University
of Southern
California

Applying the Metapattern Mechanism to Time Sequence Analysis

Wei-Min Shen
Bing Leng

USC/Information Sciences Institute

Arun Chatterjee

Motorola, Inc.

June 1995

ISI/RR-95-398

INFORMATION
SCIENCES
INSTITUTE

310/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Applying the Metapattern Mechanism to Time Sequence Analysis

Wei-Min Shen
Bing Leng

USC/Information Sciences Institute

Arun Chatterjee

Motorola, Inc.

June 1995

ISI/RR-95-398

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1995	Research Report

4. TITLE AND SUBTITLE

Applying the Metapattern Mechanism to Time Sequence Analysis

5. FUNDING NUMBERS

(none)
Contract called
MOTOROLA-SHEN

6. AUTHOR(S)

Wei-Min Shen, Bing Leng, and Arun Chatterjee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

8. PERFORMING ORGANIZATON REPORT NUMBER

ISI/RR-95-398

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

Motorola, Inc.
5918 West Courtyard Drive
Austin, TX 78730

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT

UNCLASSIFIED/UNLIMITED

12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper reports an application of the metapattern mechanism to analyze time sequence data for semiconductor process control and fault-detection. The challenge of this task includes the high-dimensionality and large quantity of data, several types of uncertainties in measurements, and lack of effective machine learning tools for time sequence analysis. Using metapatterns, we effectively find a nice angle to view the sequences and compare positive with negative sequences. Such views and comparisons helped us to develop a new technology called time-sequence fault-detector. The main idea behind the detector is to learn a typical positive sequence for each control parameter from the classified sequences, along with thresholds for allowed deviations for values, durations, and time shifts. The typical sequence and the thresholds are then used on-line as a basis to detect faults in new and unclassified sequences incrementally. At present, this detector is realized as a new action for metapatterns (composed from previous metapattern actions), and performs well on the example sequences that are currently available to us. Ultimately, this detector will perform on-line on Motorola's semiconductor plasma machines to detect deficiencies of operations on wafers in real-time.

14. SUBJECT TERMS

Classification, clustering, discrimination analysis, metapattern, time sequence data

15. NUMBER OF PAGES

13

16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Applying the Metapattern Mechanism to Time Sequence Analysis

Wei-Min Shen & Bing Leng
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{shen,leng}@isi.edu

Arun Chatterjee
Motorola Inc.
5918 West Courtyard Drive
Austin, Texas 78730
arun_chatterjee@email.mot.com

Abstract

This paper reports an application of the metapattern mechanism (Shen, Ong, Mitbander and Zaniolo 1994) to analyzing time sequence data for semiconductor process control and fault-detection. The challenge of this task includes the high-dimensionality and large quantity of data, several types of uncertainties in measurements, and lack of effective machine learning tools for time sequence analysis. Using metapatterns, we effectively find a nice angle to view the sequences and compare positive with negative sequences. Such views and comparison helped us to develop a new technology called *time-sequence fault-detector*. The main idea behind the detector is to learn a *typical positive sequence* for each control parameter from the classified sequences, along with *thresholds* for allowed deviations for values, durations, and time shifts. The typical sequence and the thresholds are then used on-line as a basis to detect faults in new and unclassified sequences incrementally. At the present, this detector is realized as a new action for metapatterns (composed from previous metapattern actions), and performs well on the example sequences that are currently available to us. Ultimately, this detector will perform on-line on Motorola's semiconductor plasma machines to detect deficiencies of operations on wafers in real-time.

Key Words: metapattern, time sequence data, clustering, classification, and discrimination analysis.

1 Introduction

Time sequence data pose some unique challenges for today's knowledge discovery tools. Because of their high dimensionality and large quantity, time sequence data require good facilities to identify critical attributes and focus attention to the important aspects of the data. Due to the multiple sources of uncertainties, such as variations on values, durations, and time shifts, time sequence data demand more sophisticated techniques to deal with noises. Furthermore, since there are temporal dependencies between values of a parameter, time sequence data often cause many off-shelve machine learning algorithms to break down or perform at a sub-optimal level.

In this paper, we report our progress of applying the metapattern mechanism (Shen, Ong, Mitbander and Zaniolo 1994) to the analysis of time sequence data. As described in earlier papers, a metapattern is in essence a second-order template that specifies the type of patterns to be discovered. For example, let P, Q and R be variables for predicates, then a metapattern

$$P(X,Y) \wedge Q(Y,Z) \Rightarrow R(X,Z)$$

specifies that the patterns to be discovered are transitivity relations $p(X,Y) \wedge q(Y,Z) \Rightarrow r(X,Z)$, where p, q, and r are specific predicates. One possible result of this metapattern is the pattern

$$\textit{citizen}(X,Y) \wedge \textit{officialLanguage}(Y,Z) \Rightarrow \textit{speaks}(X,Z)$$
with a probability (say 0.93),

where *citizen, officialLanguage*, and *speaks* are relations that bind to P, Q, and R, respectively, in the current database.

In general, a metapattern can be viewed as a two-part specification: the left-hand side specifies a constraint on how data should be prepared, and the right-hand side specifies an action to be applied on the prepared data. For example, the left-hand of the above example, when P and Q are bound to specific predicates p and q respectively, is a constraint on how to fetch those data pairs (X,Z) that satisfy $p(X,Y) \wedge q(Y,Z)$. The right-hand of that example, when R is bound to a specific predicate r, is interpreted as an action that computes the ratio of (X,Z) pairs, among those returned by the left-hand side, that satisfy $r(X,Z)$. (In fact, the precise syntax for the right-hand side of the above metapattern is *TruthRatioComputer*$(R(X,Z))$, where *TruthRatioComputer* is an action that performs the described procedure.)

At the present, the mechanism supports the following four basic metapattern actions:

- *TruthRatioComputer*$(r,[X,Z,...])$, where r is a predicate and $[X,Z,...]$ is a tuple of variables for a given set of data. This action returns an expression $r(X,Z,...)$ along with a number (such as the probability 0.93 in the above example) indicating the ratio of the given data tuples that satisfy the predicate $r(X,Z,...)$.

- *Ploter*$([X,Z,...])$. This action simply plots the given set of data.

- *Classifier*$([X,Z,...])$. This action uses the CDL incremental learning algorithm (Shen 1994) to return a set of class descriptions that best classify the given data.

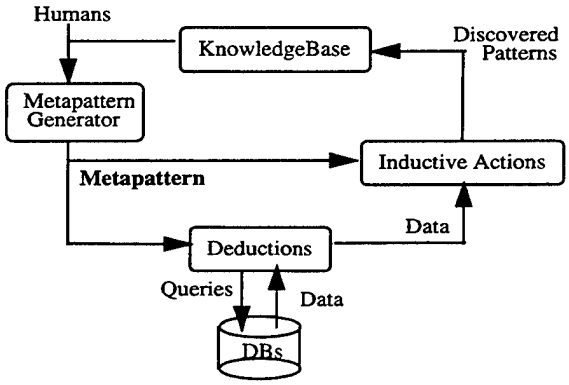

Figure 1: Metapattern and the Discovery Loop

- *Cluster*($[X, Z, ...]$). This action uses a Bayesian-based Cobweb clustering algorithm (Fisher 1987; Shen 1994) to return a set of cluster descriptions that best cluster the given data.

As shown in Figure 1, metapatterns serve as the link between the inductive and deductive aspects of knowledge discovery, effectively facilitating a deductive-inductive discovery loop. Metapatterns outline the data collecting strategy for the deductive part of the loop; they serve as the basis for the generation of specific queries, which are obtained by instantiating the variables in the metapatterns with values representing tables and columns in the database of interest. These instantiated queries are then run against the database to collect relevant data. Similarly, metapatterns also serve as a generic description of the class of patterns to be discovered and help guide the process of data analysis in the inductive part of the loop. The patterns discovered from the database adhere to the format of the current metapattern.

Metapatterns can be specified by human experts or alternatively, they can be automatically generated from the database schema. Either way, they serve as a very important interface between human "discoverers" and the discovery system. Using metapatterns, human experts can focus the discovery process onto more profitable areas of the database; the system generated metapatterns provide valuable clues to the human expert regarding good start points for the database searches and also serve as the evolutionary basis for the development of user specified metapatterns more attuned to the discovery goals as envisaged by the human expert.

For time sequence databases, metapatterns provide a flexible way to carve the data in a particular way and give inspirations to human analysts. Using metapatterns, we effectively find a nice angle to view the sequences and compare positive with negative sequences. Such views and comparison helped us to develop a new technology called *time-sequence fault-detector*. The main idea behind the detector is to learn a *typical positive sequence* for each control parameter from the classified sequences, along with *thresholds* for allowed deviations of values, durations, and time shifts. This typical sequence and the thresholds are then used on-line as a basis to detect faults in new and unclassified sequences incrementally. At the present, this detector is realized as a new action for metapatterns (composed from previous primitive actions), and performs well on the example sequences that are currently available to us. Ultimately, this detector will perform on-line on Motorola's semiconductor plasma

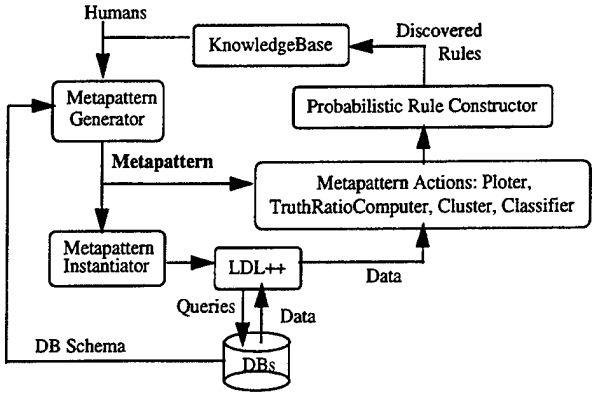

Figure 2: The Current Implementation of Metapattern Mechanism

machines to detect deficiencies of operations on wafers.

In the rest of this paper, Section 2 gives an overview of the metapattern mechanism, Section 3 describes in detail the problems we face in fault-detection in Plasma processes, Section 4 provides our initial solutions and results, and finally, Section 5 concludes the paper with discussions and future work.

2 The Metapattern Mechanism

Figure 2 illustrates the current implementation of the metapattern mechanism. The deductive part of the system is a deductive database technology called $\mathcal{LDL}$++ (Naqvi and Tsur 1989; Arni, Ong, Tsur, and Zaniolo 1994), and the inductive part of the system includes a set of actions such as *TruthRatioComputer, Ploter, Cluster*, and *Classifier*. Given a database and its schema, the metapattern generator suggests a series of valid metapatterns. Users can either pick some of the suggested metapatterns or they can specify their own metapatterns to be run. On being provided with a metapattern as input, the instantiator generates a set of corresponding $\mathcal{LDL}$++ queries by instantiating the predicate variables in the metapattern with relevant table names and column names. These $\mathcal{LDL}$++ queries are then run against the database to gather the data. The resulting data is fed to one or more of the metapattern actions, which acts on the data and returns appropriate results. These results are then used by the Probabilistic Rule Constructor to build final patterns in the format specified in the current metapattern. If desirable, the discovered patterns can be stored persistently in a online knowledge base for various uses.

To illustrate the mechanism, consider, for example, a database that has 4 tables, each with an arity of 2. Suppose that table1 and table2 specify ingredients of chemical compounds, and table3 and table4 specify properties of compounds (These facts can be presented to the system by experts or be extracted automatically from the schema by the metapattern generator). If we are interested in finding the relation between ingredients and properties, then a proper metapattern is as follows:

$$Ingredients(X, c_1, c_2) \wedge Property(X, Y) \Rightarrow Cluster(Y)$$

where *Ingredients* and *Property* are variables for table and column names, X and Y are variables for data, c_1 and c_2 are constants, and *Cluster* is a metapattern action. Given this metapattern, the instantiator instantiates it into a series of $\mathcal{LDL}$++ queries with variables bound to appropriate table and column names. In the current example, such $\mathcal{LDL}$++ queries are:

$$Q1(Y) \leftarrow table1(X, c1),\ table2(X, c2),\ table3(X, Y)$$
$$Q2(Y) \leftarrow table1(X, c1),\ table2(X, c2),\ table4(X, Y)$$

Each rule is run against the database, and a set of values that satisfy the constraints on the right-hand side of the rule is fed into the Cluster action. The cluster classifies these values into a set of classes, each with a mean value, a variance, and a likelihood. For example, when c_1='BX89' and c_2='GF102', the Y values that satisfy the first rule may be classified into two classes: (m_1, v_1, l_1) and (m_2, v_2, l_2), where m_i are mean values, v_i are variances, and l_i are the likelihood of the class.

These clustering results are then fed into the Rule Constructor, which uses the current metapattern as a template to generate a final rule. For example, suppose the two classes above are: (m_1=2.4, v_1=3.5, l_1=0.97) and (m_2=202.0, v_2=0.5, l_2=0.03), then the final rule constructed using the metapattern may look like this:

$$\text{Ingredients}(X,\text{BX89},\text{GF102}) \wedge \text{table3}(X, Y) \Longrightarrow$$
$$\text{Clusters}(Y) \equiv \{(2.4, 3.5, 0.97), (202.3, 0.5, 0.03)\}$$

This rule says that for all compounds that contains ingredients 'BX89' and 'GF102', the majority values Y of their property "table3" are near 2.4, but there are few values scattering around 202.3. This type of rules can be used in multiple purpose. They can help chemists to design new compounds with certain properties (e.g., adding sugar makes a dish sweat), or they can signal the potential errors or exceptional values in the database. For example, those values that around 202.3 in the above example can very well be errors because their occurrence has such a low probability.

Up to date, the metapattern mechanism has had three applications: discovering patterns from a large common-sense knowledge base, from a telecommunication database containing a large number of real telephone circuits, and from a chemical research database representing over 30 years of chemical research experiments. Interested readers can find more details in (Shen 1992; Shen, Ong, Mitbander and Zaniolo 1994).

3 The Problem of Plasma Process Faulty Detection

The data to be analyzed in this paper are time sequences from the manufacture of semiconductor wafers. Like many other industrial processes, wafer manufacture requires very tight process control, yet contains some element of "black art". The extremely high costs of the manufacturing equipment and infrastructure, as well as the nature of the industry provide strong motivations for improving the efficiency of the process, the quality of the products, and yield.

Semiconductor wafer manufacture consists of four main operations performed several times over. These operations are: growth or deposition, patterning or photolithography, etching, and diffusion or implantation. Each operation consists of multiple steps during which the wafer is subject to specific physical and chemical conditions according to a recipe. Testing the unfinished product between manufacturing steps is expensive and difficult. Reworking a bad product is almost impossible. This leads to two problems. First, when a problem occurs at a particular step, it may go undetected till final test is performed, thereby tying up downstream processing on a product that has already been doomed to be scraped. Second, when final test indicates that a product is of bad quality, it is usually difficult to determine which single step in the manufacture is the source of the problem.

Both of these problems would be solved if it were possible to collect the physical and chemical conditions (called in-process data) of wafer processing, and to automatically determine the success or failure of each manufacturing step by inspecting this data. Until recently, it was difficult to access the in-process data for most semiconductor manufacturing operations. Recent efforts by semiconductor equipment manufacturers and semiconductor wafer manufacturers have resulted in the establishment of a common interface (SECS: Semiconductor Equipment Communications Standard) through which different manufacturing tools can make their in-process data available.

The thrust of this work is to use the metapattern mechanism outlined above to correlate in-process data from the etch operation to the quality of the etch. We specifically study metal etch using reactive ion etch techniques in plasma etchers.

A reactive ion etching operation is a process in which reactive gas plasma ions are accelerated to the wafer surface where both chemical reaction and sputtering take place in a controlled manner to produce the desired etch profile. Typically, etch follows a photolithography operation. In the case of metal etch, a wafer is covered with metal in a metalization step. Then the desired patterns are drawn using photolithography. Finally, etching is used to remove the excess metal, leaving behind the required patterns of metal.

Sensors within a plasma etcher measure several hundred parameters. A subset of these parameters are periodically captured (approximately once a second) and made available as in-process data. This constitutes the time sequence data. In a typical scenario, a wafer can spend more than 200 seconds in an etcher. A hundred parameters measured once a second yield a maximum of 20,000 measurements for a single wafer. The large amount of time sequenced data make this a particularly interesting application of machine learning techniques.

Technically, the task we face is that given a set of positive and negative wafer etching operation examples, each being represented as about 100 data sequences (one sequence per parameter) with more than 200 time steps, to induce a set of probabilistic rules that can detect defective wafers during their manufacture process in real-time. Such detection may occur even before a manufacture operation is completed so that timely corrections can be made to the process to minimize the loss of productivity.

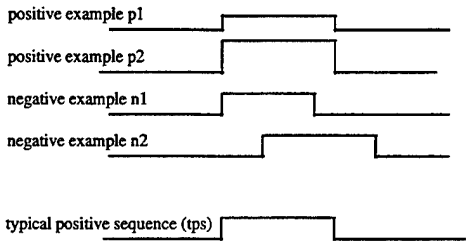

Figure 3: Constructing a Typical Positive Sequence from Examples

4 The Time-Sequence Fault-Detector

The data collected from wafer etching operations have a large quantity and high dimensions. We have 65 example operations, each has about 100×200 values (parameters × time steps). These pose challenges to most off-shelve machine learning technologies, for most of them prefer training data as attribute-value pairs. In order to use them effectively, one would have to treat each wafer operation (1) as a single, huge training example that has 100×200 attributes; (2) as 200 training examples, one for each time step, with 100 parameters as attributes; (3) as 100 training examples, one for each parameter, with 200 time steps as attributes; or (4) some combinations of the above. None of these, however, could capture the temporal information in the sequence data.

To solve this problem, we used various metapatterns to view the data in order to get useful observations. After learning from experts that operations with the same "recipe number" ought to have similar behavior, we found one metapattern very useful:

$$recipe\_number(W, c_1) \wedge Parameter(W, V) \Rightarrow Ploter(V)$$

where *recipe_number* is a specific parameter, *Parameter* is a variable for any other parameter in the operation, W is a variable for wafers, c_1 is constant, V is a variable for values, and $Ploter(V)$ is a metapattern action that plots the sequence corresponding to V. This metapattern allows us to visualize a single parameter space (i.e., a set of "similar" sequences) across all training operations (both positive and negative) that have the same recipe number.

From the plotted sequences, we notice that they indeed behave similarly and they seem to share some underlying regularity along the time line. (For some parameters, the behavior is so "similar" that there is no difference between the positives and the negatives.) Furthermore, the negative sequences seem to deviate more from the regularity then the positive sequences. From these observations, we realize that (1) we need only to focus on the parameters that make a difference, (2) a single training "unit" should be a sequence itself (not as pairs of attribute-value), and (3) our real problem is how to approximate the underlying regularity for each parameter and characterize the deviations so that the negative sequences can be distinguished from the positive ones.

To approximate the underlying regularity for a parameter, we take a simple approach by

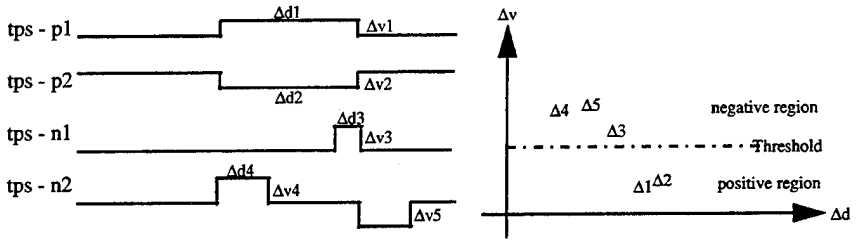

Figure 4: Learning Threshold for Allowed Deviations of Values, Durations, and Time Shifts

constructing a *typical positive sequence* (TPS) based on the average of all positive sequences. This can accomplished by our existing metapattern action *Cluster*. For example, in Figure 3, a typical sequence (tps) is induced from two positive sequences, p1, and p2, by taking the mean values along the time line.

To characterize the deviations, we notice that all deviations can be represented in terms of three "features:" the difference in values (Δh), the difference in durations (Δw), and the difference in times that changes occur (Δs). Using these features, we can represent the deviations of a sequence (relative to a TPS) as points in a two-dimensional space as follows.

First, we subtract the typical sequence from each positive and negative sequence to produce a *difference sequence* as shown in the left-hand side of Figure 4. And then, we extract Δds and Δvs from a difference sequence, where Δd represents the time deviation and Δv represents the value deviation. For example, in the difference sequence "tps-p1" in Figure 4, $\Delta d1$ is the time deviation and $\Delta v1$ is the value deviation when comparing p1 with tps. Δds and Δvs together approximate the three deviations, Δh, Δw, and Δs defined above. As illustrated in the current example, the Δds and Δvs in "tps-p1" and "tps-p2" approximate the Δh, while the Δds and Δvs in "tps-n1" and tps-n2" approximate the Δw and Δs. As the result of this procedure, the deviations of positive and negative sequences relative to a TPS are mapped onto points in a two dimensional space defined by ΔD and ΔV as shown in the right-hand side of Figure 4.

Given the points in this two-dimensional space, our task is to learn the threshold(s) between the positive points and the negative points. This again can be accomplished by our existing metapattern action *Classifier*. To complete our example, the threshold is shown in the right-hand side of Figure 4.

As we have seen now, a time-sequence fault-detector composes two elements: a typical positive sequence and a threshold. Using these two elements, we can detect potential faults in new sequences in real-time. To do so, the typical sequence is constantly compared with incoming sequence to generate time and value deviations. These deviations are then matched against the threshold(s). If any of these deviation falls on the negative side of the threshold,

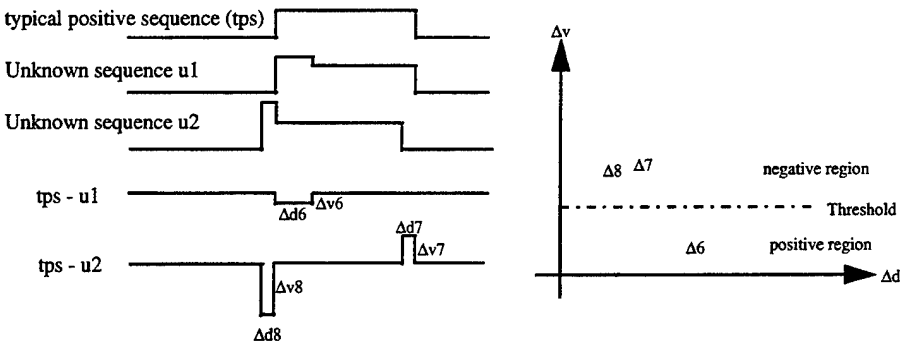

Figure 5: Online fault-detection using Typical Positive Sequence and Threshold

the detector triggers an alarm indicating that there is a possible fault in the current sequence.

Consider, for example, the two new (unknown) sequences, u1 and u2, in Figure 5. Both u1 and u2 are similar to the typical sequence tps except that they have an irregular pulse. In addition, u2 has been shifted a little to left. When comparing u2 with tps, as shown in the difference sequence, tps-u2, as soon as $\Delta d8$ and $\Delta v8$ are generated and matched against the threshold, an alarm will be triggered because this point is above the threshold. The sequence, u1, on the other hand, goes through the same process without triggering any alarm because its deviations, $\Delta v6$ and $\Delta d6$, are below the threshold.

Using this approach, we have successfully classified all the example etch operations we have. We are now waiting for more data from the manufacture for further testing and evaluation, the results of which will be reported in the near future.

5 Conclusions and Future Work

We have presented an application of the metapattern mechanism to time sequence analysis. Using metapatterns, we have effectively made some interesting observations which helped us to find a good solution to the problem. Interestingly, the solution employs several metapattern actions that already exist in the mechanism, such as *Ploter*, *Cluster*, and *Classifier*, and can be implemented as a single new metapattern:

$$recipe\_number(W, c_1) \wedge Parameter(W, V) \Rightarrow BuildFaultDetector(V)$$

where the new action *BuildFaultDetector* performs the procedure described in the last section. This metapattern results a set of fault-detection rules for the recipe number c_1 (one rule for each parameter).

The shortcoming of the approach is that parameters in an operation are assumed to be independent. This seems appropriate for the data we have so far. If this becomes an obstacle in the future, we plan to relax the assumption by assuming pairs (or more) of parameters are independent from each other so that the same approach can still be useful. However,

due to the high computational complexity, the effectiveness of this naive extension remains to be seen.

Nevertheless, the initial success of applying the metapattern mechanism to time sequence analysis has further strengthened our belief that metapattern is a powerful tool for data mining. Research and implementations are currently underway to make it an on-line service for the Plasma machines in Motorola.

References

[1] Arni, Ong, Tsur, and Zaniolo. 1994. $\mathcal{LDL}$++: A Second Generation Deductive Database System, Working Paper.

[2] Fisher, D. H., 1987. Knowledge acquisition via incremental conceptual clustering, *Machine Learning*, 2:139-172.

[3] Naqvi and Tsur. 1989. A Logical Language for Data and Knowledge Bases. W. H. Freeman Company.

[4] Shen, W.-M. 1992. Discovering Regularities from Knowledge Bases. *International Journal of Intelligent Systems*, 7(7), 623-636.

[5] Shen, W.-M. 1994. *Autonomous Learning from the Environment.* Computer Science Press. W.H. Freeman.

[6] Shen, Ong, Mitbander, and Zaniolo. 1994. Metaqueries for Data Mining. In *Advances in Knowledge Discovery and Data Mining*, Edited by Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy. MIT Press.