace. The categorical correlation is plotted on the ordinate and is always in the range $[0-1]$. A value of 0 indicates the two traces have no states in common within the window W. A value of 1 indicates that two traces contain 32 identical states (not necessarily in the same order) within the window W.

2D Views: We also visual compare the two dimensional graph of state versus time of the original trace and the synthesized traces.

9.3 Results

Kruskall-Wallis test: Table 13 shows the results of the Kruskall-Wallis test on the test case traces (Section 9.1). With the exception of the Dining Philosophers trace and the Star Wars trace, all the other traces synthesized from our model match their corresponding original traces in 7 segments out of 10 or more. The model is exceptionally accurate for the GE message trace, IBM CICS trace, mean-change trace, and Periodic trace. The Kruskall-Wallis test does not show our model to be good fit for the Dining Philosophers trace. However, this result is contradicted by other two validation tests for this trace, namely the categorical correlation plot and 2D Views. A possible explanation for the Kruskall-Wallis test results for the Dining Philosophers trace is the fact that this trace has a very large number of unique states (more than 10% of the trace length).

Correlation Plots and 2D Views: The correlation plots and 2D Views of the test case traces are displayed in Figures 5-11.

The 2D views of the test case trace compare very favorably with their synthetic counterparts. The 2D views of the GE message trace, IBM CICS trace, Dining Philosophers trace, mean-change trace match their synthetic counterparts exceptionally well. The notable exception is the 2D view of the GE memory trace and its synthetic counterpart.

Most test case traces, with the exception of the Dining Philosophers trace, the Star Wars trace, the Ingalls trace, and the GE memory trace display an adequate degree of correlation with their synthetic counterparts (the correlation coefficient is greater than 0.5). The IBM CICS trace, WWW trace, mean-change trace, and Periodic trace show notably high degree of correlation with their synthetic counterparts. This indicates that our model is a good fit for these traces.

The autocorrelation plots for numerical traces show that our model does not adequately track the autocorrelation present in the these traces. For the Star Wars trace, the Ingalls trace, the ACM trace, and the Time-trend trace, all highly autocorrelated traces our model does not track the autocorrelation adequately. For the Periodic trace and the mean-change trace which are not highly autocorrelated our model performs adequately.

10 Model Size

We reduced the number of partitioning events (Section 7) in the models of all the test case traces. We also reduced the number of states in the Star Wars trace and the Ingalls trace by half.

Table 14 displays the size of the model for every test case trace. In most cases the model is approximately 40% the trace size. Three notable exceptions to the above are the Periodic trace, Time-trend trace, and the GE memory trace. The models for the Periodic and Time-trend are extremely small because these traces are generated from very simple distributions (distributions with very few behavioral changes). The GE memory trace is an extremely short trace (376 *events*) with an extremely large number of states (48). The GE memory trace also has a high degree of autocorrelation. All these factors require our model to store a lot of information about it and consequently the model of the GE memory trace is not very compact.

11 Sources of Error

The application of our model to the test case traces (Section 9.1) demonstrates that a group of traces such as the IBM CICS trace, GE message trace, mean-change trace, and the ACM trace are modeled exceptionally well. In contrast, our model does poorly on the Star Wars trace. We discuss next various shortcomings in the model and the techniques we use to estimate and reduce the model parameters that introduce error in our modeling approach.

Error in Model: We have chosen a probabilistic approach to represent a trace. Consequently, a certain amount of error is attributable to the variance of the stochastic process that we use to model the trace. The definition of the PISP (Section 3) assumes that the joint probability mass function for the trace in one interval is independent of the joint probability mass function of the trace in any other interval. This assumption is violated in many traces that are highly autocorrelated. This is another source of error in our model.

Error in Parameter Estimation: We estimate the probability of occurrence of a state in the trace by the rate of occurrence of that state. The rate of occurrences of a state are estimated as the slopes of a piecewise linear regression fitted to the REG path for the state (see Section 4). The estimates of probabilities are consequently only as good as the functions fitting the REG path. Thus, if the REG paths are smooth curves instead of straight lines our estimation procedure will contribute to the error in the model.

Error in Model Reduction: Model reduction is accomplished by reducing the number of unique states as well as by reducing the number of partitioning events per state. Consider the former first. Removing a state from the trace causes the probability of occurrence of this state to be shared between neighboring states. If the state being removed happens to be an frequently occurring state this can contribute significantly towards the error in the final model.

Reducing the number of partitions implies fitting the REG path for a state with a piecewise linear regression that has fewer straight lines. Clearly, the probability estimates thusly obtained are approximate. This contributes towards the error in the final model.

12 Conclusions

The fundamental idea explored in this paper was that of representing the time-dependent behavior in a trace file by fitting a function to a graph path representing the rate of occurrence of each state in a trace. A general polynomial could be used, but instead we observed empirically that the paths in many traces consist of straight line segments; thus partitioning the time interval of the trace into segments, and using regression for fitting lines within each segment, was used here. The derivative of the paths — simply the slope of each straight line — lead to an estimate of the probability mass function for each state. Collecting the estimates for all intervals lead to a time-dependent model, called the Piece Independent Stochastic Process (PISP).

The accuracy and parsimony of the method was explored using twelve test traces. The validation results (Section 9.3) show that in a third of our test cases the original trace is indistinguishable from traces synthesized from the model according to a statistical test of homogeneity. At best our model is 7/1000 the size of the original trace, but typically the model requires around 40%. The tests revealed a weakness of our model: it is inaccurate in traces that display long range autocorrelation; this is a result of our assumption of independence of intervals.

Several open problems are suggested by this work:

- Our modeling approach builds a stochastic process model of trace data based on one sample realization. This approach is necessary in many instances because many traces contain observational data that cannot be replicated. In general, however, a model based on an ensemble of traces from the same system will be a better predictor behavior of that system. Building on our approach the REGs of individual traces in an ensemble can be used to build a model of the ensemble.

- Our model is appropriate for trace data which can be considered as a time series such that the index parameter of the time series is equally spaced and monotonically increasing. Thus our model ignores the actual time stamps associated with the trace data. The analysis of trace data where the index parameter is not equally spaced is an open research area.

- A key assumption in the piecewise independent stochastic process model is that the joint probability mass function for the trace in one interval is independent of the joint probability mass function of the trace in any other interval. In reality this assumption is violated by many traces. The definition of the piecewise independent stochastic process model should be augmented such that the correlation structure of the trace factors into the model.

- We use a collection of straight lines to fit a REG path. Instead, polynomial functions can be fitted to REG paths. This approach may especially be appropriate when REG paths are smooth high order curves.

- State space explosion is a major problem in building models of trace data. We present one technique to reduce the number of unique states in the model. Alternative effective techniques to reduce state space size will have a significant impact on controlling the size of our model.

All the tools needed to construct a piecewise independent stochastic process model from traces are implemented in the CHITRA trace analysis tool. CHITRA can be executed with a Java-capable World-Wide Web browser from http://www.cs.vt.edu/~chitra.

References

[ADM92] M. Abrams, N. Doraswamy, and A. Mathur. Chitra: Visual analysis of parallel and distributed programs in the time, event, and frequency domain. *IEEE Trans. on Parallel and Distributed Systems*, 3(6):672–685, November 1992.

[ARM96] M. Abrams, R. Ribler, and A. Mathur. Two performance tool design issues and chitra's solutions. In *SIGMETRICS Symposium on Parallel and Distributed Tools*. ACM, Philadelphia, PA, May 1996.

[Ber94] J. Beran. *Statistics for Long-Memory Processes*. Chapman Hall, New York, 1994.

[BJ76] G. E. P. Box and G. M. Jenkins. *TIME SERIES ANALYSIS forecasting and control, Revised Edition*. Holden-Day, Oakland, California, 1976.

[FSZ83] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tuning of Computer Systems*. Prentice-Hall, London, 1983.

[Gar93] M. W. Garrett. *Statistical Analysis of a Long Trace of Variable Bit Rate Video Traffic*. PhD thesis, Columbia Univ., 1993. Chapter IV.

[GH93] A. J. Goldberg and J. L. Hennessy. Mtool: An integrated system for performance debugging shared memory applications. *IEEE Trans. on Parallel and Distributed Systems*, 4(1):28–40, January 1993.

[HL96] D. P. Heyman and T. V. Lakshman. Source models for vbr broadcast-video traffic. *IEEE Trans. on Networking*, 4(1):40–48, February 1996.

[HTL92] D. P. Heyman, Ali Tabatabai, and T. V. Laskshman. Statistical analysis and simulation study of video teleconference traffic in atm networks. *IEEE Trans. on Circuits and Systems for Video Technology*, 2(1):49–58, March 1992.

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[Mat96] A. Mathur. *A Stochastic Process Model for Transient Trace Data*. PhD thesis, Computer Sci. Dept., Virginia Tech, October 1996.

[Ott88] L. Ott. *An Introduction to Statistical Methods and Data Analysis*. PWS-Kent, Boston, 3rd edition, 1988.

[PF94] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. In *Proc. SIGCOMM94*, pages 257–268. ACM, London, August 1994.

[RW93] Diane T. Rover and Abdul Waheed. Multiple-domain analysis methods. In *ACM/ONR Workshop on Parallel Debugging and Performance*, proceedings appeared in *ACM SIGPLAN Notices*, 28(12), Dec. 1993, pages 53–63. ACM, San Diego, May 1993.

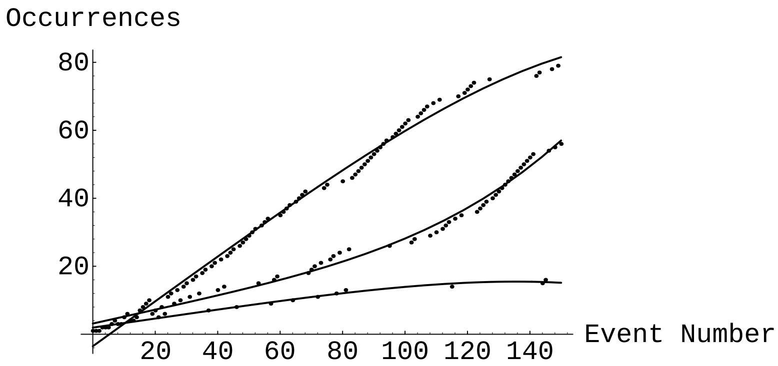


Figure 1: Polynomial Fit for the WWW Trace REG (top curve is directory, middle is graphics, bottom is text and HTML)

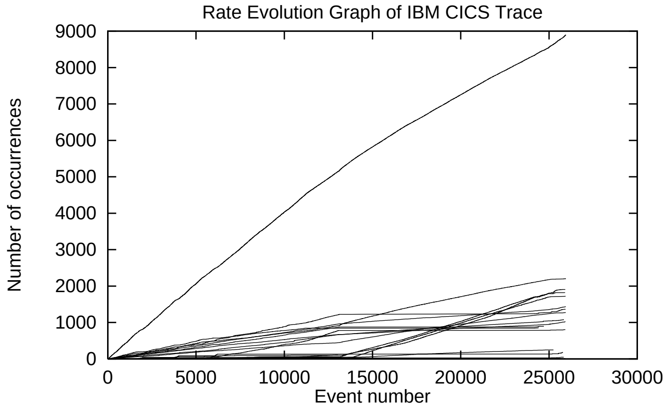


Figure 2: REG for the IBM CICS Trace

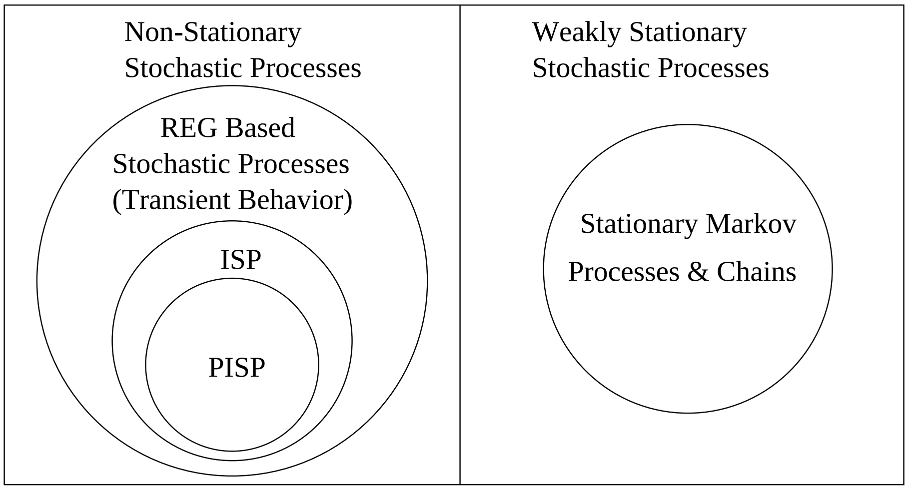


Figure 3: Stochastic Process Models Based on the REG

Table 1: Description of Test Case Traces

Name	State	Data Type	Num States	Length
GE message trace	Processor Number	Categorical	31	409
Dining Philosophers trace	Philosopher Numbers	Categorical	597	5857
UoS Ethernet trace	Host Name	Categorical	11	3772
IBM CICS trace	Transaction Type	Categorical	24	25953
GE memory trace	Memory Address	Categorical	48	376
WWW trace	File Type	Categorical	7	9139
Ingalls trace	Packet Size	Numerical	199	4076
ACM trace	Packet Size	Numerical	18	1739
Star Wars trace	Packet Size	Numerical	236	170998
Periodic trace	Numerical State	Numerical	2	100
Time-trend trace	Numerical State	Numerical	2	999
Mean-change trace	Numerical State	Numerical	4	999

Table 2: Cache Hit Trace

Cache Hit Rate																	
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X_t	H	H	H	H	M	M	M	M	H	H	H	H	M	H	M	H	M

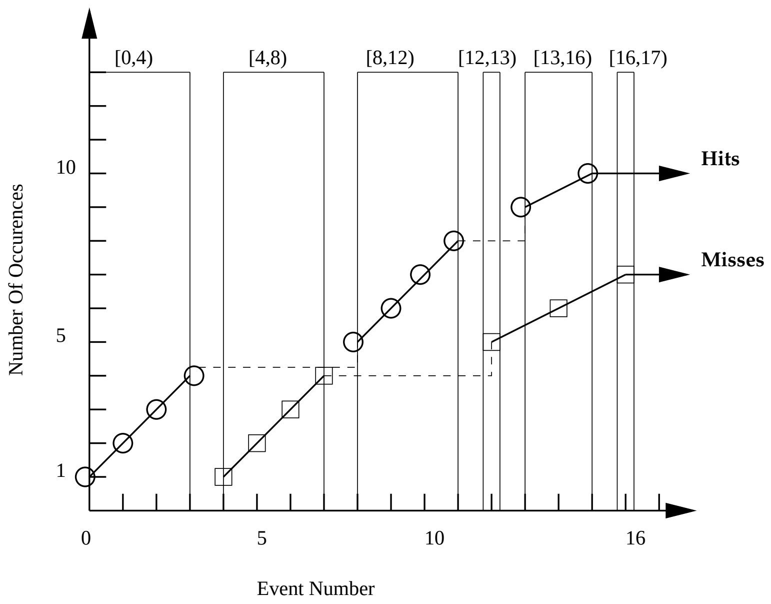


Figure 4: REG for the Cache Trace

Table 3: Rate Vectors Φ_t

Rate Vectors Φ_t $(t \in \mathbf{R})$		
Φ_t	$x_i = H$	$x_i = M$
Φ_0	1	0
Φ_4	0	1
Φ_8	1	0
Φ_{12}	0	1
Φ_{13}	0.5	0.5
Φ_{16}	0	1
Φ_{17}	0	0

Table 4: Joint Probability Mass Function

Time dependent *Pmf* $(\mathbf{\Phi})$		
Interval	$P(x_i = H)$	$P(x_i = M)$
[0,4)	1	0
[4,8)	0	1
[8,12)	1	0
[12,13)	0	1
[13,16)	0.5	0.5
[16,17)	0	1
$[17,\infty)$	0	0

Table 5: Partitioning Events and Corresponding Rates for State H

Partitioning Events and Rates						
Partitioning Event $(t \in \mathbf{R_0})$	0	4	8	12	13	16
Rate $(\phi_{0,t})$	1	0	1	0	0.5	0

Table 6: Partitioning Events and Corresponding Rates for State M

Partitioning Events and Rates					
Partitioning Event $(t \in \mathbf{R_1})$	0	4	8	12	17
Rate $(\phi_{1,t})$	0	1	0	0.5	0

Table 7: Rate Vectors Φ^0

Rate Vectors Φ_t $(t \in \mathbf{R_0})$		
Φ_t	$\Phi_{0,t}$	$\Phi_{1,t}$
Φ_0	1	0
Φ_4	0	1
Φ_8	1	0
Φ_{12}	0	1
Φ_{13}	0.5	0.5
Φ_{16}	0	1

Table 8: Rate Vectors Φ^1

Rate Vectors Φ_t $(t \in \mathbf{R_1})$		
Φ_t	$\Phi_{0,t}$	$\Phi_{1,t}$
Φ_4	0	1
Φ_8	1	0
Φ_{12}	0	1
Φ_{17}	0	0

Table 9: Compact REG for State H

Compact REG for *Hits* $(x_i = H)$						
Event Number (t)	0	4	8	12	13	15
Number of Occurrences $(g(x_i, t))$	1	4	4	8	9	10

Table 10: Compact REG for State M

Compact REG for *Misses* $(x_i = M)$				
Event Number (t)	4	8	12	16
Number of Occurrences$(g(x_i, t))$	1	4	5	7

Table 11: Partitioning Events and Corresponding Rates for State H

Partitioning Events and Rates		
Partitioning Event$(t \in \mathbf{R_0})$	0	16
Rate$(\Phi_{H,t})$	0.59	0

Table 12: Partitioning Events and Corresponding Rates for State M

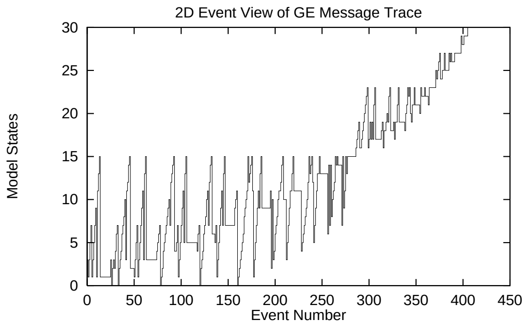
Partitioning Events and Rates		
Partitioning Event$(t \in \mathbf{R_1})$	4	17
Rate$(\Phi_{M,t})$	0.48	0

Table 13: Results of the Kruskall-Wallis Test (Ordered by % of Segments that Pass the Test)

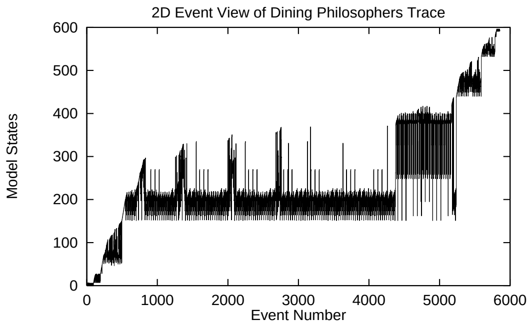
$\alpha = 0.01$, $\chi^2 = 6.6349$					
% of Segments that Pass the KW Test					
Name	Seed Index				
	1	*11*	*21*	*53*	*99*
Mean-change trace	100	100	100	100	100
Periodic trace	100	100	100	100	100
GE Message trace	100	90	90	100	90
ACM trace	90	100	90	80	90
Ingalls trace	80	90	90	90	90
IBM CICS trace	90	80	80	90	90
WWW trace	70	90	70	90	90
UoS Ethernet trace	80	80	80	80	70
Time-trend trace	80	80	70	80	70
GE Memory trace	70	80	60	60	80
Dining Philosophers trace	40	40	40	40	40
Star Wars trace	40	30	40	30	40

Table 14: Model Size for Test Case Traces (Ordered by Model Size as a % of Trace Size)

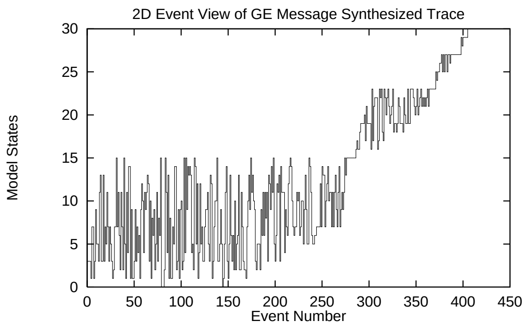
Storage in bytes			
Name	Trace Size	Model Size	Model Size as % of Trace Size
WWW Trace	18281	120	0.7
Time-trend Trace	2001	229	11.4
Periodic Trace	201	30	14.9
Star Wars Trace	620325	10309	16.6
UoS Ethernet Trace	7740	1840	23.7
IBM CICS Trace	57279	16656	29.1
Mean-change Trace	2001	773	38.6
Ingalls Trace	16221	6480	39.9
Dining Philosophers Trace	22890	9477	41.4
GE Message Trace	1051	562	53.5
ACM Trace	3997	2178	54.0
GE Memory Trace	1047	887	84.7



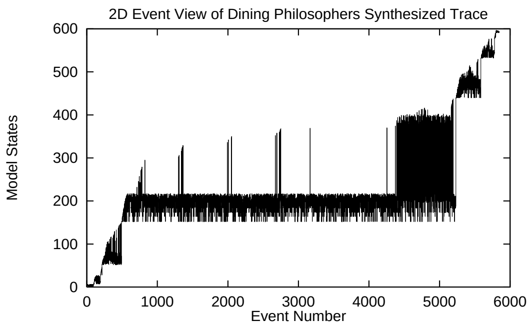
(a)



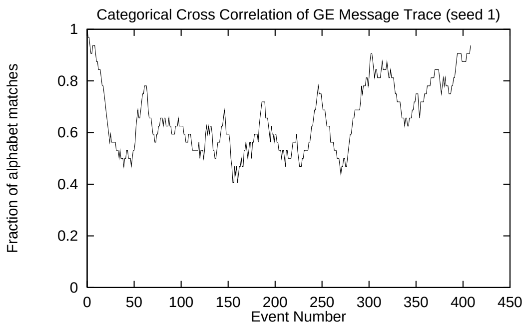
(b)



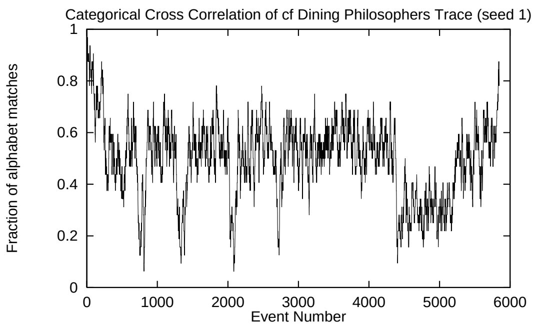
(c)



(d)

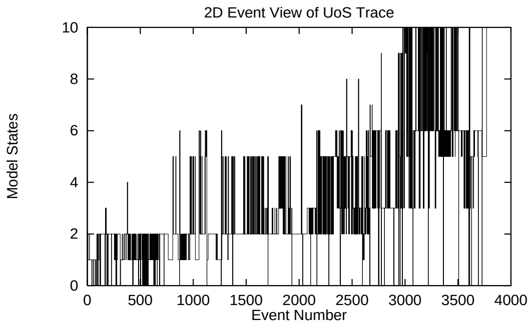


(e)

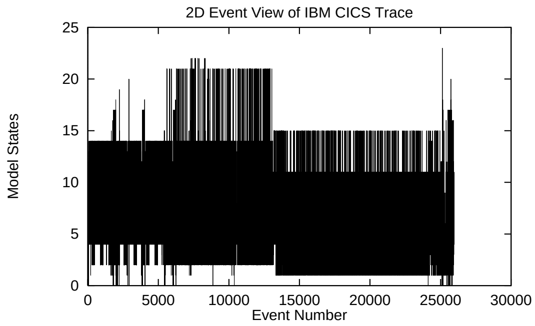


(f)

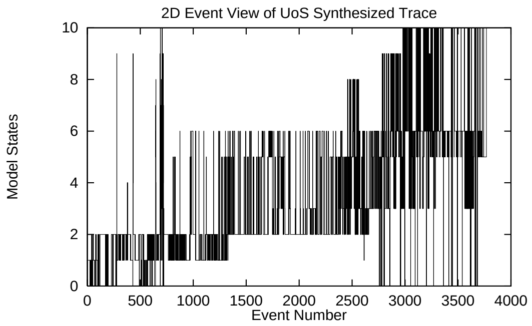
Figure 5: 2D Views: (a) GE message trace, (b) Dining Philosophers trace, (c) GE message synthesized trace, (d) Dining Philosophers synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) GE message original versus synthesized trace, (f) Dining Philosophers original versus synthesized trace



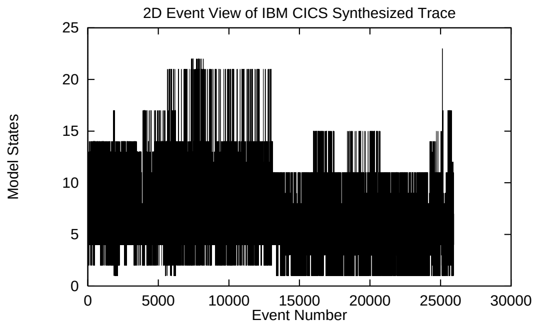
(a)



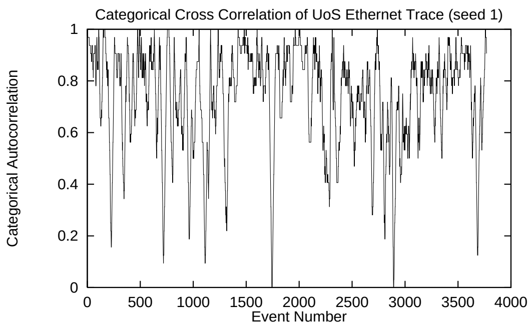
(b)



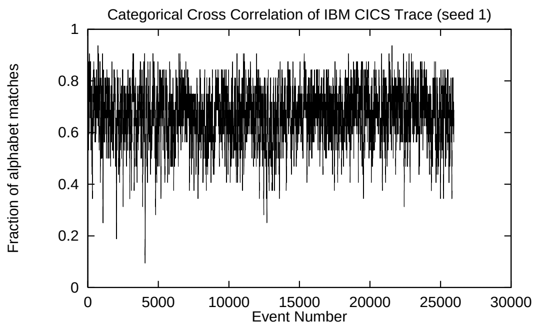
(c)



(d)

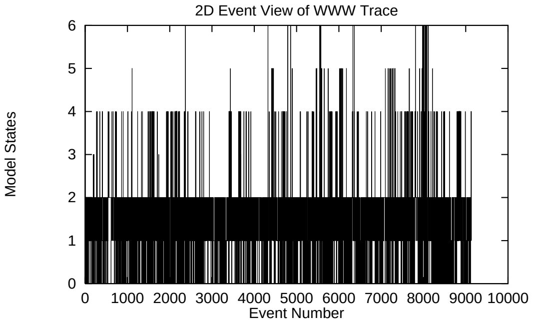


(e)

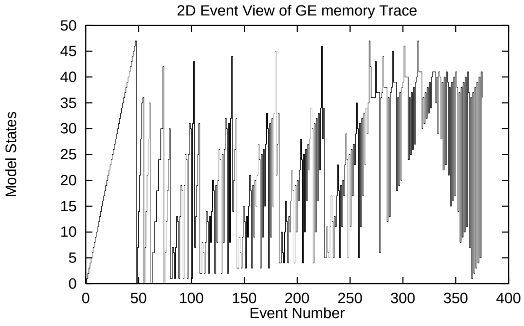


(f)

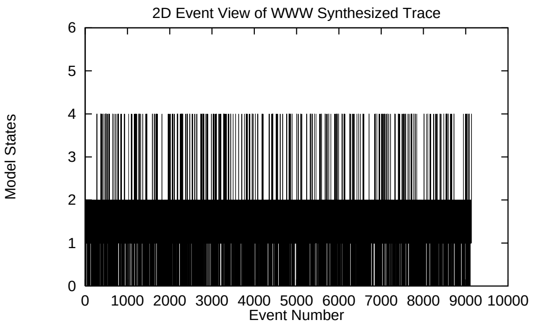
Figure 6: 2D Views: (a) UoS trace (b) IBM CICS trace, (c) UoS synthesized trace, (d) IBM CICS synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) UoS original versus synthesized trace, (f) IBM CICS original versus synthesized trace



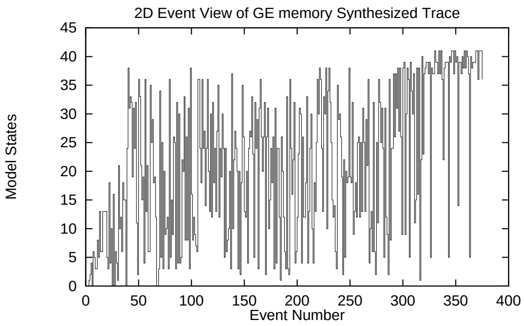
(a)



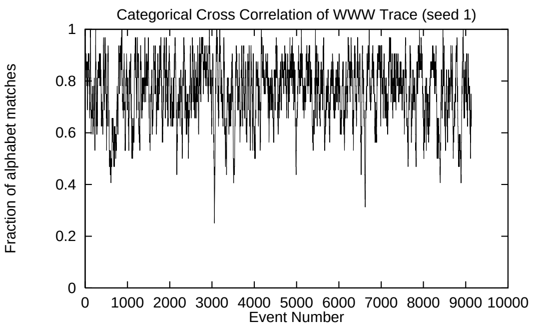
(b)



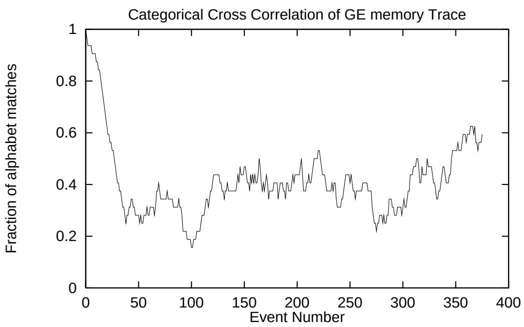
(c)



(d)

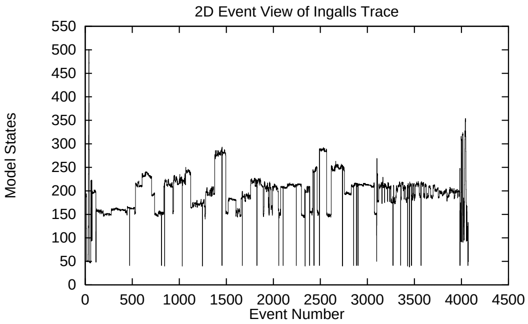


(e)

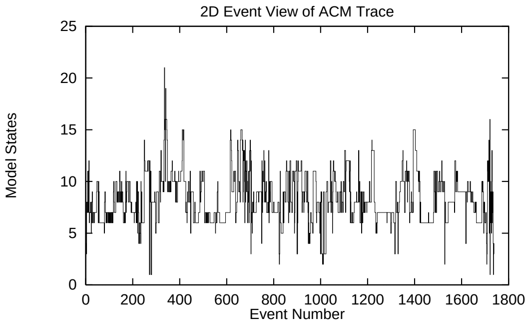


(f)

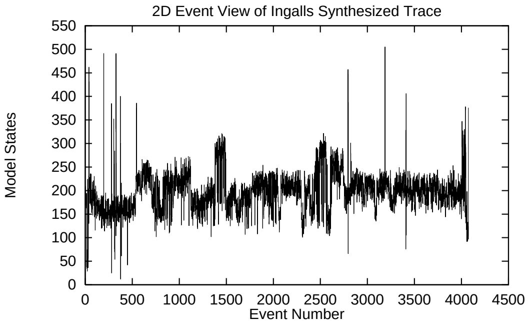
Figure 7: More 2D Views: (a) WWW trace (b) GE memory trace, (c) WWW synthesized trace, (d) GE memory synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) Periodic original versus synthesized trace, (f) GE memory original versus synthesized trace



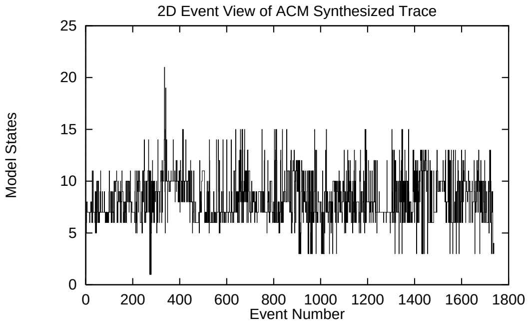
(a)



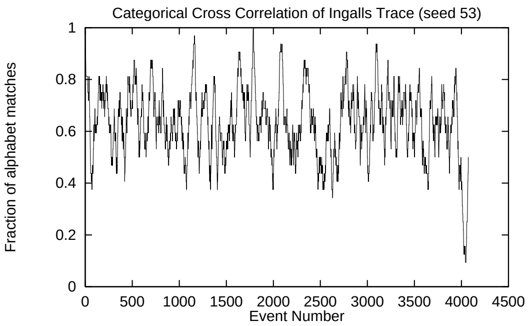
(b)



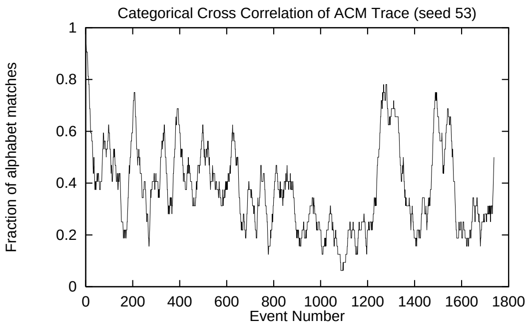
(c)



(d)

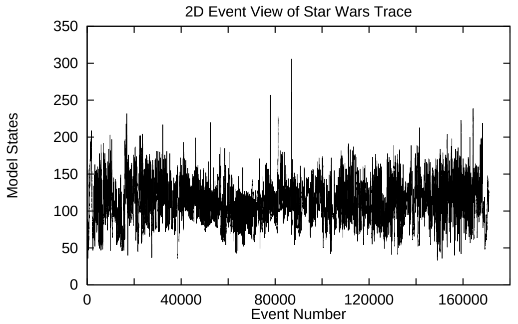


(e)

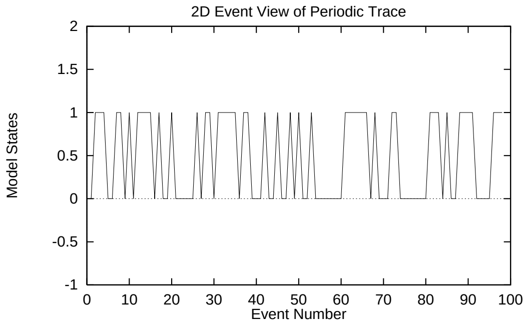


(f)

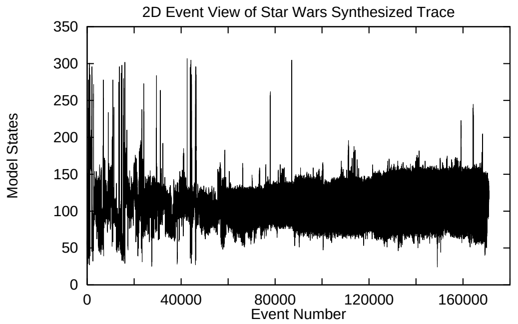
Figure 8: More 2D Views:(a) Ingalls trace, (b) ACM trace, (c) Ingalls synthesized trace, (d) ACM synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) Ingalls original versus synthesized trace, (f) ACM original versus synthesized trace



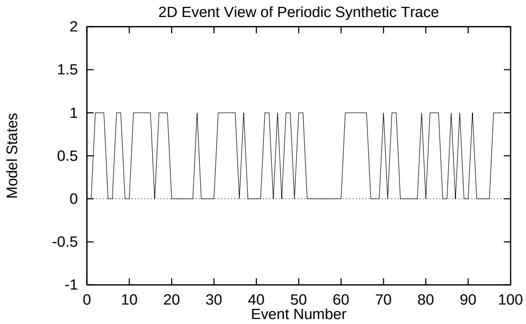
(a)



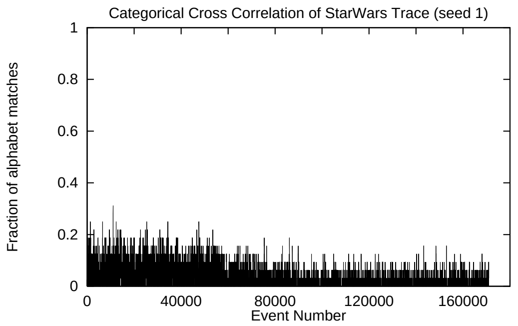
(b)



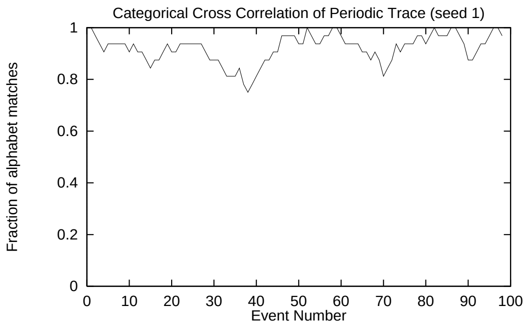
(c)



(d)



(e)



(f)

Figure 9: More 2D Views: (a) Star Wars trace, (b) Periodic trace, (c) Star Wars Synthesized trace, (d) Periodic Synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) Star Wars original versus synthesized trace, (f) Periodic original versus synthesized trace

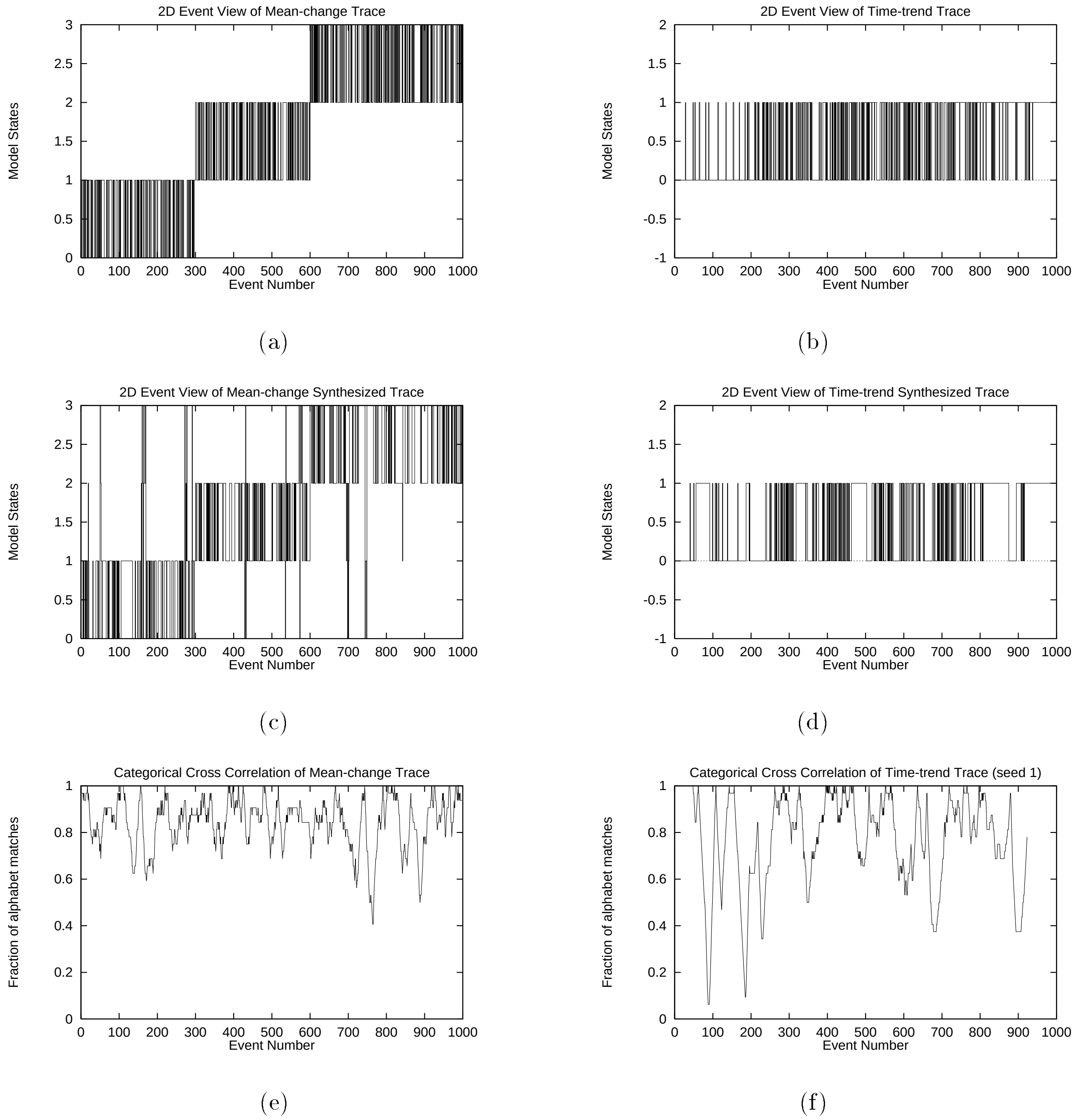
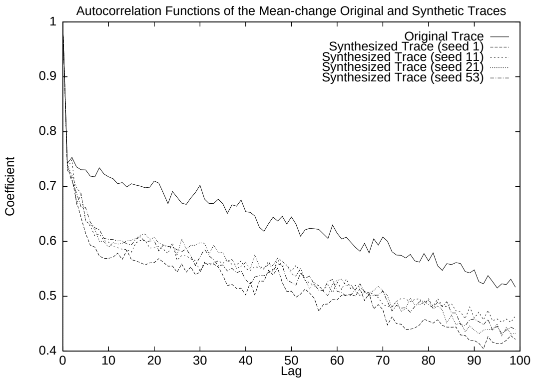
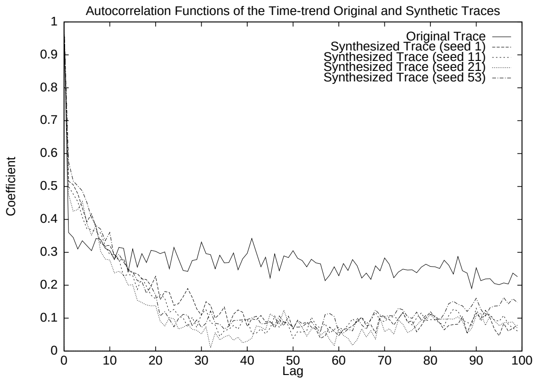


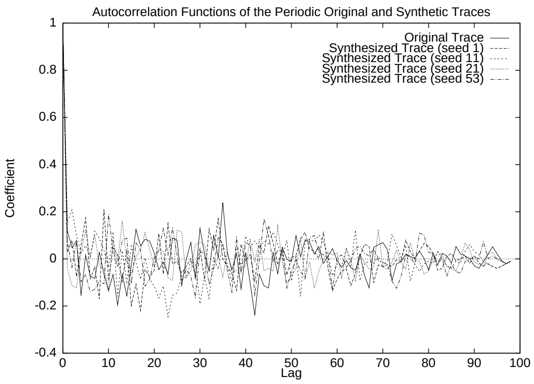
Figure 10: More 2D Views: (a) Mean-change trace (b) Time-trend trace, (c) Mean-change synthesized trace, (d) Time-trend synthesized trace. Categorical Alphabet Match Cross Correlation Graphs: (e) Mean-change original versus synthesized trace, (f) Time-trend original versus synthesized trace



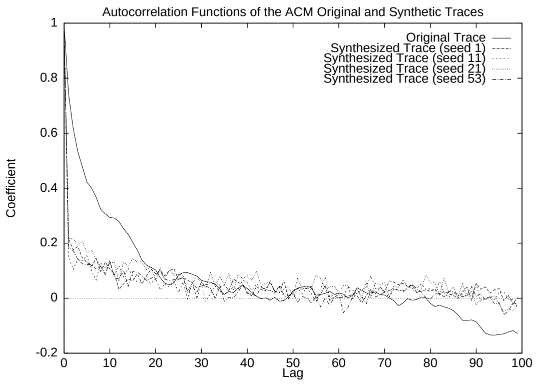
(a)



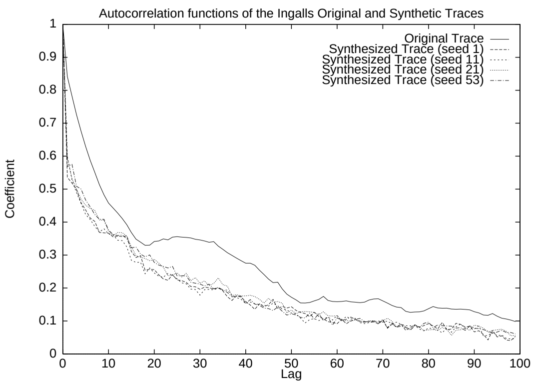
(b)



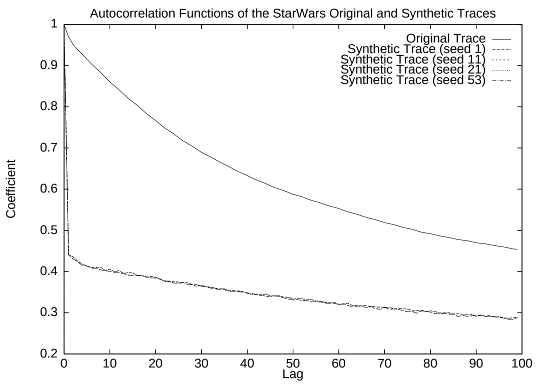
(c)



(d)



(e)



(f)

Figure 11: Autocorrelation Plots: (a) Mean-change trace (b) Time-trend trace, (c) Periodic trace, (d) ACM trace, (e) Ingalls trace, (f) Star Wars trace