Beyond Software Performance Visualization*

Marc Abrams Timothy J. Lee Horacio T. Cadiz
Krishna Ganugapati

Virginia Polytechnic Institute and State University
Department of Computer Science
Blacksburg, VA 24061-0106
(703) 231-8457 abrams@cs.vt.edu

Computer Science Department TR 94-07
Systems Research Center SRC-94-005

February 16, 1994

Abstract

Performance visualization tools of the last decade have yielded new insights into the behavior of sequential, parallel, and distributed programs. However, they have three inherent limitations: (1) They only display what happened in *one* execution of a program. (This is dangerous when analyzing concurrent applications, which are prone to non-deterministic behavior.) (2) A human uses one or more bandwidth-limited senses with a visualization tool. (This limits the scalability of a visualization tool.) (3) The relationship of "interesting" program events are often separated in time by other events; thus discerning time dependent behavior often hinges on finding the "right" visualization – a possibly time-consuming activity. CHITRA93 complements visualization systems, while alleviating these limitations. CHITRA93 analyzes a set (or ensemble) of traces by combining the visualization of a few traces with a statistical analysis of the entire ensemble (overcoming (1)); and reduces the ensemble to empirical models that capture the time dependent relationships of "interesting" program events through application, programming language, and computer architecture independent analysis tools (addressing (2) and (3)). CHITRA93 incorporates: transforms, such as aggregation, that simplify the ensemble and reduce the state space size of the models generated; a user interface that allows some transforms to be selected by editing the visualization with a mouse; homogeneity tests that allow partitioning of an ensemble; an efficient semi-Markov model generation algorithm whose computation time is linear in the sum of the lengths of the traces comprising the ensemble; and a CHAID-based model that can fathom non-Markovian relationships among transitions in the traces. The use of CHITRA93 is demonstrated by partitioning ten parallel database traces with nearly 8,000 states into two homogeneous subsets, each modeled by a 20 state irreducible, periodic (non-Markovian) stochastic process.

CR categories: C.4 [Performance of Systems]: Modeling techniques, D.2.8 [Software]: Metrics – Performance measures, D.1.3 [Programming Techniques]: Concurrent programming

Keywords: Parallel software, empirical performance modeling, visualization, traces

*This work was supported in part by National Science Foundation grant NCR-9211342 and a summer 1993 grant from the U.S. Navy NSWC administered by the VPI&SU Systems Research Center

1 Introduction

A key impediment to widespread use of parallel and distributed systems is our limited ability to analyze concurrent software performance. This paper presents a new analysis approach, in which empirical models are constructed from a set, or *ensemble*, of trace files collected during multiple program executions. The method is coupled with well-known trace visualization methods, and implemented in a system called CHITRA93[1]. The CHITRA93 paradigm is "visualize some, transform and model all." A user visualizes select traces, selects transforms that are applied to all traces based on visualization, and then uses one or more models of all traces to gain insight into the entire data set. Thus CHITRA93 uses visualization only as a window to examine pieces of a massive data set.

Visualization systems of the last ten years (e.g., [5, 9]) have yielded much insight into time dependent program behavior. Performance visualization systems are becoming popular tools for commercial parallel processors. Yet visualization systems have failed to achieve a breakthrough in helping people write concurrent programs. We attribute this failure to three inherent limitations:

1. Visualization shows what happened in only *one* execution of a program. Execution behavior varies with the input data, the background load, and the hardware configuration (e.g., memory size, number of processors). This problem is acute for concurrent programs that run on asynchronous processors, because nondeterminism creates variations whenever a program is rerun.

2. A human uses one or more bandwidth-limited senses with a visualization tool, such as sight to view a computer monitor. In addition the ability of the brain to interpret sensual input has some limit. Therefore there is a limit on the bandwidth of information that a visualization can communicate to a user. Originally visualization had a dramatic impact because it offered an enormous bandwidth increase over a 24 row by 80 column character text display of a trace file. However the visualization bandwidth limitation precludes scaling the method up to thousands or millions of processors. Visualization research today seeks ways to push the limit upward; examples include data sonification and immersing a person in the data through virtual reality. But a human has five senses with limited input bandwidth, so a hard limit will eventually be reached.

[1] *Chitra* is a Sanskrit word meaning a beautiful or pleasing picture.

3. Insights into time dependent program behavior often are apparent only if the "right" presentation (e.g., visualization, animation, or sonification) of the trace data is constructed. For example, the Pablo designers discovered an old program bug in a trace file visualized many times only after adding a particular sonification [8]. Often "interesting" events may be separated in time by many other events. Unfortunately, deriving the right presentation can be labor intensive, and sometimes the labor can be amortized only over a few programs.

Our motivation is to design a system that complements visualization systems, but addresses these three limitations. We have three objectives, matching the above three limitations: (1) Analyze a *set* of trace files; (2) sharply *reduce* the amount of information that needs to be communicated to a human, thereby operating within the fixed bandwidth of the output device; and (3) find a suitable representation for time dependent behavior that applies to any application program, programming language, or computer architecture.

Our approach is embodied in a system called CHITRA93, based on our earlier system CHITRA91 [1]. CHITRA93 combines visualization, trace transformation, and model generation modules into an integrated system controlled through a Motif graphical user interface. The key to achieving our three objectives is the augmentation of visualization techniques with the ability to build empirical models of the time dependent behavior of a system that fit multiple observations of the system. A model has many possible uses: to represent observed data parsimoniously (or compactly), to compare behavior of different system configurations, to act as a synthetic workload for a simulation of the system being observed, to validate assumptions made about the system being observed, to characterize non-deterministic system behavior, to identify and diagnose performance and correctness problems in a system, to test correct system operation, and to explore the physics of computation.

1.1 Problem Statement

Certain terminology is required to make the problem statement precise. The behavior of a system can be described by enumerating the sequence of *states* it follows during execution. A *system state* is an n-tuple enumerating the values of n system attributes at some point during its execution. For example, a state

may represent the section of source code that each process in a parallel program is currently executing, the value of one or more program variables, the memory address last referenced, or a combination of these. The choice of state is governed by the objective of a performance study. A system is described by the sequence of system states that it traverses during operation, termed a *system state sequence* (SSS). A *timed SSS* is obtained by augmenting an SSS with the time at which each state is entered. Let L denote the number of states in an SSS; $S_0, S_1, \ldots, S_{L-1}$ denote an SSS; and E_i denote the entry time of state S_i $(0 \leq i < L)$. Then the corresponding timed SSS is a sequence of ordered pairs (E_0, S_0), (E_1, S_1), $\ldots$, (E_{L-1}, S_{L-1}), where $E_0, E_1, \ldots, E_{L-1}$ is a strictly increasing sequence. Each pair of successive states (S_k, S_{k+1}) (for $0 \leq k < L-1$) is a *transition*. Transition number k is *out* of state S_k and *into* S_{k+1}. An *event* causes a transition; the *event number* of S_i is i. Henceforth in this paper, a *trace file* is defined to be a timed SSS, and an *ensemble* is a set of timed SSS's.

The last term required for the problem statement is from statistics: *homogeneous*. In statistics, one must identify a universal population, and then calculate statistics of a sample of the population. In CHITRA93, the universal population is the set of all possible execution sequences of a system, and the sample is the execution sequence in a single trace file. A set of trace files are homogeneous with respect to some system attribute if the samples of those attributes contained in the traces are drawn from the same population or from identical distributions.

This paper describes CHITRA93's solution to two problems arising in ensemble analysis:

P1: Determine if the ensemble is homogeneous. If not, partition it into mutually exclusive, exhaustive, and homogeneous subsets called *sub-ensembles*.

P2: Construct a model fitting the traces comprising each sub-ensemble. The model should yield insight into the ensemble by compactly summarizing the dynamic behavior of the sub-ensemble, distinguishing whether patterns of system states exist or whether the state sequence is completely random.

Establishing homogeneity (P1) has two principal benefits. First, it assists in answering whether there is a sufficient number of trace files in the ensemble to permit statistical inference about the universal population of all possible executions of the system. Second, it permits classification of *replicated components*

in a massively parallel system, such as processes. If each trace contains a state sequence representing a single component, then a homogeneous set of traces identifies which components behave similarly.

1.2 How CHITRA93 Solves the Problem

CHITRA93 uses a four step procedure to solve problems P1 and P2:

Step-A: Simplify the ensemble by applying transforms. *Simplification* is the process of reducing one or both of the following: the number of model states required to represent the ensemble (in Step-D) and the duration of system time that the ensemble represents. A *transform* maps each trace in an ensemble to a simpler trace. A user may visualize as few as one trace of an ensemble to select a transform, and CHITRA93 automatically applies that transform to all traces in the ensemble. Information about the results of transforming non-visualized traces are then presented.

Step-B: Test the hypothesis that the traces in an ensemble are homogeneous.

Step-C: For non-homogeneous ensembles, select partitions using one or more of the following: visualization, the result of applying transforms, the structure of the model that CHITRA93 would generate, or statistical tests.

Step-D: Construct one model fitting each sub-ensemble.

This paper addresses *building* models through the four steps above. *Using* models for performance problem diagnosis is illustrated in [1].

2 Example: Parallel Database

CHITRA93 will be illustrated in §3 using a parallel database with record locking. The database consists of twenty-eight records, each with two keys: a *gender* key (two valued), and a *geographic region* key (fourteen valued). There are sixteen transaction types, which are assigned ids in the range zero through fifteen. Two transaction types use all records with a common gender; the remaining fourteen exclusively use all records with a common geographic region. To prevent deadlock, records are assigned ids zero through twenty-seven, and transactions acquire records in ascending id order. Fig. 1 specifies which transaction types use which records. For example, transaction type 02 acquires the records for all females and males in region zero (F0 and M0). Transactions are executed on sixteen processors of a parallel computer.

Each processor executes one non-preemptive process that invokes one transaction type.

Each process repeats the following cycle: wait for a long think time (process state T), then initiate a long transaction (consisting of acquiring locks (A), exclusively using locked records (L), and releasing locks (R)), then think a short time (t), and then initiate a short transaction (consisting of acquiring locks (A), exclusively using locked records for a short time (l), releasing locks (R)). "Long" has twice the duration of "short." The system state is the concatenation of all process states. Thus in state (A,A,L,L,L,L,L,L,L,L,L,L,L,L,L,L), also written as A^2L^{14}, the two gender transactions (0 and 1) are waiting to acquire locks, while the rest (2 to 15) are holding records for a long period.

The database is implemented as a C++ program using the Presto 0.4 thread package [2] and running on 16 80386-based processors of a Sequent Symmetry multiprocessor computer. Each lock is implemented as a (non-preemptive) spin-lock associated with one shared memory address. The think and record use periods are simulated by loops that execute noop instructions for a specified period of time. The instrumentation traces state transitions in all processes for a small portion of the program run time.

In terms of analyzing locking behavior in the parallel database example, problem P1 indicates whether the number of traces available – 10 – is sufficient. (An alternative to the study in this paper is to determine if all transactions are homogeneous in the sense of experiencing similar lock acquisition delays, imposing equal processor loads, or gaining fair access to records. For example, are the two gender transactions homogeneous?) Problem P2 will analyze the dynamic use of records by all transactions. Is a probabilistic description appropriate, or does the program tend to display deterministic behavior?

3 Sample Analysis Session with CHITRA93

3.1 Preliminaries: Opening and Mapping Traces

3.1.1 Opening an Ensemble

During a CHITRA93 session, a user identifies which trace files constitute ensembles for the analysis by opening trace files, as one would open documents in a word processor with a graphical user interface. However, unlike using a word processor, the user can click on and hence select *multiple* filenames in the standard Motif file selection dialog box. Multiple selection defines an ensemble. Ensembles are identified

as 0, 1, ... Traces within ensemble e are numbered $e.0$, $e.1$, ...

When a user opens an ensemble, CHITRA93 loads an *instance* of the trace files in the ensemble. Each ensemble instance $\mathcal{E}$ will have multiple *versions*, denoted $\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_v, \ldots$. Version $\mathcal{E}_0$ is created when the ensemble is first opened; successive versions are created each time an ensemble is transformed (§3.2.2). Version $\mathcal{E}_0$ is identical to the original disk copy; other versions modify the loaded instance, not the original disk copy. A user can open multiple instances of the same ensemble, permitting each instance to be modified independently. This is useful for comparing the model obtained by different sequences of transforms.

CHITRA93's root window (not illustrated) contains a list box showing all ensembles. Operations on ensembles, such as transforming, opening views of, or closing an ensemble, may be performed by first selecting an ensemble name in the list box of the root window and then selecting an operation from a root window menu.

3.1.2 Mapping Trace Records to Model States

Analysis of an ensemble instance with CHITRA93 starts with generating an *initial* mapping from system states to the states used in the empirical model generated later in Step-D.

Let $\mathcal{S}(\mathcal{E}_v)$ denote the set of all system states that appear in all traces in ensemble version $\mathcal{E}_v$. CHITRA93 views each system state S_i in a trace file as an experiment outcome. For each ensemble instance, CHITRA93 defines a random variable, $X_{\mathcal{E}_v}$, mapping all outcomes – the elements of $\mathcal{S}(\mathcal{E}_v)$ – to non-negative integers, called *model states*. Let $\mathcal{M}(\mathcal{E}_v)$ denote the range of $X_{\mathcal{E}_v}$. Letting the domain of t be the interval bounded by the first and last timestamp in a particular trace file in $\mathcal{E}_v$, then $X_{\mathcal{E}_v}(t)$ represents the trace file as the realization of a stochastic process. $X_{\mathcal{E}_v}$ is subscripted by $\mathcal{E}_v$ because CHITRA93 defines a different mapping for each ensemble instance, and modifies the mapping whenever a new ensemble version is created. Subscripts are dropped when the ensemble and version are clear from context. When an ensemble is originally opened, CHITRA93 generates an *initial* mapping $X_{\mathcal{E}_0}$ as follows. Informally, CHITRA93 assigns as model states successive non-negative integers the first time it encounters a system state when scanning each of trace files 0, 1, ... from start to end. Thus model state 0

is the first system state in trace number 0, and model state 1 is the next system state in trace 0 that does not duplicate the first system state. Formally, suppose that an ensemble contains n trace files, and that there are m unique system states in the n files. Create a list by concatenating the n SSS's in increasing trace number order (0 followed by 1, and so on). Next, scan the resulting file from start to end, and delete any system state that duplicates a system state occurring earlier in the list. The list must have length m. Let $s_0, s_1, \ldots, s_{m-1}$ denote the resultant list. (Note that $\mathcal{S}(\mathcal{E}_0) = \{s_0, s_1, \ldots, s_{m-1}\}$.) Then $X_{\mathcal{E}_0}$ is defined as $X_{\mathcal{E}_0}(s_i) = i$. Therefore the model state space, $\mathcal{M}(\mathcal{E}_0)$, is the integers $0, 1, \ldots, m-1$.

When ensemble $\mathcal{E}_0$ is loaded, CHITRA93 defines the initial mapping $X_{\mathcal{E}_0}$, and then represents each trace in $\mathcal{E}_0$ by two sequences: an *occupancy time sequence* and a *model state sequence*. Trace file (E_0, S_0), $(E_1, S_1), \ldots, (E_{L-1}, S_{L-1})$ is represented as occupancy time sequence $E_1 - E_0, E_2 - E_1, \ldots, E_{L-1} - E_{L-2}$ and model state sequence $M_0, M_1, \ldots, M_{L-2}$, where $M_i = X_{\mathcal{E}_0}(S_i)$. Note that the last system state, S_{L-1}, is dropped because its occupancy time cannot be computed.

Example: Consider Fig. 2. For simplicity of discussion, this Fig. 2 represents a three process database (processes 00, 01, and 02 in Fig. 1), not the sixteen process database described in §2. The window banner in Fig. 2 contains "Text view of 1.0." Hence the view shows one of CHITRA93's views – the text view – and the data being viewed is a single trace file – trace 0 in ensemble 1. Text view lists on successive lines, for each state S_i in a portion of the viewed trace file, the four quantities i, E_i, $X(S_i)$, and S_i. Thus the right hand column contains the SSS.

From Fig. 2, the first system state recorded when instrumentation was activated was (A,A,T), in which processes 0 and 1 were executing transactions that were waiting to acquire locks, and process 2 was in a long think period before initiating another transaction. The next state transition occurred at time 241, when process 1's transaction finished acquiring record locks and began using the locked records for a short period (state A,l,T). The first eleven states are shown (S_0=(A,A,T) through S_{10}=(T,T,t)). They are unique and hence correspond to model states 0,1,. . . , 10. For example, the ninth system state, (T,T,L), is mapped to model state 8. The vertical scroll bar on the right window edge indicates that the eleven states shown comprise a small portion of the trace file.

Note that while the number of *system* states is exponential in the number of components, the number

of *model* states, m, is bounded by the sum of the lengths of trace files in $\mathcal{E}_0$. For example, each database system state component can assume one of six process states (A, l, L, R, t, and T). Therefore a sixteen process program has almost 6^{16} system states (all 16-tuples in which each element is one of the process states T, t, A, L, l, or R representing legal states). In contrast the L system states in a trace file map to at most $L-1$ model states (Recall that the last system state is ignored; hence the term "-1.").

3.2 Step-A: Ensemble Simplification

Ensemble simplification is motivated by the potential for state space explosion in building an empirical model in Step-D: the number of model states required to represent an ensemble $\mathcal{E}$ when first opened is proportional to the sum of the lengths of traces in $\mathcal{E}$. The database ensemble has 7896 system states. Other traces – such as memory address reference or network traffic traces – may require tens or hundreds of thousands of model states. To gain insight into an ensemble, or to derive a compact model of an ensemble, the ensemble ideally should be simplified to a constant number of states, usually a few tens of model states. Transforms exist to achieve this. A constraint on simplification is that calculation of a performance measure from the original and simplified ensembles should differ by no more than a predetermined amount appropriate for the purpose of the resultant model. Some transforms are specified through trace visualization; hence CHITRA93's views are first described.

3.2.1 Open Views of Select Traces

Text view (Fig. 2) is one of nine views of traces, sub-ensembles, or ensembles in CHITRA93. The other views are two-dimensional (2D) plots ("2D-time," plotting model state versus time, and "2D-event," plotting model state versus event number), FFT ("FFT-time," displaying a periodogram of the 2D-time graph, and "FFT-event," displaying a periodogram of the 2D-event graph), homogeneity ("Adjacency matrix," displaying a composite view of whether elements of $\mathcal{E}$ have the same transitions, and "Kruskal-Wallis" (KW), displaying the result of testing the hypothesis of homogeneity of the means of occupancy time in one model state), and models ("Semi-Markov," displaying a semi-Markov model of $\mathcal{E}$, and "CHAID-based," displaying the best predicting past or future position for each state in $\mathcal{E}$).

To open a view, one selects an ensemble in the root window list box described in §3.1.1, and then

one of the nine views from the "View" menu. The result is a dialog box (not illustrated) requesting two pieces of information. First, there is a list box listing the traces comprising the ensemble, through which the user specifies which traces the view should represent. Selecting all traces views the entire ensemble. Second, there is a pair of radio buttons marked "Combined" and "Separate" to select whether the selected traces should be viewed in a single window, or each trace in a separate window, respectively. For example, a separate view of 3 traces in text view opens 3 text windows, with one trace displayed in each window. On the other hand, a combined view of 3 traces in the semi-Markov model view results in one window with a semi-Markov model that fits the three selected traces. Often it is useful to open both separate and combined views that include the same trace. For example one could compare the semi-Markov model of a single trace to the model of the entire ensemble by opening a separate view of one trace and a combined view of the ensemble.

Examples of Views: Shown in Figs. 3 and 4 are the 2D time views corresponding to the 3 and 16 transaction databases, respectively. (Several other views are illustrated later.) Fig. 3 is the same trace shown in text view (Fig. 2). The x-axis is continuous and represents the time interval represented by the trace displayed. The y-axis is discrete and represents integers ranging from 0 to the maximum element of $X_{\mathcal{E}_0}$. A y-axis coordinate of i represents model state i. Thus, referring to the text view (Fig. 2), y coordinates 0, 1, and 2 represent model states (A,A,T), (A,l,T), and (A,R,T). The y-axis labels can be displayed either as system states (Fig. 3) or model states (Fig. 4). In Fig. 4 the y-axis label "1256" indicates that the model states plotted are in the range [0,1256] – thus there are as many as 1257 model states for this trace. (Some values x, for $0 \leq x \leq 1256$, may not occur in the displayed trace, but do occur in another trace of the ensemble. The entire ensemble contains 7896 states.) The horizontally plotted lines represent a period during which the system remains in the corresponding y-axis state. The horizontal line length equals the state occupancy time. Element M_i of the model state sequence is represented by line segment $(E_i, M_i), (E_{i+1}, M_i)$. Thus the first horizontal line has endpoints (0,0) and (241,0) because the system was in model state 0 (A,A,T) between times 0 and 241. Vertical plotted lines, such as the line with end points (241,0) and (241,1), represent state transitions. A cross is plotted at each point representing state entry, such as (0,0) and (241,1). Some vertical lines have multiple crosses

meaning that a succession of rapid state transitions occurred that are represented by the same column of pixels in the window.

3.2.2 Transformation

To reduce the number of states required to model the ensemble, a user applies transforms that aggregate model states. A transform is invoked from *one* view, but is applied to *all* views, following the principle stated in §1 of visualization as a window into a massive data set.

Recall that a transform maps each trace in an ensemble to a simpler trace, either containing fewer model states or representing a shorter duration. Also recall that an ensemble is treated as a set of realizations of a stochastic process $X(t)$. Transforms either reduce the range of X or reduce the interval over which t is defined. CHITRA93 uses the classical method of controlling state space explosion: a hierarchical model, in which all but the bottom level contain aggregate model states.

CHITRA93 has three categories of transforms: aggregation, which reduces the range of X by replacing model states by aggregate model states and hence defining a hierarchical model; clipping, which reduces the interval over which t is defined by extracting a portion of a trace; and projection, which reduces the range of X by applying user defined system state transformation rules. Only aggregation is used and hence explained in this paper. The application of the clipping and projection to a single trace file is described in [1]. They are applied to an ensemble simply by applying them independently to all traces in the ensemble.

Aggregation transforms replace each occurrence of certain subsequences of model states in a model state sequence by a single state. There are two ways to aggregate in CHITRA93: by pattern and by filter. The two will be explained by describing how each transform maps ensemble version $\mathcal{E}_v$ to version $\mathcal{E}_{v+1}$. Let m denote the maximum value in $\mathcal{M}(\mathcal{E}_v)$.

Aggregation by pattern: Let μ be a sequence (i.e., a pattern) of model states. The aggregation transform maps μ in version $\mathcal{E}_v$ to version $\mathcal{E}_{v+1}$ as follows. Initially set $\mathcal{E}_{v+1}$ to $\mathcal{E}_v$. For each trace in $\mathcal{E}_{v+1}$, do the following. Let M and O denote the model and occupancy time sequences corresponding to the trace. The pattern μ may occur multiple times in M; repeat the following for each instance x of μ

Event No.	*Timestamp*	*Model State*	*System State*
0	0	0	A,A,T
1	241	1	A,l,T
2	589	11	Aggregate11
3	2976	10	T,T,l

Table 1: Result of replacing model state pattern 2, 3, 4, 5, 6, 7, 8, 9 in Fig. 2.

in M. Let y denote the occupancy time subsequence in O corresponding to M. Replace x by $m+1$ and y by the sum of the elements in y. Note that after the transform, if pattern μ was found then the model state space size grew by one: $\mathcal{M}(\mathcal{E}_{v+1}) = \mathcal{M}(\mathcal{E}_v) \cup \{m+1\}$.

As an example, suppose that Fig. 2 contains the *entire* trace rather than just the first eleven states. If μ is the model state subsequence 2, 3, 4, 5, 6, 7, 8, 9, then after aggregation by pattern the resulting execution sequence is shown in Table 1. Note that CHITRA93 synthesizes a new system state name for a new model state m, namely "Aggregate$m$."

Because a user will not normally visualize all traces in an ensemble, feedback on whether the pattern arises in non-visualized traces is required. Thus CHITRA93 reports the number of times the pattern was found in each trace in the ensemble, and the fraction of each trace's duration that was replaced by the pattern.

Aggregation by filter: The filter transform replaces each contiguous subsequence of states that meets a *filtering criterion* in each model state sequence in an ensemble instance $\mathcal{E}_v$ by a model state whose value exceeds all model states in $\mathcal{M}(\mathcal{E}_v)$. The filter criterion is that either the occupancy time or the number of instances in $\mathcal{E}_v$ of a state is less than a certain value.

Let $f(\mathcal{E}_v)$ denote the set of model states in $\mathcal{E}_v$ that meet the filter criterion. A *run* is a subsequence of a model state sequence in $\mathcal{E}_v$ such that all states in the subsequence are in $f(\mathcal{E}_v)$ and the predecessor and successor states of the subsequence in the model state sequence are not in $f(\mathcal{E}_v)$. The filter transform maps $\mathcal{E}_v$ to $\mathcal{E}_{v+1}$ as follows. Initially set $\mathcal{E}_{v+1}$ to $\mathcal{E}_v$. Let the number of distinct runs in $\mathcal{E}_{v+1}$ be n_r. Replace all instances of runs in $\mathcal{E}_{v+1}$ by model states $\{m+1, m+2, \ldots, m+n_r\}$ such that all runs sharing the same predecessor and successor state are replaced by the same model state. For each subsequence

in an occupancy time sequence in $\mathcal{E}_{v+1}$ that corresponds to a run, replace the subsequence by the sum of its elements.

For example, let $\mathcal{E}_v$ contain two traces with model state sequences of $M_0 = (0,1,2,3,0,1,2,0,1,2)$ and $M_1 = (0,2,4,3,0,2,1,2)$. Consider a filter to remove all states that occur less than 3 times in the whole ensemble; thus $f(\{M_0, M_1\}) = \{3,4\}$. Then M_0 contains one run, (3), and M_1 contains one run, (4, 3). These runs both have 2 as their predecessor and 0 as their successor; therefore they both will be replaced by the same model state. The replacement state, $m+1$, is 5 because $\mathcal{M}(\mathcal{E}_v) = \{0,1,2,3,4\}$. The transform maps M_0 to (0, 1, 2, 5, 0, 1, 2, 0, 1, 2) and M_1 to (0, 2, 5, 0, 2, 1, 2). In the first trace, the occupancy time of 5 is the occupancy time of 3; in the second trace the occupancy time of 5 is the sum of the occupancy times of states 4 and 3.

Example: Analysis of the 16 process database program from §2 starts with opening the ten traces comprising the ensemble. One trace is arbitrarily chosen for visualization with the 2D time view (Fig. 4), namely trace 0.0. There are as many as 1257 model states in trace 0.0, so we will try applying a transform. The trace in Fig. 4 may have one or more patterns of some form; some guesses are indicated by the encircled graph areas in Fig. 4. Thus aggregation by pattern may be appropriate at some point. The other feature that Fig. 4 suggests is that perhaps the longer the trace length, the more of the 6^{16} possible system states randomly arise in the trace. This is suggested by drawing a non-decreasing curve through the top points of the graph; the curve either levels off or continues to increase. Visualizing the longest trace in the ensemble (not illustrated), which is twice as long as trace 0.0, confirms that the number of states continues to increase.

To identify whether there are many random system states, we will apply a transform to aggregate by filter. This transform is selected from the "Transform" menu in Fig. 4, and results in a histogram displaying the number of model states in $\mathcal{S}$ versus occupancy time (not illustrated). The histogram partitions occupancy time into bins, and fills bins with model states according to the occupancy time of each model state instance in $\mathcal{E}_v$. The histogram indicates that almost all states have an occupancy time of less than 1% of the total program execution time. Therefore, we will select a filter threshold of 1% to filter states that occur less than 1% of the time. Fig. 5 shows the result for trace 0.0. The

maximum y axis label has increased to 1267; thus 1267-1256=11 new model states were created for this trace alone. However, the *number* of model states in the ensemble decreased: $||\mathcal{M}(\mathcal{E}_{v+1})|| < ||\mathcal{M}(\mathcal{E}_v)||$ (in our example $||\mathcal{M}(\mathcal{E}_{v+1})|| = 32$ and $||\mathcal{M}(\mathcal{E}_v)|| = 7896$). Most model states in the range [0,1256] no longer occur in trace 0.0; they are aggregated into the eleven new model states. The change in shape of the graph to a dense rectangle means that filter states were introduced throughout the trace from beginning to end.

3.2.3 Consistency of Views under Transformation

Each open view of an ensemble or its member traces displays the latest version. Thus, each time a transform is applied to an ensemble, all open views of the ensemble or its traces are updated to maintain view consistency. For example, after the filter transform was applied to generate Fig. 5, the text view now lists aggregate states produced by the filter transform (similar to Table 1).

3.3 Step-B: Homogeneity Testing

In the last subsection, we applied one filter transform to all traces in ensemble. The next step is to test whether the ensemble is homogeneous. CHITRA93 provides two methods of testing. The homogeneity tests are done now, rather than earlier, because it is easier to apply the tests after the transform application in Step-A reduces the state space size from 7896 to 32.

3.3.1 Method 1: Adjacency Matrix

The adjacency matrix for an ensemble $\mathcal{E}$ is formed as follows. Let $n = ||\mathcal{M}(\mathcal{E})||$, and let $m_0, m_1, \ldots, m_{n-1}$ denote a sorted list of the elements of $\mathcal{M}(\mathcal{E})$. The adjacency matrix is an $n \times n$ matrix. Element i, j is equal to the percentage of the traces that contain at least one transition from state m_i to m_j. Thus, if element i, j is 0, no traces have a transition from model state m_i to m_j; while an entry of 100% indicates that all traces contain at least one instance of the transition. Nearly all non-zero entries in the adjacency matrix of a homogeneous ensemble should be close to 100%.

Example: Fig. 6 contains an adjacency matrix for all ten database traces. (Entries equal to zero and percent signs are not displayed to improve the matrix readability.) The elements are mostly 40% and 60%, indicating that 4 or 6 of 10 traces contain each transition. For example, 40% of the traces contain a transition from model state 13 to 7903. This adjacency matrix suggests that the ensemble is *not* homogeneous due to the prevalence of entries 40% and 60%.

3.3.2 Method 2: Analysis of Ranks

CHITRA93 contains a second homogeneity test that is a classical statistical test of the hypothesis that all samples of the occupancy time of one state in a set of traces are homogeneous: the Kruskall-Wallis rank sum test [7, pp. 422–426], which is an analysis of ranks test. This nonparametric test was selected because it does not require any assumption about the distribution underlying the data.

To use the test, the user selects a single state that occurs in all traces, along with a significance level for the χ^2 test that underlies the KW method. The KW test is applied to a set of samples, each consisting of the set of occupancy times of each occurrence of the selected state in one trace. The null hypothesis tested is that the samples in all traces are drawn from identical distributions.

Example: The adjacency matrix test in §6 suggests that the ensemble of 10 traces in the database example is not homogeneous. To further test homogeneity, the KW test is applied next to the most time consuming system state (L^2A^{12}) in the entire ensemble, in which both gender transactions use locked records for a long period, and the remaining transactions await lock acquisition. After selecting the KW view and then selecting to view all traces in the ensemble in the "Combined" mode, the user is then presented with a dialog box (not shown) in which he selects one state (L^2A^{12}) to test and specifies a significance level (α, the probability of a so-called Type I error (rejecting the hypothesis when actually true), set here to 1%, which means that the hypothesis will be rejected when actually true 1% of the time). The resulting window (Fig. 7) contains two panes. The top pane lists statistics about the occupancy time samples of state (L^2A^{12}) in each of the ten traces. For example, the state occurred 63 times in trace 0.7 and with sample mean occupancy time of 1755 time units. The bottom pane contains the test result. The KW statistic (420.18) exceeds the critical χ^2 value (21.67), and thus the hypothesis is

rejected. Repeating the KW test (not illustrated) with the second most time consuming state (A^2L^{14}) also results in rejection of the null hypothesis.

3.4 Step-C: Ensemble Partitioning and Sub-ensemble Simplification

Three pieces of evidence that the ensemble of ten database traces is not homogeneous have now been collected: the adjacency matrix, the KW test of occupancy time of state (L^2A^{12}), and the KW test of state A^2L^{14}. Therefore the traces comprising the ensemble must be partitioned into mutually exclusive, exhaustive, and homogeneous subsets, or sub-ensembles. The CHITRA93 mechanisms described so far allow three partitioning criteria:

Using adjacency matrix: Partition the ensemble so that all entries in the adjacency matrix for each sub-ensemble is either nearly 0 or nearly 100%.

Using KW test: Partition the ensemble so that the occupancy time samples of one or more system states from traces within a sub-ensemble pass the KW test.

Using pattern aggregation: Partition the ensemble so that all traces within a sub-ensemble contain the same pattern.

Example: The third criteria will be effective for our example. To apply it, visualize any single trace from the ensemble. Trace 0.0 will again be used (Fig. 5). To identify whether a pattern is present, first zoom the 2D time view of Fig. 5 in the x-direction so that the window displays 4% of the time axis. Fig. 8 contains a pattern. The user selects the pattern by using the mouse to draw a bounding box over a portion of the graph (see Fig. 8); all states corresponding to crosses by the box form the pattern. CHITRA93 then searches all trace files in ensemble 0 for instances of the pattern. All instances visible in open views of the ensemble are highlighted in yellow (This appears as the white portion of the curve in Fig. 8, starting just before time 106527 and extending to the right edge of the window.). The portions of the curve that do not match the pattern are not highlighted (This appears as a grey curve extending from the left window edge to just before time 106527.). The portion that does not match in Fig. 8 fails to match because it contains the encircled portion labeled "perturbation."

The user now has the option of applying the pattern or deselecting the pattern (and then applying an alternate transform or selecting another pattern). Applying the pattern replaces each instance of the pattern in each trace by a single, new, aggregate model state. Fig. 8 now changes to Fig. 9. The new aggregate state is model state 1268 (which is one more than 1267, the largest model state for trace 0.0 in Fig. 8, consistent with the definition of aggregation (§3.2.2)). Unzooming Fig. 8 in the x-axis direction so that 100% of the trace duration is visible in the window shows visually the trace portion that matches the pattern. Fig. 10 shows the unzoomed trace. All but five narrow intervals of time match the pattern – a dramatic contrast to the original trace in Fig. 4.

The pattern was effective in simplifying trace 0.0. To what degree were the other nine traces in the ensemble simplified? Applying the transform not only updates the 2D time view as shown in Figs. 9 and 10, but also reports, for each trace, the fraction of states in each trace that match the pattern. The pattern matches over 90% of the duration of traces 0.0, 0.4, 0.7, and 0.8. However it does *not* match the remaining six traces. This suggests an ensemble partition: $\mathcal{E}' = \{0.0, 0.4, 0.7, 0.8\}$ and the complement, $\overline{\mathcal{E}'}$. Next we arbitrarily choose one trace, say 0.5, from the non-matching sub-ensemble ($\overline{\mathcal{E}}$). We open a 2D view of 0.5 (not illustrated), zoom, see that there is a pattern, and select and replace the pattern. Chitra93 reports that all traces in $\overline{\mathcal{E}'}$ match the pattern. Therefore $\overline{\mathcal{E}}$ need not be further partitioned. Note that the partition sizes (4 and 6 of 10) are consistent with the 40% and 60% entries in the adjacency matrix (Fig. 6).

If $\mathcal{E}'$ and $\overline{\mathcal{E}'}$ are really two homogeneous sub-ensembles, then the homogeneity tests from §3.3 applied separately to each sub-ensemble should yield more positive results than those in §3.3. This is indeed the case. The adjacency matrix for each sub-ensemble contains elements that are almost all 100%. The KW test, when applied separately to $\mathcal{E}'$ and $\overline{\mathcal{E}'}$ yields acceptance on the same two states tested in §3.3. Thus the evidence supports the conclusion that the partition yields two homogeneous sub-ensembles, and we are ready to model each sub-ensemble.

3.5 Step-D: Model Generation

Let $\mathcal{E}_v$ denote the version resulting after Step-C of a set of homogeneous traces forming a sub-ensemble. CHITRA93 views such traces as a set of realizations of a stochastic process $X_{\mathcal{E}_v}(t)$. CHITRA93 uses $\mathcal{E}_v$

to infer $X_{\mathcal{E}_v}(t)$.

3.5.1 Semi-Markov chain model

The normal representation of a stochastic process $X_{\mathcal{E}_v}(t)$ in CHITRA93 is as a time invariant, continuous time semi-Markov chain [3, pp. 290–294]. The state space of the chain is the set of model states, $\mathcal{M}(\mathcal{E}_v)$, that occur in ensemble version $\mathcal{E}_v$. The definition of a semi-Markov chain requires enumeration of the occupancy time distribution for each state in $\mathcal{M}(\mathcal{E}_v)$ and enumeration of a transition rate matrix. Instead of these, CHITRA93 displays in model view statistics of occupancy time and a one step probability transition matrix, P. The transition rate matrix can be calculated from the occupancy time statistics and P. The semi-Markov view displays P rather than the rate matrix because we feel that a user gets more intuition about the sub-ensemble behavior from the former than from the later. Matrix P contains one row and one column for each element of $\mathcal{M}$, similar to the adjacency matrix (§3.3). Element i, j of P is the percentage of transitions in $\mathcal{E}_v$ matching (m_i, m_j).

Example: Fig. 11 illustrates the semi-Markov view of the sub-ensemble consisting of database traces $\{0.0, 0.4, 0.7, 0.8\}$ after the filter and pattern aggregation transforms are applied. The banner contains "0.[0,4,7,8]" to identify which partition the view represents. The top pane contains P, which is an irreducible chain. The bottom pane lists, for each state m in $\mathcal{M}(\mathcal{E}_v)$, statistics of the occupancy times of all instances of m in $\mathcal{E}_v$. The scroll bars indicate that the illustration omits a few rows and columns from each pane. Fig. 12 contains two transition graphs. The thick and thin black lines of the left one corresponds to P; the other corresponds to the remaining sub-ensemble ($\{0.1, 0.2, 0.3, 0.5, 0.6, 0.9\}$).

We started with an ensemble containing nearly 8,000 states. Through two aggregation transforms (by filter and by pattern), the ensemble has been reduced to two chains with 20 states each. In addition, the model reveals that the execution of the database converges to one of two cycles, represented by the two graphs in Fig. 12.

3.5.2 CHAID-based model

The semi-Markov model can be improved. The semi-Markov model defines the probability distribution of the next model state conditioned only on the current model state. Although the semi-Markov model is fast to compute – the time complexity is proportional to the sum of the lengths of current trace versions used to build the model – there is no inherent reason why a system cannot violate this assumption. Therefore CHITRA93 incorporates one alternative (but more computationally costly) model: CHAID-based.

Given a data set that contains vectors in which one component represents the discrete values of a dependent variable and the remainder represent the discrete values of a set of independent variables, the CHAID (Chi-squared automatic interaction detection) [6] method partitions the independent variable values into mutually exclusive and exhaustive subsets. Let d (respectively, c) denote the number of values of the dependent (respectively, independent) variable. The partitioning procedure first constructs c contingency tables [7, pp. 249–253], each of dimension $c \times d$, one for each vector component that represents an independent variable. In a contingency table, each row (column) represents one dependent (independent) variable value. Let $d(i)$ (respectively, $c(j)$) denote the value corresponding to row i (column j) of the contingency table (for $0 \leq i < c$ and $0 \leq j < d$). Table element i, j represents the fraction of the data set in which the value of the vector components to which the table corresponds has $d(i)$ as the dependent variable value and $c(i)$ as the independent variable value. A procedure is then applied to reduce the number of rows by combining (according to a constraint) values so as to maximize the significance level of the table. The procedure calculates, for all possible value combinations, a χ^2 statistic; the table with the highest statistic is the most significant. The reduced table with maximum χ^2 statistic for *all* independent variables are compared; the one with the highest χ^2 value determines the final answer of which independent variable best predicts the dependent variable.

CHITRA93 implements the CHAID method, but employs a slightly different significance test based on the Bonferroni inequality recommended by Biggs, DeVille, and Suen [4]. The modified test keeps the type 1 error rate in calculating the significance of contingency table(s) yielding the best partition below the user-specified significance level. (A type 1 error is to call a partition significant when in reality

no relationship exists between the dependent and independent variables.) Hence we call the model "CHAID-based."

A state s is called *branching* if there are transitions out of s into two or more states in the ensemble. The CHAID-based model is used to analyze branching states in traces, corresponding to entries in matrix P that are not 0 or 100%. For example, in Fig. 11, 50% of the transitions out of branching state 13 are into state 7899; the rest are into state 7903. The CHAID-based model analyzes as its vectors all subsequences in each SSS $S_{k-W_l}, S_{k-W_l+1}, \ldots, S_{k-1}, S_k, S_{k+1}, \ldots, S_{k+W_u-1}, S_{k+W_u}$ in $\mathcal{E}_v$, where S_k is a branch outcome (i.e., 7899 or 7903), to identify which state S_{k-1} ($l \neq 0$) best predicts S_k. If $l = -1$ is the best predictor then the semi-Markov model is the best representation. Otherwise the CHAID-based model yields, in column j of its transition matrix, the probability distribution of transitions into each state analyzed conditioned on the state l positions away.

The non-negative integer values W_l and W_u define the lower and upper edges of a *window* of states that are to be analyzed. If $W_u = 0$ (respectively, $W_l = 0$) then only states in the past (future) with respect to the state in position k are analyzed. The computational cost of using CHAID grows linearly with the window size $W = W_u - W_l + 1$. Thus the CHITRA93 user must choose the window size by balancing the time he will allow for analysis with his estimate of the distance (in number of positions) between the state in position k and the best predictor state. A logical choice for W is a positive integral multiple of the period in number of transitions of the mean number of transitions required for the chain (matrix P) to return to state s_k.

Example: The CHAID-based model can be used to analyze whether the transition out of state 13 is truly random in sub-ensemble $\{0.0, 0.4, 0.7, 0.8\}$. Opening CHITRA93's CHAID-based model view yields a screen similar to Fig. 13. Unlike Fig. 11, the larger window is actually blank when the view is first selected, and is filled after the user optionally changes parameters and clicks the "Apply" button in the "Model Parameters" dialog box. The first pair of radio buttons ("Merge method") will not be discussed here. The next radio buttons specify whether the analysis is independent or joint. In independent analysis, the CHAID procedure is applied independently to each state that the user selects (via the "List of states to analyze" buttons, described later), or once for all user selected states. In the former case,

for each state m selected for analysis, the contingency tables always have $d-2$ columns corresponding to states m and $\overline{m}$. In the later case, if n states are selected for analysis, then $d = n$, and each column corresponds to one of the selected states. The window size is selected by the three radio buttons under "dependent position." One can analyze just the past ($W_u = 0$), just the future ($W_l = 0$), or both ($W_l \neq 0$, $W_u \neq 0$). Selecting any of the three radio buttons displays a second dialog box (not shown) in which the non-zero values of W_l and W_u are entered. The significance level has the same meaning as it did in the KW rank sum test. Finally, the "List of states to analyze" radio buttons control how the data passed to CHAID is formed. Let $\mathcal{S}$ denote a set of model states. The "All" button sets $\mathcal{S}$ to all model states. Clicking on the "Specify" button pops up a second box (not illustrated) containing a list of all model states. The user must select at least two model states, forming set $\mathcal{S}$. The result is to pass only those model state subsequences of length W in all traces in $\mathcal{E}_v$ satisfying $s_k \in \mathcal{S}$ to CHAID (which is all subsequences in the case of "All.") In this case we select "specify" and select the branch outcomes: $\mathcal{S} = \{7899, 7903\}$. Hitting the "Apply" button starts the CHAID calculation and yields the content shown in the other window in Fig. 13.

The CHAID-based model is displayed in three window panes; only the top is relevant here. The top pane contains a matrix with one column per state in $\mathcal{S}$. The numbers in square brackets ("[-2]" for both states 7899 and 7903 in Fig. 13) list the best predictor position, l. *Therefore the best predictor for the branch outcomes is the state two steps back, thereby violating the Markov assumption.* The rows correspond to each model state found in any independent position. The percentages in the matrix list the number of times that the predecessor of the model state labeling the column was the model state labeling the row. In this case 100% (respectively, 99%) of the time state 7899 (7903) is preceded two steps back by state 2071 (7903). Applying the CHAID-based model to the other sub-ensemble yields a similar result. Combining the semi-Markov and the CHAID models replaces the thin black lines in Fig. 12 by the medium grey lines, yielding a *branchless* transition graph that is an irreducible and periodic chain.

3.6 Observations on the Sample Session

Using CHITRA93 is similar to using a statistical package to fit distributions to samples or exploring a data set with a scientific visualization system in the sense that different users may not reach identical

conclusions when given the same data to analyze. A desirable characteristic for a tool is to avoid being too sensitive to the specific sequence of steps that a user follows. How sensitive is the model generated by CHITRA93 to the steps that one follows in using the tool? For example, if an inexperienced user deviated from the steps followed in §3.2 to §3.5, would the user still discover that the ten database traces can be partitioned into two subsets, each following a different transition graph (Fig. 12)?

To answer this, consider what would have happened if we made an invalid assumption early in the analysis – that the ten trace ensemble was homogeneous – and then skipped part of the analysis – testing homogeneity (Step-C). This would appear to be a catastrophic error, because partitioning the ensemble in §3.4 lead to two models, whereas assuming homogeneity would lead to one model in §3.5.

Despite the error, we would obtain the *same* model. The model building in Step-D would be more time consuming, but the CHAID-based model would show the proper ensemble partitioning. After the filter and two pattern aggregations in §3.2 and §3.4, in Step-D the semi-Markov model would be an irreducible, aperiodic chain. In the chain, there are four possible successors to model state 13 (common to both thick black/medium grey line graphs in Fig. 12): 7899, 7903, 7912, and 7909. Subsequent application of the CHAID-based model to the four successors shows that each of these four states is best predicted by the state two steps back, and that nearly 100% of the time there is only a single state two steps back. The resulting stochastic process is decomposable into two irreducible, periodic chains – the same two appearing in Fig. 12. Therefore CHITRA93 yields the same model even with an errant user decision, but requires more computation (the CHAID-based model was run on more states than §3.5).

4 Conclusions

CHITRA93 takes as input a set (or ensemble) of time-series data files and generates empirical models fitting the ensemble. The paper exemplifies CHITRA93 as a tool for performance analysis of parallel programs, because this was our motivation for building the tool, although CHITRA93 can be used with any categorical time series data. This generality is critical for its use in software, because CHITRA93 makes no assumptions about the type of applications, programming language, or computer architecture of the traces that it analyzes.

CHITRA93's contribution is in analyzing an ensemble, rather than a single trace file, and in building a model of the ensemble, rather than just displaying a visualization or statistics. Visualization is used only as a means of examining portions of select traces to assist construction of an empirical model. The approach potentially scales to analyze data from massively parallel systems.

CHITRA93 should be viewed as a complement to visualization systems, rather than as a replacement. CHITRA93 incorporates (1) a visualization facility, (2) a set of transforms that simplify the traces and in turn reduce the state space size of the model eventually generated, (3) a user interface that allows some transforms to be selected by editing the visualization with a mouse, (4) a set of homogeneity tests that allow an ensemble to be partitioned, (5) a semi-Markov model whose computation time is linear in the sum of the lengths of all traces analyzed, (6) and a CHAID-based model that can identify non-Markovian relationships among transitions in the traces.

The fundamental premise that one should analyze ensembles rather than individual traces is borne out by the 16 processor parallel database example analyzed here. In four of ten executions, the program converged to one sequence, and the remaining six executions converged to another. Each sequence contains an interleaving of random and deterministic behavior, which were exposed by two transforms: state aggregation by filter and by pattern. The effectiveness of transforms in reducing the original 7896 model states to just two chains with 20 states each is apparent from comparing Figs. 4 and 10.

Unlike visualization systems, CHITRA93 requires some sophistication on the part of the user to interpret statistical tests and judge which transforms to apply, how to partition the ensemble, and which states to analyze. Therefore a user must make an investment in learning how to use CHITRA93. On the other hand, the degree of sophistication is similar to that required to use certain other performance analysis tools, such as simulations in analyzing output or queuing network based tools.

CHITRA93 does offer a degree of robustness in allowing users to use bad judgement during analysis but still arrive at good models. This is demonstrated in §3.6, where we intentionally made an incorrect assumption (that the ensemble was homogeneous) and repeated the analysis, but arrived at the same two models as when we correctly tested and rejected homogeneity before building the models. A bad decision just increased the computation time to obtain the models.

Several areas of future research are development of automated partitioning algorithms; exploration of homogeneity tests that identify which traces cause rejection of a homogeneity hypothesis; parallel and distributed execution of CHITRA93 to allow the tool to scale to arbitrarily sized ensembles; and development of *parametric* models, in which an ensemble contains traces representing system execution with a variety of parameter values (e.g., matrix sizes, number of processes), and the model contains formulas in terms of the parameter rather than constant values.

Acknowledgements

Anup Mathur, Mike Tiffany, Alan Batongbacal, and David Holcomb wrote portions of CHITRA93. Randy Ribler and Alan Batongbacal made helpful suggestions on the manuscript. The database traces were collected at the Argonne National Laboratory.

References

[1] 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, Nov. 1992.

[2] B. Bershad, E. Lazowska, H. Levy, and D. Wagner. Presto: A system for object-oriented parallel programming. *Software – Practice and Experience*, 18(8):713–732, Aug. 1988.

[3] U. N. Bhat. *Elements of Applied Stochastic Processes.* John Wiley, New York, 2nd edition, 1984.

[4] D. Biggs, B. de Ville, and E. Suen. A method of choosing multiway partitions for classification and decision trees. *J. of App. Stat.*, 18(1):49–62, Jan. 1991.

[5] G. Haring and G. Kotsis, editors. *Performance Measurement and Visualization of Parallel Systems*, volume 7 of *Advances in Parallel Computing.* North-Holland, Moravany, Czechoslovakia, 1993. Proc. of the Workshop on Performance Measurement and Visualization, Oct. 1992.

[6] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.

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

[8] D. A. Reed, R. A. Aydt, T. M. Madhyastha, R. J. Noe, K. A. Shields, and B. W. Schwartz. The Pablo performance analysis environment. Dept. of Comp. Sci., Univ. of IL, 1992.

[9] M. Simmons and R. Koskela, editors. *Performance Instrumentation and Visualization.* ACM Press, New York, 1990. Based on the Workshop on Parallel Computer Systems: Instrumentation and Visualization, May 1989.

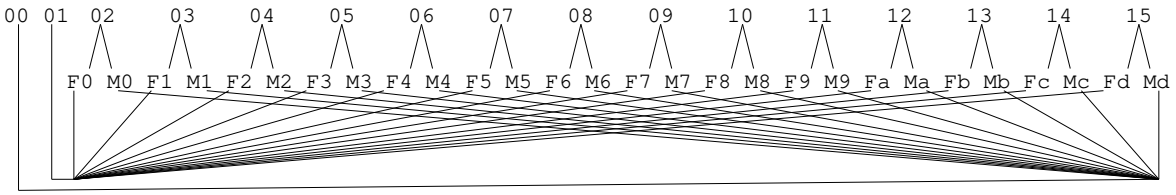

Figure 1: Graph of the database problem denoting which transaction types lock which records. The 28 records are denoted by two digits: the first digit is M or F, and denotes male or female, and the second hexadecimal digit denotes the region (0 to d). The records F0 through Md are numbered 0 to 27.

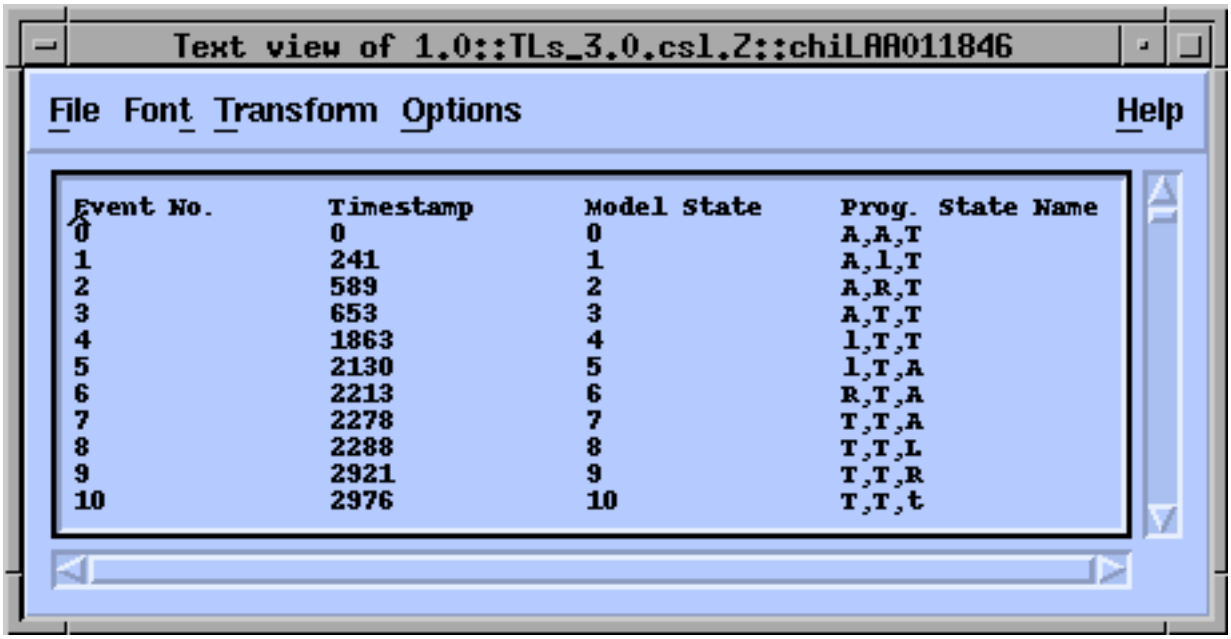

Figure 2: Text view of one database trace (with 3 processes)

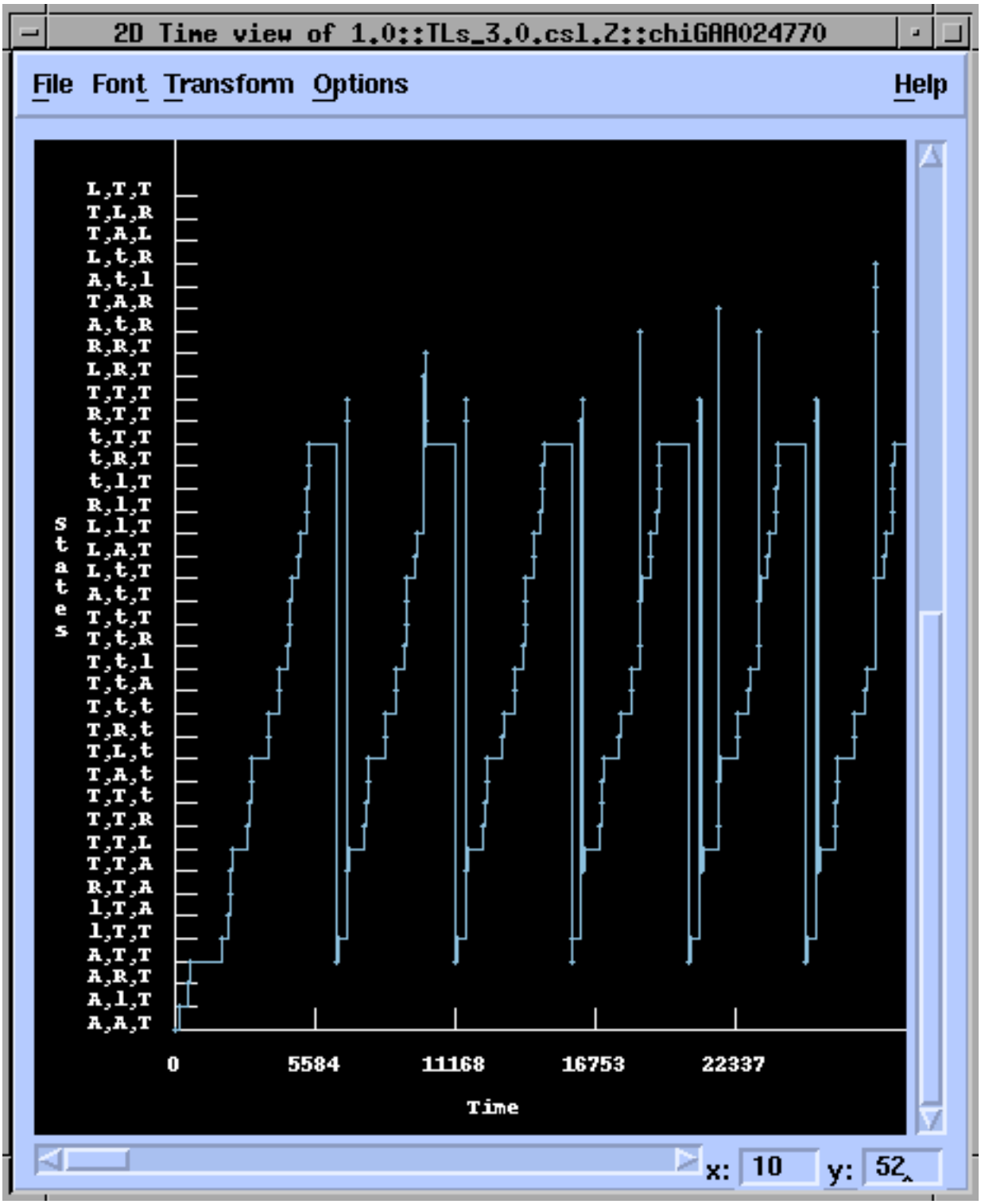

Figure 3: 2D view of one database trace (with 3 processes)

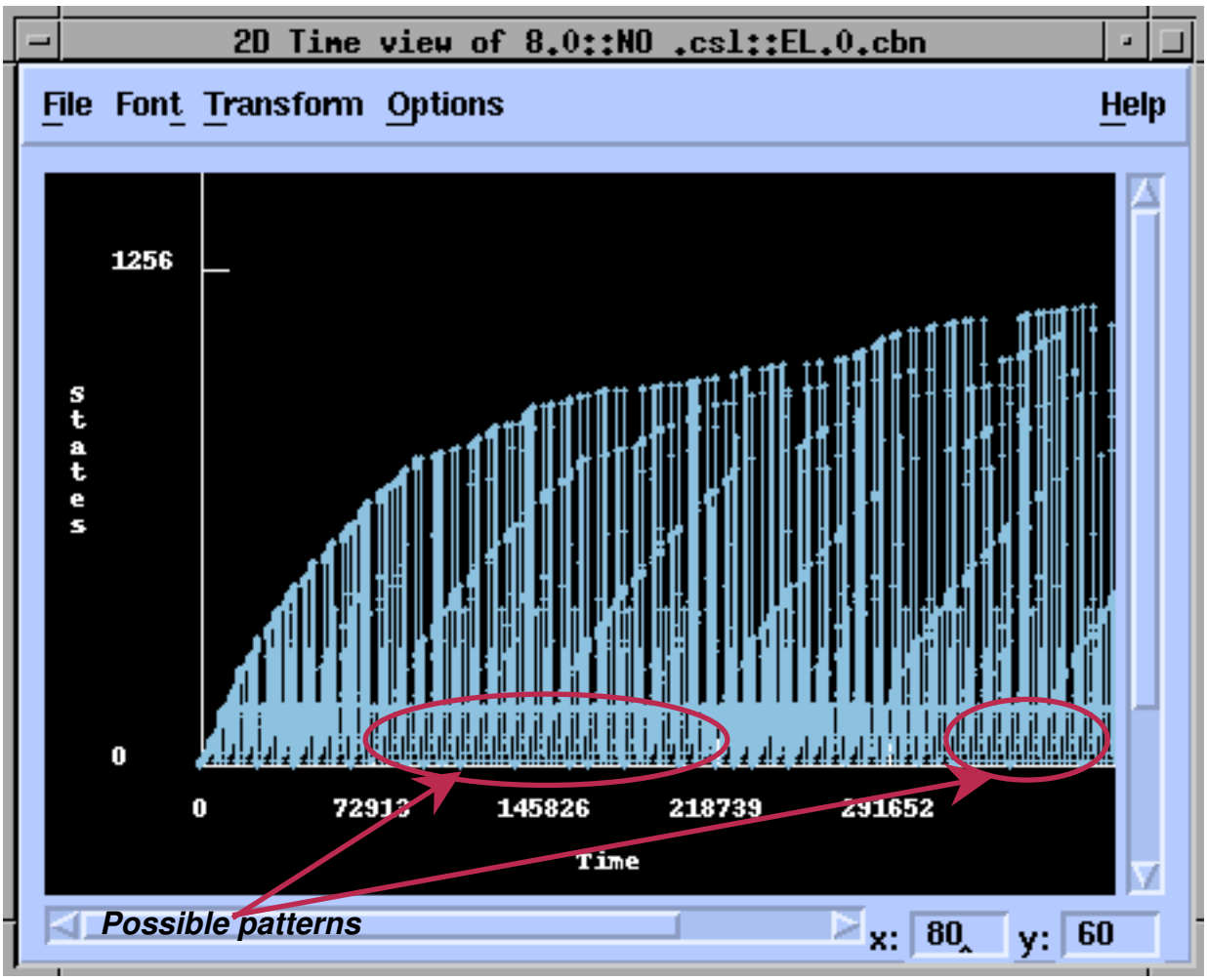

Figure 4: 2D view of one database trace (with 16 processes)

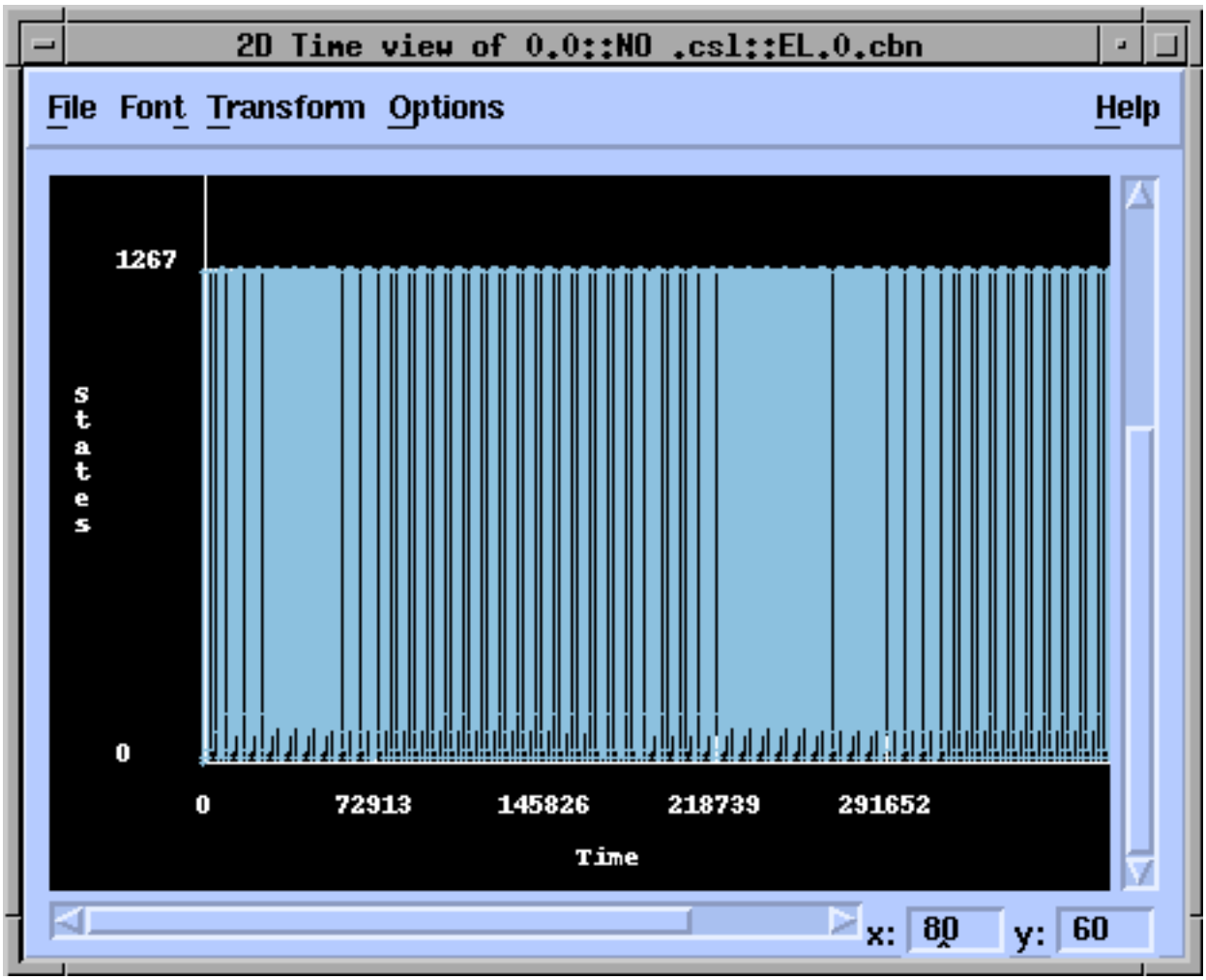

Figure 5: Result of applying 1% time filter to Fig. 4

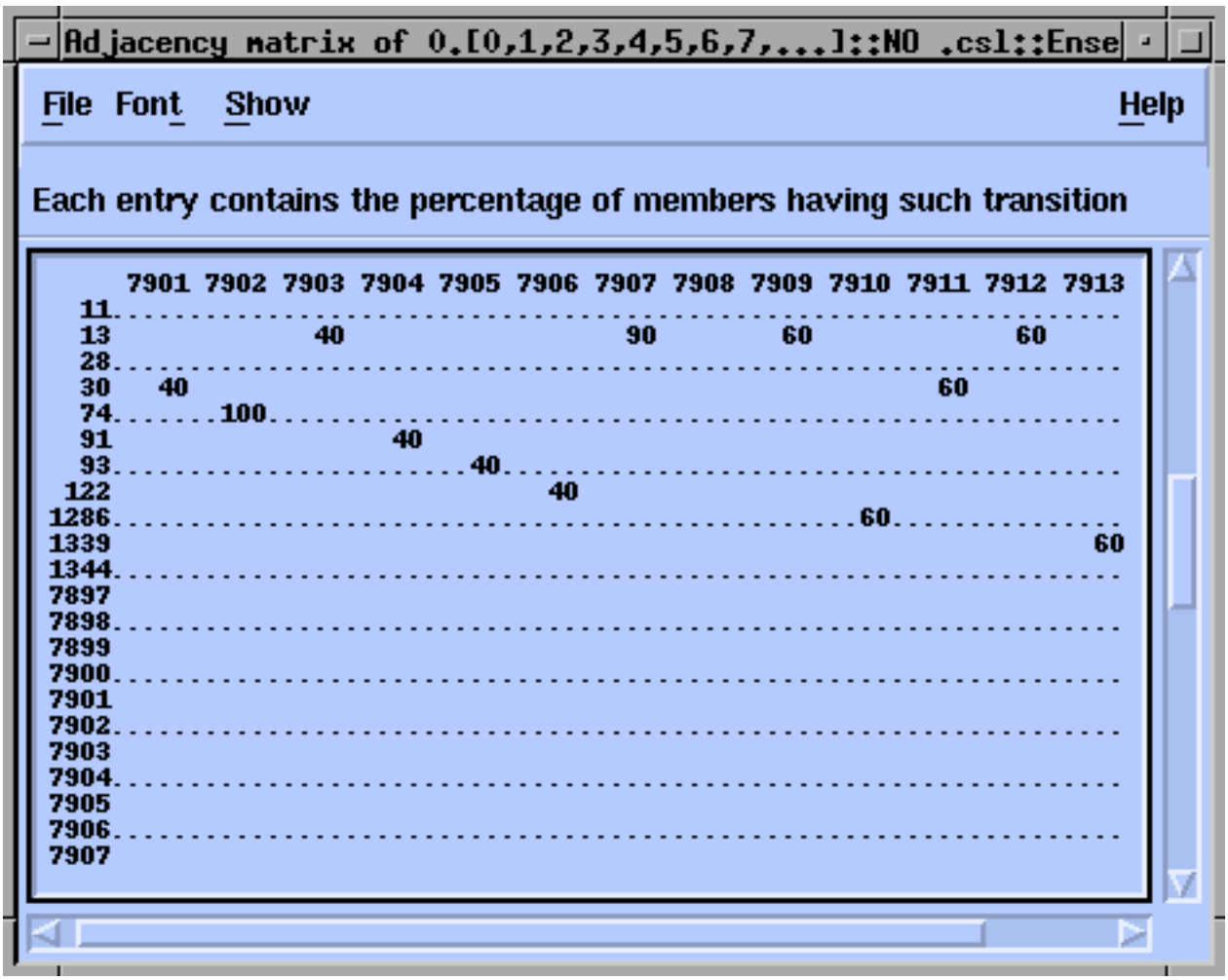

Figure 6: AM homogeneity test applied to entire ensemble

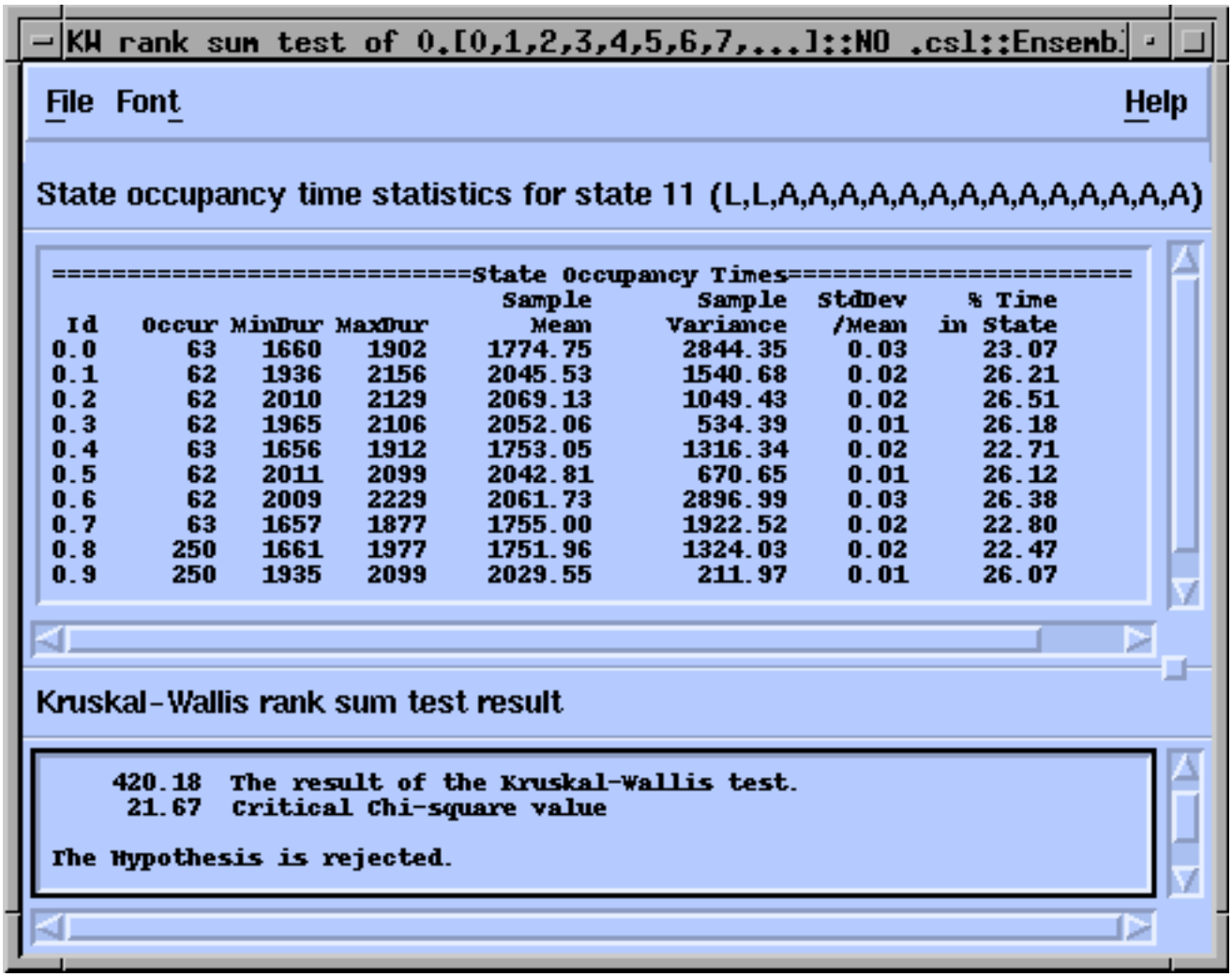

Figure 7: KW rank sum homogeneity test applied to the most time consuming state in the ensemble

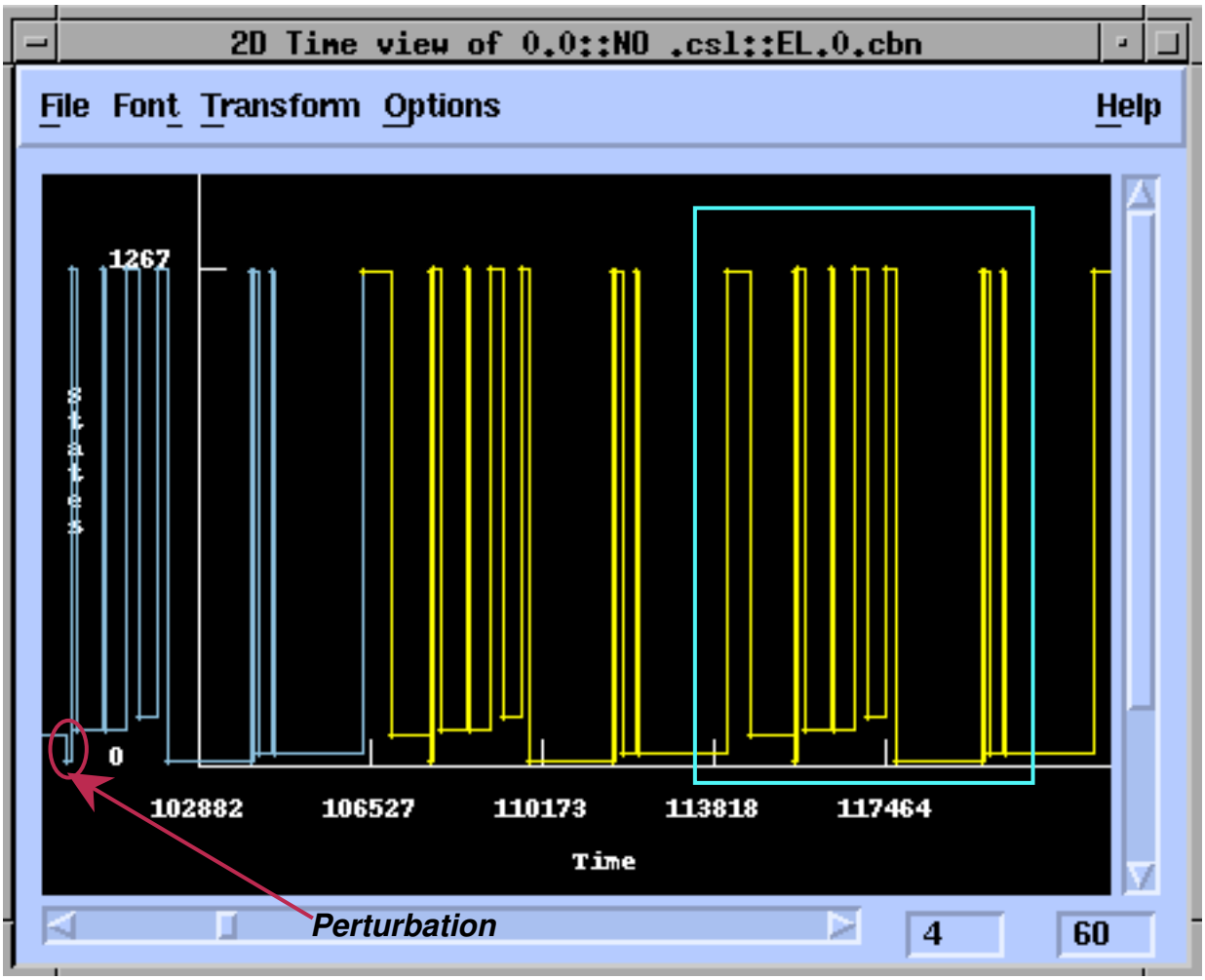

Figure 8: Result of zooming Fig. 5 in x direction to 4%. Box denotes selected pattern. Initial portion of graph, in grey, does not match pattern. Remainder of curve, in white, does match pattern.

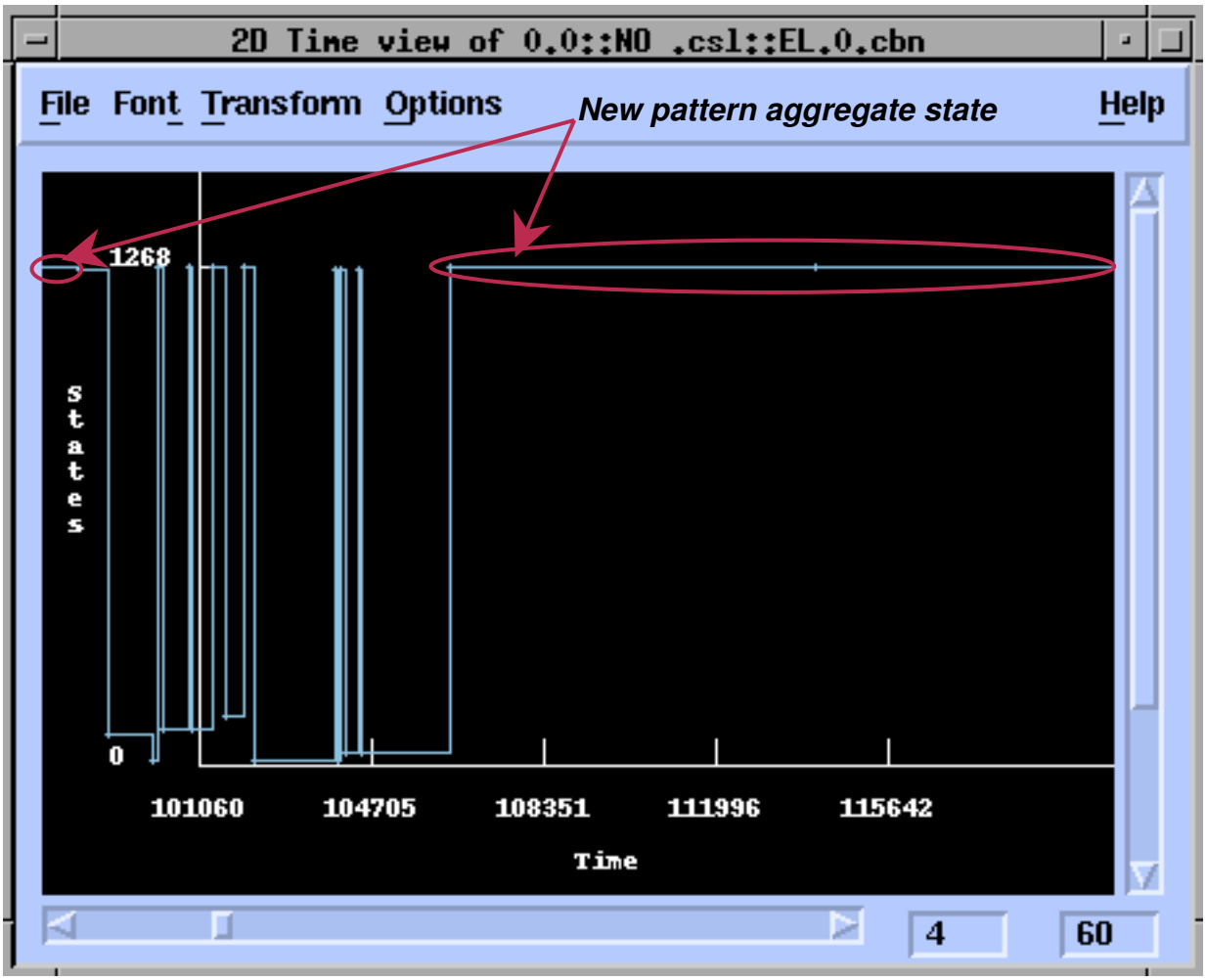

Figure 9: Result of aggregation by pattern in Fig. 8. Note new pattern aggregate state.

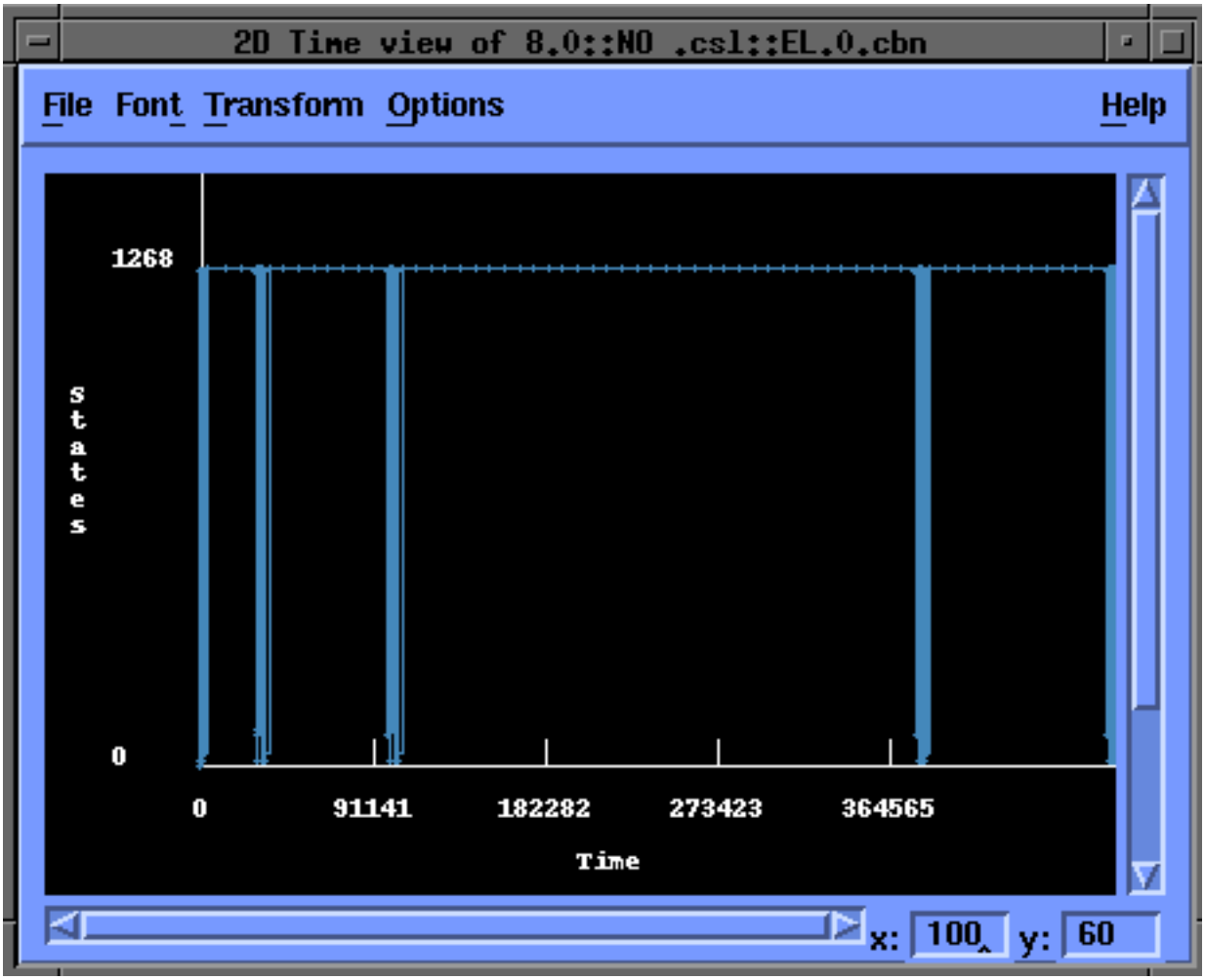

Figure 10: Result of unzooming Fig. 9 to 100%; compare to Fig. 4

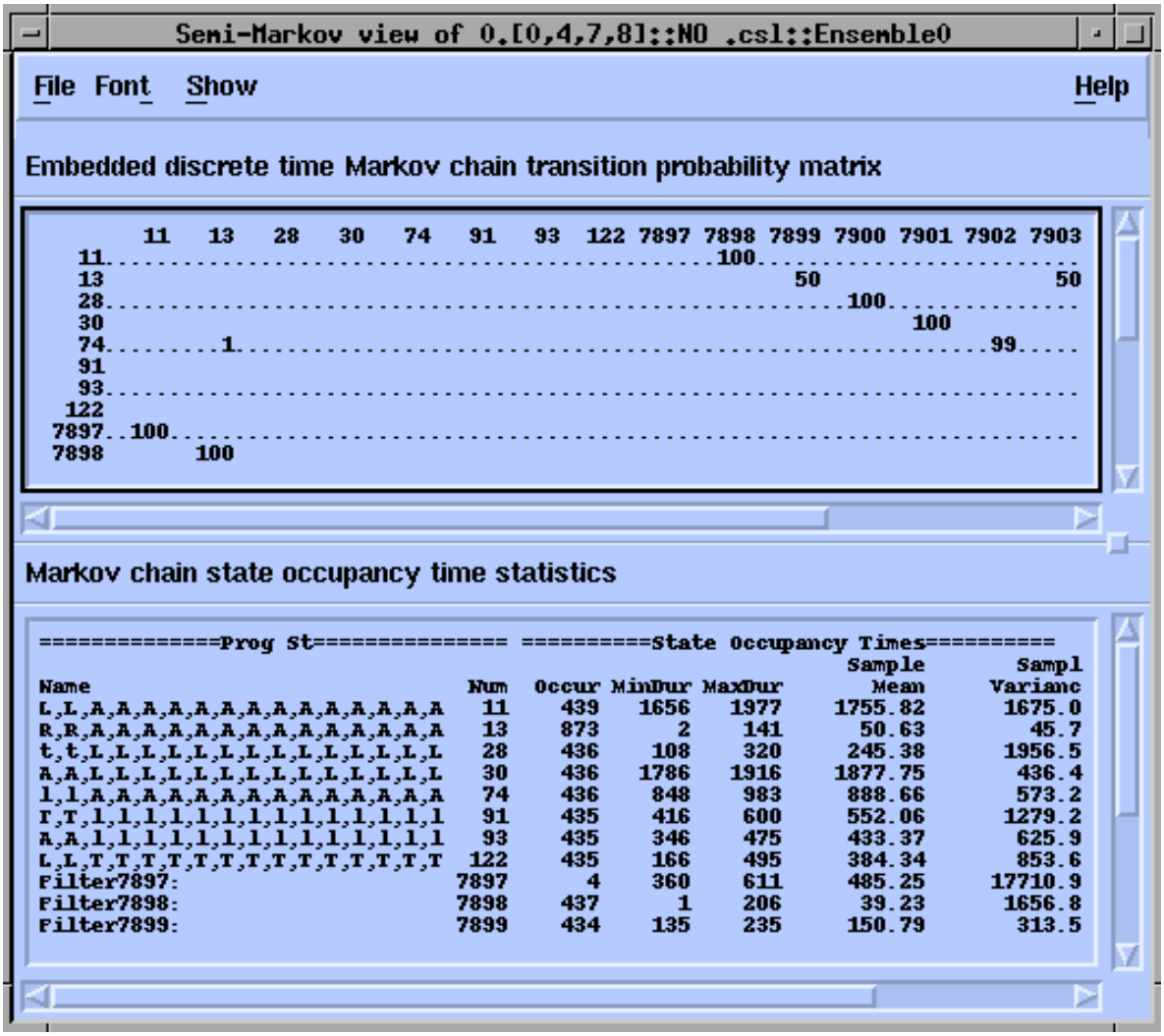

Figure 11: Semi-Markov Model view for one sub-ensemble

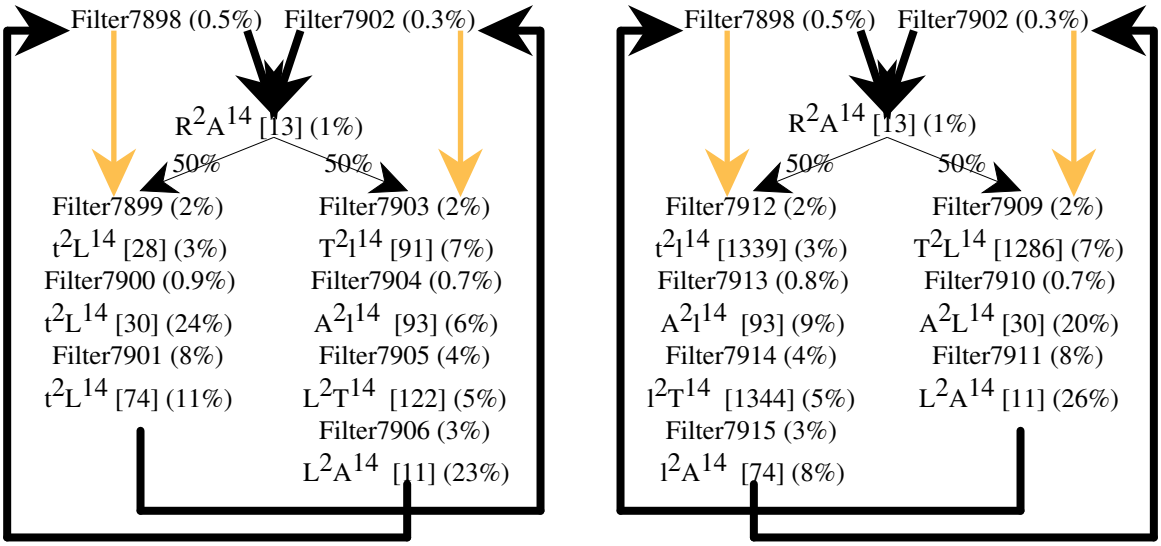

Figure 12: Final models. Thick and thin black lines represent semi-Markov model; CHAID-based model adds medium grey lines. Transitions probabilities not exceeding 1% are omitted. Left graph is partition $\{0.0, 0.4, 0.7, 0.8\}$; right is other partition.

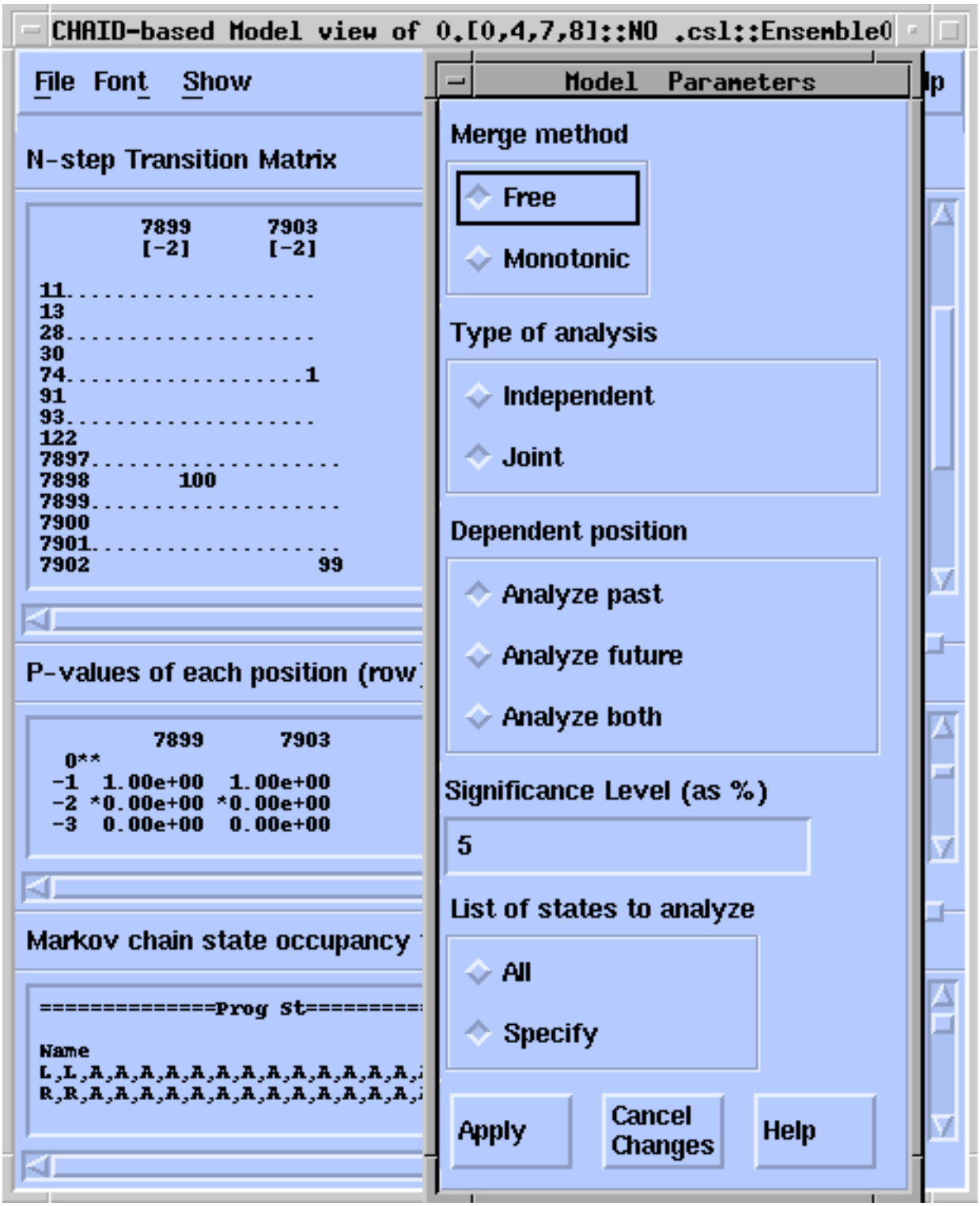

Figure 13: CHAID-based model

Note on printing this document:

All figures illustrating screen dumps are in color. Therefore you should print the final pages of this document on a color postcript level-2 printer, if available.