Discovering Unexpected Patterns in Temporal Data Using Temporal Logic

Gideon Berger
Courant Institute
New York University

Alexander Tuzhilin
Leonard N. Stern School of Business
New York University

1998

Working Paper Series
Stern #IS-98-7

Discovering Unexpected Patterns in Temporal Data Using Temporal Logic

Gideon Berger and Alexander Tuzhilin *

[1] Computer Science Department
Courant Institute
New York University
gideon@cs.nyu.edu
[2] Information Systems Department
Stern School of Business
New York University
atuzhili@stern.nyu.edu

Abstract. There has been much attention given recently to the task of finding interesting patterns in temporal databases. Since there are so many different approaches to the problem of discovering temporal patterns, we first present a characterization of different discovery tasks and then focus on one task of discovering interesting patterns of events in temporal sequences. Given an (infinite) temporal database or a sequence of events one can, in general, discover an infinite number of temporal patterns in this data. Therefore, it is important to specify some measure of interestingness for discovered patterns and then select only the patterns interesting according to this measure. We present a probabilistic measure of interestingness based on *unexpectedness*, whereby a pattern P is deemed interesting if the ratio of the *actual* number of occurrences of P exceeds the *expected* number of occurrences of P by some user defined threshold. We then make use of a subset of the propositional, linear temporal logic and present an efficient algorithm that discovers unexpected patterns in temporal data. Finally, we apply this algorithm to synthetic data, UNIX operating system calls, and Web logfiles and present the results of these experiments.

1 Introduction

There has been much work done recently on pattern discovery in temporal and sequential databases. Some examples of this work are [14, 27, 17, 10, 25, 16, 8, 18, 9, 22]. Since there are many different types of discovery problems that were addressed in these references, it is important to characterize these problems using some framework. One such characterization was proposed in [10]. In this chapter

* This work was supported in part by the NSF under Grant IRI-93-18773.

O. Etzion, S. Jajodia, and S. Sripada (Eds.): Temporal Databases - Research and Practice
LNCS 1399, pp. 281–309, 1998.

we review this framework and then focus on one specific problem of discovering *unexpected* patterns in temporal sequences. To find unexpected patterns in a sequence of events, we assume that each event in the sequence occurs with some probability and assume certain conditional distributions on the neighboring events. Based on this, we can compute an *expected* number of occurrences of a certain pattern in a sequence. If it turns out that the *actual* number of occurrences of a given pattern significantly differs for the expected number, then this pattern is certainly *unexpected* and, therefore, is interesting [23, 24]. We present an algorithm for finding such patterns and test it on several types of temporal sequences, including Web logfiles and sequences of OS system calls.

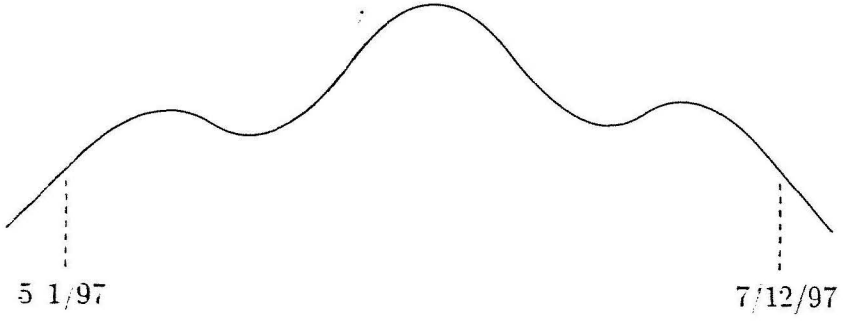

Fig. 1. An example of the *head_and_shoulder* pattern.

2 Characterization of Knowledge Discovery Tasks in Temporal Databases

Characterization of knowledge discovery tasks in temporal databses, proposed in [15] is represented by the 2-by-2 matrix presented in Table 1. The first dimension in this matrix defines the two types of temporal patterns. The first type of a temporal pattern specifies how data changes over time and is defined in terms poral *predicates*. For example, the pattern

$$head\_and\_shoulder(IBM, 5/1/97, 7/12/97)$$

es that the stock of IBM exhibited *head_and_shoulder* trading pattern [15] /1/97 until 7/4/97, as is shown in Figure 1). The second type of temporal is is *rules*, such as "if a stock exhibits a head-and-shoulder pattern and r cash levels are low, then bearish period is likely to follow."

e second dimension, the *validation/generation* dimension, refers to the e of the discovery task. In *validation* the system focuses on a particular

pattern and determines whether it holds in the data. For example, we may want to validate if the *head_and_shoulders* pattern holds for the IBM stock in a given data set or that a certain rule "holds" on the data. The second purpose of discovery can be the *generation* of new predicates or rules that are previously unknown to the system. For example, the system may attempt to discover new types of trading rules in financial applications.

Categorizing patterns in terms of the above two dimensions leads to a two-by-two classification framework of the knowledge discovery tasks, as presented in Table 1. We will describe each of the four categories in turn now.

	Validation	Generation
Predicates	I	III
Rules	II	IV

Table 1. Types of Knowledge Discovery Tasks.

Class I. The discovery tasks of this type involve the validation of previously defined predicates over the underlying database. For example, assume that we have the temporal database of daily closing prices of stocks at some stock exchange, STOCK(SYMBOL,PRICE,DATE), where SYMBOL is the symbol of a security, PRICE is the closing price of that stock on the date DATE. Consider the following predicate specifying that the price of a certain stock bottomed out and is on the rise again over some time interval:

$$bottom\_reversal(x, t_1, t_2) = (\exists t)(t_1 < t < t_2 \wedge decrease(x, t_1, t) \wedge increase(x, t, t_2))$$

where $increase(x, t_1, t_2)$ and $decrease(x, t_1, t_2)$ are predicates specifying that the price of security x respectively "increases" and "decreases" over the time interval (t_1, t_2)[1].

Then we may want to *validate* that the predicate $bottom\_reversal(x, t_1, t_2)$ holds on the temporal relation STOCK(SYMBOL,PRICE,DATE). This validation can take several forms. For example, we may want to find for the predicate *bottom_reversal* if one of the following holds:

$$bottom\_reversal(IBM, 5/7/93, 8/25/93),$$
$$bottom\_reversal(IBM, t1, t2),$$
$$bottom\_reversal(x, 5/7/93, 8/25/93)$$

[1] Note that we do not necessarily assume monotonic increases and decreases. Predicates *increase* and *decrease* can be defined in more complex ways, and we purposely leave it unspecified how to do this.

The first problem validates that the stock of IBM experienced the "bottom reversal" pattern between 5/7/93 and 8/25/93. The second problem finds all the time periods when IBM's stock had "bottom reversal," and the last problem finds all the stocks that had "bottom reversals" between 5/7/93 and 8/25/93.

One of the main issues in the problems of Class I (predicate validation problem) is to find *approximate* matching patterns. For example, for the IBM stock to exhibit the bottom reversal pattern between 5/7/93 and 8/25/93, it is not necessary for the time series of IBM stock to match predicate *bottom_reversal* exactly. Another example of the approximate matching problem of Class I comes from the speech recognition applications where sounds and words are matched only approximately against the speech signal.

There has been extensive work done on Class I problems in signal processing [20], speech recognition [6, 21], and data mining communities. In the data mining community these types of problems are often referred as similarity searches and have been studied in [1, 3, 4, 12, 13, 8].

Class II. Discovery tasks of Class II involve validation of previously asserted *rules*. For example, consider the rule: "If a price correction in a stock is seen before the announcement of big news about the company, then insider trading is likely."

$$Correction(stock, t_1, t_2) \wedge Big\_news(stock, t_3) \wedge Soon\_after(t_3, t_2) \\ \rightarrow Insider\_trading(stock, t_1, t_2)$$

where *Correction*, *Big_news*, *Insider_trading* and *Soon_after* are user-defined predicates (views) defined on relations STOCKS and NEWS.

Evaluation of this rule on the data entails finding instances of variables *stock*, t_1, t_2, t_3 and the "statistical strength" of the rule (e.g. measured in terms of its confidence and support [2]) that make the rule hold on the data (in statistical terms).

As in the case of Class I problems, one of the main issues in rule validation is the problem of approximate matching. The need for approximate matching arises for the following reasons. First of all, rules hold on data only in statistical terms (e.g. having certain levels of confidence and support). Secondly, some of the predicates in the rule can match the data only approximately (as is the case with Class I problems from Table 1). Moreover, certain temporal operators are inherently fuzzy. For example, temporal operator $Soon\_after(t_1, t_2)$ is fuzzy and

to be defined in "fuzzy" terms[2].

ass III. Discovery tasks of Class III involve the discovery of new interesting ate-based patterns that occur in the database. In order to discover such ns, the system should know on what it should focus its search because are potentially very many new patterns in the database. In other words, stem should know what to look for by letting the user specify what is

e that it is not appropriate to define this operator in terms of the temporal ator *Next* because of the inherent ambiguity of the term "soon." Although this ator can be defined in many different ways, one natural approach would be ugh the use of fuzzy logic [28].

interesting. For example, the pattern *bottom_reversal* may be interesting because it provides trading opportunities for the user.

Although there are many different measures of interestingness for the user, such as frequency, unexpectedness, volatility, and periodicity [10], the most popular measure used in the literature is *frequency* of occurrence of a pattern in the database [17, 16, 18]. In particular, [17, 16] focus on discovering frequent *episodes* in sequences, whereas [18] discovers frequent patterns in temporal databases satisfying certain temporal logic expressions.

In this chapter, we use a different measure of interestingness. Instead of discovering frequent patterns in the data, we attempt to discover *unexpected* patterns. While it is sometimes the case that the discovery of frequent patterns offers useful insight into a problem domain, there are many situations where it does not. Consider, for example, the problem of intrusion detection on a network of workstations. Assume we define our events to be operating system calls made by some process on one of these workstations. We conjecture, then, that patterns of system calls differ for ordinary users as opposed to intruders. Since intrusion is a relatively rare occurrence the patterns we would discover using frequency as our measure of interestingness would simply be usage patterns of ordinary users offering us no information about intrusions. Instead what we propose is to assign exogenous probabilities to events and then attempt to discover patterns whose number of occurrences differs by some proportion what would be expected given these probabilities. In the example of intrusion detection we would assign the probabilities of events to reflect the frequency of events in the presence of no intruders. Then if an intrusion did occur, it would presumably cause some unexpected pattern of system calls which can be an indication of this event.

As will be demonstrated in Section 3, the new measure of interestingness requires discovery techniques that significantly differ from the methods used for the discovery of frequent patterns. The main reason for that is that unexpected patterns are not monotone. These notions will be made more precise in Section 3.

Class IV. Discovery tasks of Class IV involve discovery of new rules consisting of interesting relationships among predicates. An example of a temporal pattern of this type is the rule stating that "If a customer buys maternity clothes now, she will also buy baby clothes within the next few months."

Discovery tasks of Class IV constitute challenging problems because, in the most general case, they contain problems of Class III (discovery of new predicates) as subproblems. The general problem of discovering interesting temporal rules using the concept of an *abstract* [11] has been studied in [7]. Discovery of temporal association rules was studied in [5, 25].

In this section, we reviewed a characterization of knowledge discovery tasks, as presented in [10]. In the rest of this chapter, we will focus on one specific Class III problem dealing with discovery of unexpected patterns. In the next section, we will formulate the problem. In Section 4 we will present an algorithm for finding unexpected patterns, and in Section 5 we will present experiments evaluating this algorithm on several applications.

3 Discovering Unexpected Patterns in Sequences: The Problem Formulation

We start this section with an intuitive presentation of the problem and then provide its more formal treatment.

We want to find unexpected patterns, defined in terms of temporal logic expressions, in sequences of events. We assume that each event in the sequence occurs with some probability and assume certain conditional distributions on the neighboring events. Based on this, we can compute an *expected* number of occurrences of a certain pattern in a sequence. If it turns out that the *actual* number of occurrences of a given pattern significantly differs for the expected number, then this pattern is certainly *unexpected* and, therefore, is interesting [23, 24].

In this chapter, we first present a naive algorithm that finds all unexpected patterns (such that the ratio of the actual number of occurrences to the expected number of occurrences exceeds a certain threshold). After that, we present an improved version of the algorithm that finds most of the unexpected patterns in a more efficient manner. We also experimentally compare the naive and the more efficient algorithms in terms of their performance.

More formally, let $E = \{\alpha, \beta, \gamma, \ldots\}$ be a finite alphabet of events. We use a subset of propositional linear temporal logic to discover temporal patterns over the events. The basic temporal operators of this system are $\alpha \mathbf{B}_k \beta$ (α before$_k$ β) which intuitively means that α occurs followed by an occurrence of α within k subsequent events, $\alpha \mathbf{N} \beta$ (α next β) α occurs and the next event is β, and $\alpha \mathbf{U} \beta$ (α until β) which means before β occurs a sequence of α's occurs. This is often called the *strong until* [26]. While the before operator is actually redundant as $\alpha \mathbf{B} \beta$ can be expressed as $\neg(\neg\alpha \mathbf{U} \beta)$ we have chosen to include it separately for simplicity and efficiency. A pattern of events is defined as a conjunction of ground events over these operators. For example, the simplest case is $\alpha \mathbf{N} \beta$. Some additional examples are $(\delta \mathbf{U}((\alpha \mathbf{N} \beta) \mathbf{B} \gamma))$ and $\alpha \mathbf{N} \beta \mathbf{N} \gamma$.

In the pattern discovery algorithm presented in Section 4.2 we consider the following fragment of the Propositional Temporal Logic (PLTL). The syntax of this subset is as follows. The set of formulae of our subset is the least set of formulae generated by the following rules:

(1) each atomic proposition P is a formulae;

(2) if p is a formula and q is a formula containing no temporal operators then $p\mathbf{B}_K q$, $p\mathbf{N}q$, $q\mathbf{U}p$, $q\mathbf{B}_K p$, $q\mathbf{N}p$ are formulae.[3]

: assume an exogenous probability distribution over the events. While these may be dependent or independent, depending on the problem domain of

gnore disjunctions because what seems to occur in practice when disjunctions are ved is that the disjunction of a very interesting pattern, E, with an uninteresting ern, F, results in an interesting pattern $E \vee F$. This occurs not because $E \vee F$ truly s any insight into our problem domain but rather because the interestingness] "drags up" the interestingness measure of $E \vee F$ to the point where it also mes interesting. We choose instead to simply report E as an interesting pattern. decision to omit conjuctions and negation will be made clear shortly.

interest we assume independence of the events unless explicitly stated otherwise. For instance, in the application we consider in Section 5.3, events are described as hits on Web pages. In this case the probability that a user goes from Web page P to Web page Q is clearly dependent on the links that exist on page P. In other cases independence may be more appropriate. In any case, given an a priori set of event probabilities, we can compute expected values for the number of occurrences of any temporal pattern in our string. For example, the expected number of occurrences of $\mathsf{E}[\![\alpha \mathbf{B} \beta]\!]$, assuming the events α and β are independent, can be computed as follows. Let X_n be the number of occurrences of the pattern $\alpha \mathbf{B} \beta$ up to the n^{th} element of the input string and α_n the number of α's up to the n^{th} element of the input string. Then

$$\begin{aligned}\mathsf{E}[\![X_n]\!] &= \mathsf{Pr}[\![\beta]\!][X_{n-1} + \alpha_{n-1}] + (1 - \mathsf{Pr}[\![\beta]\!])(X_{n-1}) \\ &= \mathsf{Pr}[\![\beta]\!][X_{n-1} + \mathsf{Pr}[\![\alpha]\!](n-1)] + (1 - \mathsf{Pr}[\![\beta]\!])(X_{n-1}) \\ &= \mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!](n-1) + X_{n-1}\end{aligned}$$

Therefore,

$$\mathsf{E}[\![X_n]\!] - \mathsf{E}[\![X_{n-1}]\!] = \mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!](n-1)$$

Also, $\mathsf{E}[\![X_2]\!] = \mathsf{Pr}[\![\alpha]\!] * \mathsf{Pr}[\![\beta]\!]$. From this recurrence equation, we compute $\mathsf{E}[\![\alpha \mathbf{B}_K \beta]\!]$ for the input string of length N as

$$\mathsf{E}[\![\alpha \mathbf{B} \beta]\!] = \frac{\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]N(N-1)}{2}$$

The expected number of occurrences of patterns of other forms can be similarly computed as

$$\mathsf{E}[\![\alpha \mathbf{N} \beta]\!] = \mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!](N-1) \tag{1}$$

$$\mathsf{E}[\![\alpha \mathbf{B}_K \beta]\!] = \mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!](K)(N-K) + \frac{\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!](K)(K-1)}{2}$$

$$\mathsf{E}[\![\alpha \mathbf{U} \beta]\!] = \frac{\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]}{1 - \mathsf{Pr}[\![\alpha]\!]} \sum_{i=2}^{N-1} 1 - \mathsf{Pr}[\![\alpha]\!]^i + \mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]$$

As was stated earlier, we will search for the unexpected temporal patterns in the data, where unexpectedness is defined as follows:

Definition 1 *Let P denote some temporal pattern in string S. Let $\mathsf{A}[\![P]\!]$ be the actual number of occurrences and $\mathsf{E}[\![P]\!]$ the expected number of occurrences of pattern P in S. Given some threshold T, we define a pattern P to be* unexpected

if $\frac{\mathsf{A}[\![P]\!]}{\mathsf{E}[\![P]\!]} > T$. *The ratio* $\frac{\mathsf{A}[\![P]\!]}{\mathsf{E}[\![P]\!]}$ *is called the* Interestingness Measure (IM) *of the pattern* P *and will be denoted as* IM(P). [4]

This is a probabilistic measure of interestingness whereby a pattern is unexpected if its actual count exceeds its expected count by some proportion T. As the following theorem indicates, however, this is a difficult problem.

Problem (INTERESTINGNESS):
Given a string of temporal events $V = v_1, v_2, \ldots, v_r$, does there exist an interesting pattern in V of the form $X_1 \mathbf{B_k} X_2 \mathbf{B_k} \ldots \mathbf{B_k} X_m$ for an arbitrary m?

Theorem 1 *The INTERESTINGNESS problem is NP-complete.*

Proof: See Appendix.

While we are trying to find interesting patterns that contain a variety of temporal operators in an arbitrary order, this theorem states that finding interesting patterns that only use the BEFORE operator is hard. Furthermore, we would like to put no restrictions on the "interesting" patterns we discover. We would simply like to find *all* patterns that are interesting. The following theorem, however, shows that it is necessary to impose some bounds on the size of the patterns that we uncover, since in the case of unrestricted patterns, the most unexpected pattern will always be the entire string.

Theorem 2 *Consider a string of temporal events* $V = v_1, v_2, \ldots, v_N$ *and a temporal pattern* T. *If the length of* T *(number of temporal operators in it), length(*T*)* $<$ N - 1, *then there exists another pattern* P *such that length(*P*)* $=$ *length(*$T+1$*) and* $IM(P) \geq IM(T)$, *where the length of a pattern is defined as the number of events in the pattern.*

Proof:
Let $\mathsf{A}[\![T]\!] = \beta$ and $\frac{\mathsf{A}[\![T]\!]}{\mathsf{E}[\![T]\!]} = \alpha$ and $Z = \{z_1, z_2, \ldots, z_m\}$ the set of all events.

We want to prove that $\exists\ z_i \in Z$ s.t. $\frac{\mathsf{A}[\![T\mathbf{N}z_i]\!]}{\mathsf{E}[\![T\mathbf{N}z_i]\!]} \geq \alpha$

Assume this is not true for $z_1, z_2, \ldots, z_{m-1}$ and show that it must be true for z_m. By this assumption and because of (1)

$$\frac{\mathsf{A}[\![T\mathbf{N}z_i]\!]}{\mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_i]\!](N-1)} < \alpha \qquad \forall z_i, i = 1, 2, \ldots, m-1.$$

ɔre, $\mathsf{A}[\![T\mathbf{N}z_i]\!] < \alpha \mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_i]\!](N-1)$.

her measure of interestingness is to find patterns P for which $\mathsf{A}[\![P]\!]/\mathsf{E}[\![P]\!] < T$. problem can be treated similarly. We have chosen not to search for these patterns ıse they are complimentary to the ones described in Definition 1. If a pattern ; found to be interesting in our formulation then P will be interesting in this limentary formulation for some new threshold. Thus in the interest of simplicity oose to solve these complimentary problems separately and ignore negation.

Then,

$$\sum_{i=1}^{m-1} \mathsf{A}[\![T\mathbf{N}z_i]\!] < \sum_{i=1}^{m-1} \alpha \mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_i]\!](N-1)$$
$$= \alpha \mathsf{Pr}[\![T]\!](N-1) \sum_{i=1}^{m-1} \mathsf{Pr}[\![z_i]\!]$$
$$= \alpha \mathsf{Pr}[\![T]\!](N-1)(1 - \mathsf{Pr}[\![z_m]\!])$$

Since, $\sum_{i=1}^{m} \mathsf{A}[\![T\mathbf{N}z_i]\!] = \mathsf{A}[\![T]\!] = \beta$,

$$\mathsf{A}[\![T\mathbf{N}z_m]\!] > \beta - \alpha \mathsf{Pr}[\![T]\!](N-1)(1 - \mathsf{Pr}[\![z_m]\!])$$
$$\frac{\mathsf{A}[\![T\mathbf{N}z_m]\!]}{\mathsf{E}[\![T\mathbf{N}z_m]\!]} > \frac{\beta - \alpha \mathsf{Pr}[\![T]\!](N-1)(1 - \mathsf{Pr}[\![z_m]\!])}{\mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_m]\!](N-1)}$$
$$= \frac{\beta}{\mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_m]\!](N-1)} - \frac{\alpha \mathsf{Pr}[\![T]\!](N-1)(1 - \mathsf{Pr}[\![z_m]\!])}{\mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_m]\!](N-1)}$$
$$= \frac{\beta}{\mathsf{Pr}[\![T]\!]\mathsf{Pr}[\![z_m]\!](N-1)} - \frac{\alpha(1 - \mathsf{Pr}[\![z_m]\!])}{\mathsf{Pr}[\![z_m]\!]}$$
$$\left(\text{since } \frac{\beta}{\mathsf{E}[\![T]\!]} = \frac{\beta}{\mathsf{Pr}[\![T]\!](N-1)} = \alpha\right)$$
$$= \frac{\alpha}{\mathsf{Pr}[\![z_m]\!]} - \frac{\alpha(1 - \mathsf{Pr}[\![z_m]\!])}{\mathsf{Pr}[\![z_m]\!]}$$
$$= \alpha \qquad \square$$

Intuitively, this theorem tells us that given an interesting temporal pattern, there exists a longer pattern that is more interesting. In the limit then, the most interesting pattern will always be the entire string of events, as it is the most unlikely.

In order to cope with this, we restrict the patterns that we look for to be of length less than or equal to some length limit. Of course, still the most interesting pattern we will find will be one whose length is equal to the length limit. Nevertheless, it is often the case that an interesting pattern that is not the most interesting provides valuable insight into a given domain as we will see later in discussing our experiments.

4 Algorithm

4.1 Naive Algorithm

A naive approach to discovering interesting patterns in an input sequence might proceed as follows. Sequentially scan over the input string discovering new patterns as we go. When a new pattern is discovered a record containing the pattern itself as well as a count of the number of occurrences of the pattern is appended

to a list of all discovered patterns. This is repeated until all patterns up to a user-defined maximum length, have been found. More precisely, the algorithm proceeds as follows

Definition 2 *BEFOREK: A user defined constant that determines the maximum number of events that X can precede Y by, for $X\mathbf{B_K}Y$ to hold.*

Input:

- Input String
- Event Probabilities: the exogenously determined probabilities of each atomic event.
- BEFOREK
- The threshold T for interestingness. That is the value that, if exceeded by the interestingness measure of a pattern, deems it interesting.
- Maximum allowable pattern length (MAXL).

Output:

- All discovered patterns P such that $IM(P) > T$.

Algorithm:

```
Scan the input string to determine the interestingness measure
of each event in it, and initialize list L with all these
events
WHILE L is not empty DO
   Amongst all the patterns of L, choose the pattern C
     with the largest interestingness measure as the next
     candidate to be expanded.
   Expand C as follows. Scan the input string looking
     for occurrences of C.  When an instance of C is
     discovered, expand it both as a prefix and as a
     suffix. By this we mean, record all occurrences of
     (C op X) and (X op C) where op ranges over the temporal
     operators, and X ranges over all events. Finally,
     compute the interestingness of all these newly
     discovered patterns C'.
   IF Length(C') < MAXL THEN add C' to the list L.
   Remove C from L.
ND WHILE
utput interesting patterns.
```

ote that the algorithm we just presented is tantamount to an exhaustive and is therefore not very efficient. We propose a more efficient algorithm, although is not guaranteed to find *all* interesting patterns, offers speed up minimal loss of accuracy. The idea is to expand on the approach presented of beginning with small patterns and expanding only those that offer the tial of leading to the discovery interesting, larger patterns.

4.2 Main Algorithm

The difficulty involved in finding interesting patterns is in knowing where to look. When interestingness is measured simply by some count (i.e. the number of occurrences exceeds some threshold) as is done in [17] it is obvious that for a pattern to be frequent so must its component partial patterns be frequent. With this in mind, the technique that has been used in [17] is to expand all patterns whose count exceeds this threshold and stop when no more exist. When using our interestingness measure, however, this is not the case. That is, a pattern can be unexpected while its component sub-patterns are not. This lack of monotonicity in our interestingness measure is most easily understood with an example.

Example: Let the set of events be $E = \{A, B, C\}$. Assume the probability of these events is $\mathsf{Pr}[\![A]\!] = 0.25, \mathsf{Pr}[\![B]\!] = 0.25, \text{and} \mathsf{Pr}[\![C]\!] = 0.50$. Also assume that these events are independent. Let the threshold $T = 2$. In other words, for a pattern to be interesting the value of the actual number of occurrences of the pattern divided by the expected number of occurrences of the pattern must exceed 2.0. Consider the following string of events.

```
ABABABABCCCCCCCCCCCC
```

(the length of this string $N = 20$)

Given our probabilities, $\mathsf{E}[\![A]\!] = 5$ and $\mathsf{E}[\![B]\!] = 5$. Also given the expression for computing expectations for patterns of the form $A\mathbf{N}B$.

$$\begin{aligned}\mathsf{E}[\![A\mathbf{N}B]\!] &= \mathsf{Pr}[\![A]\!]\mathsf{Pr}[\![B]\!](N-1)\\ &= (0.25)(0.25)(19)\\ &= 1.1875\end{aligned}$$

Since $\mathsf{A}[\![A]\!] = 4$ and $\mathsf{A}[\![B]\!] = 4$, both of the events A and B are not interesting (in fact the actual number occurrences of these events was less than what was expected), but the pattern $A\mathbf{N}B$ which occurred 4 times was interesting with

$$\begin{aligned}IM(A\mathbf{N}B) &= \frac{4}{1.1875}\\ &= 3.37\end{aligned}$$ □

This lack of monotonicity in our interestingness measure results in a significantly more complex problem especially in terms of space complexity. In the algorithm for discovering frequent patterns significant pruning of the search space can occur with each iteration. That is, when a newly discovered pattern is found to have occurred fewer times than the frequency threshold, it may be discarded as adding new events to it can*not* result in a frequent pattern. With our measure of interestingness, however, this is not the case. The addition of an event to an uninteresting pattern *can* result in the discovery of an interesting one. This inability to prune discovered patterns leads to an explosion in the amount of space

required to find unexpected patterns. Consequently we are limited to expanding patterns by only single literals at a time and therefore will not discover patterns like $((\alpha \mathbf{B_K} \beta)\mathbf{B_K}(\gamma \mathbf{N} \delta))$, where two patterns of size greater than one are combined via a temporal operator (before, in this example). This is the reason that we have not used conjunctions as part of our fragment of temporal logic. Since our events occur sequentially, it is impossible for conjunctions to arise unless we expanded patterns by multiple literals at a time. This does present a limitation of our algorithm and extending our fragment further is an area we are pursuing currently.

A more efficient algorithm than the naive one for finding unexpected patterns involves sequential scans over the string of events discovering new patterns with each scan. A list is maintained of those patterns discovered so far, and on each subsequent iteration of the algorithm the "best" pattern is selected from this list for expansion to be the seed for the next scan. When a pattern P is expanded, the input sequence is scanned and occurrences of P located. For each of these occurrences all patterns of the forms $XopP$ and $PopX$ are added to the list of discovered patterns, where op is a temporal operator, $\mathbf{N}, \mathbf{B_K}$ or. $\mathbf{U}$ and X is a variable ranging over all events.

Given a pattern to expand, $\alpha \mathbf{B_K} 3$, for example, during the scan we will discover all patterns, $((\alpha \mathbf{B_K} 3)\mathbf{N}\gamma), (\gamma \mathbf{B_K}(\alpha \mathbf{B_K} 3))$, etc... for all events γ.

The heart of the algorithm is how "best" patterns are chosen. We will explain it formally below (in Definition 4), but would like to give some intuition beforehand. Clearly, we would like to define "best" to mean most likely to produce an interesting pattern during expansion. By Theorem 1. we know that expanding an already interesting pattern must result in the discovery of additional interesting pattern(s). The question remains, however, amongst interesting patterns already discovered which is the best candidate for expansion, and if no interesting patterns remain unexpanded, are there any uninteresting patterns worth expanding?

Initially, the algorithm begins with a scan of the input string counting the number of occurrences (and therefore, the frequencies) of individual events. Subsequent to this, we continue to expand *best* candidates until there are no more candidates worthy of expansion. This notion will be made clear shortly.

During each scan of the input string, when a new pattern is discovered,[5] a PATTERN_RECORD is created for it consisting of the following information:

1. Pattern P (e.g. $((\alpha \mathbf{N} 3)\mathbf{B_K}\gamma)$), etc...
2. Count: How many of these patterns were found
3. Preremaining_op: One instance of this value is kept for each temporal operator. It represents the number of patterns remaining to be discovered for which P is the prefix and the operator connecting P to its suffix is op. How these values are calculated will be discussed shortly(see Definition 3).

[5] In the case of the initial scan these will simply be the events.

4. Postremaining_op: Identical to Preremaining_op for suffixes rather than prefixes.
5. Expanded(boolean): Whether or not P has been expanded.
6. INTERESTINGNESS_LIST: consists of all events in decreasing order of interestingness amongst events that can potentially complete P during expansion. One of these lists is kept for prefixes and one for suffixes as well as for *each* operator *next*, *before*, and *until*. That is, for a pattern $P = \alpha \mathbf{N} \beta$, for example, if a pattern $\gamma \mathbf{N} \delta$ has already been discovered then the occurrence of δ in $\gamma \mathbf{N} \delta$ cannot possibly complete the pattern $(\alpha \mathbf{N} \beta)\mathbf{N}X$. When determining the *best* candidate for expansion we will be interested in knowing what events can potentially complete all of the patterns we have already discovered and will ,therefore, make use of these lists. In fact, this sorted list represents an ordering of most interesting events that could complete the pattern they are associated with [6].

Definition 3 *The FORM(P) of a pattern P is a logical expression with all ground terms in P replaced by variables.*

For example, if $P = ((\alpha \mathbf{N}(\beta \mathbf{B_K} \gamma))\mathbf{B_K}\delta)$ then $\mathrm{FORM}(P) = (W\mathbf{N}((X\mathbf{B_K}Y)\mathbf{B_K}Z))$.

Given the length of the input string, we can determine the number of patterns of each form in the input string. For example, given a string of length M, the number of patterns of form $X\mathbf{N}Y$ is $M-1$. The number of patterns $X\mathbf{B_K}Y$ is $(M-K)K + ((K)(K-1)/(2))$.

Definition 4 *Given a pattern P and an operator op,* Actual_Remaining(P op X) *is the number of patterns of the form $PopX$ that have yet to be expanded. This value is maintained for each operator, op and pattern P. That is, we maintain a value for $P\mathbf{N}X, P\mathbf{B_K}X, X\mathbf{B_K}P, etc...$ Again, X ranges over all events.*

For example, if there are 20 occurrences of $P = \alpha \mathbf{B_K} \beta$ in the input string and 5 patterns of the form $((\alpha \mathbf{B_K} 3)\mathbf{N}X)$ have been discovered so far, then Actual_Remaining_Pre_Next $(((\alpha \mathbf{B_K} \beta)\mathbf{N}X)) = 15$.

We use the following heuristic to determine which discovered pattern is the best one to expand. Given an arbitrary literal D, the *best* pattern P for expansion is the pattern for whom the the value of

$$\mathsf{E}[\![\mathsf{A}[\![P \ op \ \delta]\!]/\mathsf{E}[\![P \ op \ \delta]\!]] \ \mathrm{or} \ \mathsf{E}[\![\mathsf{A}[\![\delta \ op \ P]\!]/\mathsf{E}[\![\delta \ op \ P]\!]]$$

is maximal for some δ.

[6] For problem domains with a large number of events, in the interest of scalability, partial lists may be substituted where only a list of the most interesting events is maintained.

This heuristic simply states that the pattern P that is most likely to result in the discovery of an interesting pattern is the one for whom there exists a literal δ such that the expected value of the interestingness measure of the pattern generated when δ is added to P via one of the temporal operators is maximal over all discovered patterns P and literals δ. It is necessary for us to use the *expected* value of the interestingness measure because, although we know the actual number of occurrences of both P and δ, we don't know the number of occurrences of P *op* δ or δ *op* P. How this expectation is computed follows directly from our derivations of expectations in Section 3 and is illustrated in the following example.

Example: If $P = \alpha \mathbf{N} \beta$ and *op* is *next*, then

$$\begin{aligned} &\mathsf{E}[\![\mathsf{A}[\![P\mathbf{N}\delta]\!]/\mathsf{E}[\![P\mathbf{N}\delta]\!]]\!] \\ &= (\#P)(\mathrm{FR}(\delta))/\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]\mathsf{Pr}[\![\delta]\!](K-2) \end{aligned}$$

where,
K = length of input string
$\mathrm{FR}(\delta)$ = frequency of δ's that could complete the pattern $((\alpha \mathbf{N} \beta)\mathbf{N}X)$
#P = number of occurrences of pattern P
If *op* is *before*,

$$\begin{aligned} &\mathsf{E}[\![\mathsf{A}[\![P\mathbf{B_K}\delta]\!]/\mathsf{E}[\![P\mathbf{B_K}\delta]\!]]\!] \\ &= ((\#P)(\mathrm{FR}(\delta))(BEFOREK))/\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\delta]\!](K-2)(BEFOREK) \\ &= ((\#P)(\mathrm{FR}(\delta)))/\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]\mathsf{Pr}[\![\delta]\!](K-2) \end{aligned}$$

If $P = \alpha \mathbf{B_K} \beta$ and *op* is *next*

$$\begin{aligned} &\mathsf{E}[\![\mathsf{A}[\![P\mathbf{N}\delta]\!]/\mathsf{E}[\![P\mathbf{N}\delta]\!]]\!] \\ &= ((\#P) * (\#\delta))/\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]\mathsf{Pr}[\![\beta]\!](K-2)(BEFOREK) \end{aligned}$$

Similar arguments are used for any combination of the operators *before*, *next*,and

consider the literal δ which is most likely to result in the discovery of an
ting pattern when used to complete the pattern P during expansion. We
ow argue that this measure accomplishes our goal of expanding patterns
ikely to result in the discovery of interesting patterns.
e choice of a *best* candidate for expansion proceeds in two stages. First,
he purpose of the INTERESTINGNESS_LIST for each discovered pattern.

efore and *until* these definitions are slightly erroneous due to losses of patterns
e ends of the input string. These errors are negligible, however, since the length
e input string is much larger than the length of individual patterns of interest

Each of the INTERESTINGNESS_LISTs associated with a pattern P is sorted in such a way that the event at the head of the list, when added to P is most likely to result in the discovery of an interesting pattern. An event D will be ahead of an event ϵ on this list if, $\mathsf{A}[\![\delta]\!]/\mathsf{E}[\![\delta]\!] > \mathsf{A}[\![\epsilon]\!]/\mathsf{E}[\![\epsilon]\!]$. While the expected values here are computed in the usual way, in this case, the actual values are not simply equal to the counts of δ and ϵ, respectively, but rather equal to the number of δ's and ϵ's that *could potentially be added to P*.

Lemma 1 *Given two events δ and ϵ where δ occurs before ϵ on the INTERESTINGNESS_LIST then:*

$$\frac{\mathsf{E}[\![\mathsf{A}[\![(\alpha \mathbf{N} \beta) op \delta]\!]]\!]}{\mathsf{E}[\![(\alpha \mathbf{N} \beta) op \delta]\!]} > \frac{\mathsf{E}[\![\mathsf{A}[\![(\alpha \mathbf{N} \beta) op \epsilon]\!]]\!]}{\mathsf{E}[\![(\alpha \mathbf{N} \beta) op \epsilon]\!]}$$

Proof: We prove this result for the *next* operator.
Assume, $\frac{\mathsf{E}[\![\mathsf{A}[\![(\alpha \mathbf{N} \beta)\mathbf{N}\delta]\!]]\!]}{\mathsf{E}[\![(\alpha \mathbf{N} \beta)\mathbf{N}\delta]\!]} \leq \frac{\mathsf{E}[\![\mathsf{A}[\![(\alpha \mathbf{N} \beta)\mathbf{N}\epsilon]\!]]\!]}{\mathsf{E}[\![(\alpha \mathbf{N} \beta)\mathbf{N}\epsilon]\!]}$
Let N be the length of the input string[8]. Then

$$\frac{\mathsf{E}[\![\mathsf{A}[\![(\#P)(\mathrm{FR}(\delta))]\!]]\!]}{\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]\mathsf{Pr}[\![\delta]\!]} \leq \frac{\mathsf{E}[\![\mathsf{A}[\![(\#P)(\mathrm{FR}(\epsilon))]\!]]\!]}{\mathsf{Pr}[\![\alpha]\!]\mathsf{Pr}[\![\beta]\!]\mathsf{Pr}[\![\epsilon]\!]}$$

Since $\#P$ and $\mathrm{FR}(\delta)$ are constants we can remove them from the expectations and cancel them on each side of the inequality. So,

$$\begin{aligned} \frac{(\mathrm{FR}(\delta))}{\mathsf{Pr}[\![\delta]\!]} &\leq \frac{(\mathrm{FR}(\epsilon))}{\mathsf{Pr}[\![\epsilon]\!]} \\ \frac{(\mathrm{FR}(\delta))(N)}{\mathsf{Pr}[\![\delta]\!](N)} &\leq \frac{(\mathrm{FR}(\epsilon))(N)}{\mathsf{Pr}[\![\epsilon]\!](N)} \\ \frac{\mathsf{A}[\![\delta]\!]}{\mathsf{E}[\![\delta]\!]} &\leq \frac{\mathsf{A}[\![\epsilon]\!]}{\mathsf{E}[\![\epsilon]\!]} \end{aligned}$$

Contradicts assumption that δ occurs before ϵ on the INTERESTINGNESS_LIST
The proofs for the temporal operators *before* and *until* are done similarly. □

So, it is now clear, for each discovered pattern P, which literal when added to P is most likely to produce an interesting pattern and how interesting we expect that pattern to be. In the second stage of choosing the *best* candidate we select the already discovered pattern which is likely to produce the most interesting pattern. Intuitively, we are saying that the pattern P most worth expanding is the one for which there exists a literal that is likely, when added to P, to result in the discovery of the most interesting pattern.

[8] Here $\mathrm{FR}(\delta)$ and $\mathrm{FR}(\epsilon)$ represents the frequencies of δ's and ϵ's respectively that could complete the pattern $((\alpha \mathbf{N} \beta)\mathbf{N}X)$. As discussed earlier this is not equal to the frequency of all of the δ's and ϵ's in the input string.

Given these preliminary motivations, we now present the algorithm:

Input:

- Input String
- Event Probabilities
- BEFOREK: as discussed earlier we use a bounded version of the before operator. BEFOREK is a user defined variable that is equal to the maximum distance between two events X and Y for $X\mathbf{B}_K Y$ to hold.
- Threshold T for interestingness, that is the value that if exceeded by the interestingness measure of a pattern deems it interesting
- Value of MIN_TO_EXPAND: the minimum threshold of interestingness that a pattern must have in order to become the next pattern for expansion. The algorithm will terminate if no such pattern remains.
- Maximum allowable pattern length

Output:

- List of interesting patterns, their number of occurrences and the value of their interestingness measures

Algorithm:

```
Scan the input string to determine the interestingness
measure of each event in it, and initialize list L with
all these events
WHILE there exists a pattern in L whose interestingness
 measure is greater than MIN_TO_EXPAND DO
   Choose_Next_Candidate
    the pattern P such that LENGTH(P) < MAXL which
    maximizes E{A{PopX}/E{PopX}} for all temporal
    operators op and all events X
   or
    the pattern P such that LENGTH(P) < MAXL which
    maximizes E{A{XopP}/E{XopP}} for all temporal
    operators op and all events X
   Scan for patterns for which P is the prefix or suffix
   Insert_Patterns
    append newly found patterns to list of already found
    patterns
   Update_Smaller
    (discussed below)
   Check_Larger
    (discussed below)
END WHILE
Return interesting patterns
```


The algorithm continues until there are no more patterns for which (actual remaining/expected remaining) exceeds some minimum threshold MIN_TO_EXPAND, a parameter chosen at the outset.

Update_Smaller: Consider the situation when a pattern $\alpha\mathbf{N}\beta$ is chosen as the next candidate. During the scan, patterns of the form $\alpha\mathbf{N}\beta\mathbf{N}\gamma$, $\alpha\mathbf{N}\beta\mathbf{B}_K\gamma, etc\ldots$ will be discovered. If, for example, M occurrences of $\alpha\mathbf{N}\beta\mathbf{N}\gamma$ are found then there are M fewer patterns for which $\beta\mathbf{N}\gamma$ is a suffix remaining to be found. This decreases the chance that $\beta\mathbf{N}\gamma$ will be chosen as a candidate and this change needs to be recorded.

Likewise, during the scan for $\alpha\mathbf{N}\beta$, $\delta\mathbf{N}\alpha\mathbf{N}\beta$ may be found to occur L times. Therefore, the number of remaining patterns of the form $\delta\mathbf{N}\alpha\mathbf{N}X$ to be found has decreased by L. Again this needs to be recorded.

Check_Larger: Consider the situation when α is chosen as the next candidate. During the scan we will discover some number of $\alpha\mathbf{N}\beta$, say P. As was discussed earlier, this implies that there are P patterns of the form $\alpha\mathbf{N}\beta\mathbf{N}X$ and patterns of the form $X\mathbf{N}\alpha\mathbf{N}\beta$(we can make similar statements for the other operators). Some of these may have already been discovered, however. For example, if γ was already chosen as a candidate, and some $\beta\mathbf{N}\gamma$ were found, and then $\beta\mathbf{N}\gamma$ was chosen and M ocurences of $\alpha\mathbf{N}\beta\mathbf{N}\gamma$ were found, then the number of remaining patterns of the form $\alpha\mathbf{N}\beta\mathbf{N}X$ yet to be found is not P but rather $P - M$. This again needs to be recorded.

5 Experiments

We conducted experiments on three different problem domains. The first was a simple sequence of independent events. This data was generated synthetically. The second domain we considered were sequences of UNIX operating system calls as part of the `sendmail` program. The third was that of Web logfiles. In the last case, events were dependent.

5.1 Sequential independent events

We used an input string of length 1000 over 26 different events. In this case, we assumed that each event was equally likely and that the events were independent. We searched for patterns, P, for which Length(P)$\leq$ 5. Our results are presented in Table 2. The columns of the above table are as follows:
Algorithm - The algorithm used. The naive algorithm, presented in Section 4.1, represents essentially an exhaustive search over the input string and is guaranteed to find all interesting patterns. It is included as a benchmark by which we measure the effectiveness of the main algorithm. Percentage is equal to the value for the main algorithm divided by the value for the naive algorithm times 100 for each column respectively. The first number following each algorithm(2 or4) is the value of BEFOREK used. The second number(3,4, or 6) is the interestingness threshold.

Algorithm	# of Scans	# of Expanded Patterns	# of Interesting Patterns
Naive(2,3)	416	2489	290
Main(2,3)	161	919	268
Percentage	38.7%	36.9%	92.4%
Naive(4,3)	416	3105	259
Main(4,3)	163	1073	250
Percentage	39.2%	34.6%	96.5%
Naive(2,4)	416	2489	168
Main(2,4)	161	919	164
Percentage	38.7%	36.9%	97.6%
Naive(4,4)	416	3105	171
Main(4,4)	163	1073	166
Percentage	39.2%	34.6%	97.1%
Naive(2,6)	416	2489	133
Main(2,6)	161	919	130
Percentage	38.7%	36.9%	97.7%
Naive(4,6)	416	3105	129
Main(4,6)	163	1073	127
Percentage	39.2%	34.6%	98.4%

Table 2. Results for independent sequential data.

# of Scans - The number of scans over the input sequence necessary to discover all interesting patterns found.
# of Expanded Patterns - The number of patterns discovered, interesting or otherwise.
# of Interesting Patterns - The number of interesting patterns found.

Based on the results presented in Table 2, the main algorithm did not find all interesting patterns, although it discovered most while doing less work than the naive algorithm. Also note that the main algorithm was more accurate as our threshold for interestingness increased. In other words, when our algorithm did miss interesting patterns they tended not to be the most interesting.

Sequences of OS System Calls

econd domain we investigated was a sequence of operating system calls by a sendmail program. The events consisted of the 31 different system hat the program made and our string consisted of 31769 sequential calls. time of these experiments we had no knowledge of the actual probabil- f these events. Therefore, we made an assumption that system calls are ndent from each other and estimated probabilities of individual events by scanning the string and counting the number of actual occurrences of each For each event e_i we let $Pr[e_i]$ = (number of occurrences of e_i)/ (the total length). Because of this, the interestingness of each of atomic event was by

definition exactly 1. This forced us to assign a value to MIN_TO_EXPAND that exceeds 1 or else the algorithm would not even begin. This resulted in more scans of the input string than were actually necessary to discover interesting patterns but nonetheless the improvement we achieved over the naive algorithm was consistent with our experiments in other domains (approximately three times). The following represent a selection of interesting patterns discovered. These were selected because of a combination of their interestingness as well as our confidence that these actually represent significant events due to the number of occurrences of them. These results were generated on a run where we allowed strings of up to length 5.

```
EVENT :((sigblock NEXT setpgrp) NEXT vtrace)
COUNT :2032
ACT/EXP :43.1628

EVENT :(((sigblock NEXT setpgrp) NEXT vtrace) NEXT vtrace)
COUNT :455
ACT/EXP :83.1628

EVENT :(((sigblock NEXT setpgrp) NEXT vtrace) BEFORE sigvec)
COUNT :355
ACT/EXP :52.1150

EVENT :(sigblock NEXT(setpgrp BEFOREK vtrace))
COUNT :2032
ACT/EXP :21.5814

EVENT :((sigblock BEFOREK setpgrp) NEXT vtrace)
COUNT :2032
ACT/EXP :21.5814

EVENT :((sigpause NEXT vtrace) NEXT lseek)
COUNT :1016
ACT/EXP :106.672

EVENT :(sigpause BEFOREK (vtrace NEXT lseek))
COUNT :1016
ACT/EXP :53.336

EVENT :(sigvec BEFOREK(sigpause NEXT(vtrace
        NEXT(lseek NEXT lseek))))
COUNT:29
ACT/EXP :212.349

EVENT :(sigpause BEFOREK (vtrace BEFOREK lseek))
COUNT :2032
ACT/EXP :53.336
```

```
EVENT :((vtrace NEXT lseek) NEXT lseek)
COUNT :1017
ACT/EXP :35.5112
```

In these results COUNT represents the number occurrences of the pattern EVENT and ACT/EXP represents the interestingness of this pattern. Notice a couple of things. First, most of the interesting patterns that occurred a reasonable number of times (the ones shown above) were mostly of length 3. There were, of course, more interesting patterns of longer length but the number of occurrences of these patterns was significantly fewer. Also notice that no interesting UNTIL patterns were discovered. This is because we never saw AAAAAAB, i.e. all the occurrences of until were of the form AB or AAB which were captured by NEXT or BEFORE and since fewer instances of NEXT and BEFORE were expected these proved more interesting.

These system calls are from the UNIX operating system. In the future what we propose is to assign probabilities of atomic events based on their frequencies in a period when we are confident no intrusions to the network occurred and then see if we can discover interesting patterns that correspond to intrusions.

5.3 Web logfiles

Each time a user accesses a Web site, the server on the Web site automatically adds entries to files called *logfiles*. These therefore summarize the activity on the Web site and contain useful information about every Web page accessed at the site. While the exact nature of the information captured depends on the Web server that the site uses, the only information we made use of was the user identity and the sequence of requests for pages made by each user. The Web site we considered was that of one of the schools at a major university. The users we considered were the two most frequent *individual* users. It is important to recognize that the Web logfiles simply tell us the hostname from which a request originated. Typically, there are a large number of users who may access a Web site from the same host, and the hostname, therefore, cannot be used to definitively identify individual users. We attempted to identify, with some confidence, frequent hostnames that did indeed represent individual users. We used two Web logfiles for our experiments. First, we considered a synthetic Web This included a Web site with 26 different pages and 236 total links. We an input string of length 1000 representing 1000 hits on pages of the site. his case events were hits on Web pages. Probabilities were, of course, not pendent. The probability of a user reaching a given Web page is dependent he page he is currently at. In order to compute a priori probabilities of each we declared several pages to be equally likely "entrance points", to the Web If there were N "entrance points" then each has a $\frac{1}{N}$ chance of occurring. is one of these "entrance points", P has K links on it and one of these links page G then the probability of G occurring is $(\frac{1}{N})(\frac{1}{K})$. By conducting an ustive breadth-first search we were able to calculate the probabilities of each

event occurring (i.e. each page being "hit"). When calculating expectations for various patterns, we used conditional probabilities. So, for example, the $\mathsf{E}[\![A\mathbf{N}B]\!]$ is no longer $\mathsf{Pr}[\![A]\!]\mathsf{Pr}[\![B]\!](K-1)$, where K is the length of the input string. It is now $\mathsf{Pr}[\![A]\!]\mathsf{Pr}[\![B|A]\!](K-1) = \mathsf{Pr}[\![A]\!](1/\#\text{of links in page } A)(K-1)$ if there is a link from A to B and 0 otherwise. Our results for this data are presented

Algorithm	# of Scans	# of Expanded Patterns	# of Interesting Patterns
Naive2	634	1356	464
Main2	239	528	437
Percentage	37.7%	38.9%	94.2%
Naive4	654	1564	462
Main4	245	568	437
Percentage	37.5%	36.3%	94.6%

Table 3. Results for synthetic Web logfile data.

in Table 3. The interestingness threshold for these experiments was 3.0. Once again our algorithm was able to find most interesting patterns while examining much less of the search space than the naive algorithm did.

Finally, we considered data from an actual Website from one of the schools of a major university. There were 4459 different pages on this site with 37954 different links between pages. We used Web log data collected over a period of nine months and selected out the two most frequent individual users of the site, both of whom accounted for more that 1400 hits and used these sequences of hits as our input string. Our experiments using this data were less enlightening than when we used synthetic data. The main algorithm found only a handful of interesting patterns of length greater than two. In fact, when we applied the naive algorithm we found that there were few more interesting patterns to be found at all. More specifically, the main algorithm found 2 and 3 interesting patterns of length greater than two in our two input strings, respectively. The naive algorithm found 3 and 3. The primary reason for the lack of interesting patterns of greater length was that the size of the Web site dominated the size of the input string. The fact that there were 4459 pages and our input strings were only of length 1400 made the expected number of occurrences of each event very small. So small, in fact, that even a single occurrence of many events proved interesting.

Additional factors that compounded the problem are:

1. **Changing Web Structure.** The Web architecture from which we built the graph that the algorithm was run on was from a single moment in time(we captured the structure of the Web site, including the links, on a single day, and extrapolated it to 9 months of Web log data). Over this period there were some changes to the Web site. This creates some difficulties in that the Web logfiles showed that users linked from pages to other

pages where links did not exist in the Web we were considering. In fact, there were visits to pages in the Web log data that did not exist in the site we were using. This had the effect of forcing the expected number of occurrences of any patterns that included these pages or links to be zero and thus never considered interesting either as patterns or candidates.

2. **Multiple Sessions.** While each input string we used had a length greater than 1400 events, these Web hits spanned many sessions. In fact, the average session length was approximately 10 hits. The last hit from one session immediately preceded the first hit of the NEXT session in our input string. Normally, however, a link did not exist from the last page of the first session to the first page of the NEXT session. Therefore, once again this had the effect of forcing the expected number of occurrences of any patterns that included this sequence of pages to be zero and thus never considered interesting either as patterns or candidates.

3. **Caching.** Consider what sequence of hits appears in Web log data if a user goes to pages A, B, C, D in the following order $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D$. Normally, what occurs is that a request is made(and therefore logged) for page A then page B then page C then, however, when the user goes back to page B no request is made of the server because this page has been cached on the users' local machine. Finally, a request for page D will be made and logged. Therefore, this sequence of hits will appear in the Web log data as follows: $A \rightarrow B \rightarrow C \rightarrow D$. If no link exists from page C to page D then once again the expected number of occurrences of any pattern including this sequence of events will be zero. Given the wide use by Web users of the BACK button, the effect of caching is substantial.

4. **Local Files.** Finally, many pages that appeared in the Web log data did not appear in the Web site we were using because they were files kept on individuals local machines in their own directories, rather than on the Web server. These pages had the same effect as the changes made in the Web over the nine month period.

sons Learned. The primary cause of our lack of success in finding interest- patterns in the use of our university Web site was the fact that the size of the was very large in comparison to the size of the input strings we considered. are planning to obtain Web logfiles spanning a longer period of time, for ıaller and more stable Web site. We are also considering various models to with the loss of patterns that we experienced due to the multitude of user ions.

6 Conclusions and Future Work

In this chapter, we reviewed the characterization of different knowledge discovery tasks in temporal databases (as summarized in Table 1) and focused on a Class III problem of generating unexpected predicates. In particular, we presented an algorithm for finding unexpected patterns, expressed in temporal logic, in sequential databases. We used multiple scans through the database and the step-by-step expansion of the most "promising" patterns in the discovery process. To evaluate the performance of the algorithm, we compare it with the "naive" algorithm that exhaustively discovers all the patterns and show by how much our algorithm outperforms it. We also use our algorithm for discovering interesting patterns in sequences of operating system calls and in Web logfiles.

In its current implementation, our algorithm discovers temporal patterns only of a certain type (described in Section 3). As a future work, we plan to extend our algorithm to include more complex temporal logic expressions. We also plan to extend our methods to discovering unexpected patterns in temporal databases, where the patterns will be expressed in first-order temporal logic. Finally, we plan to apply our algorithm to the problem of intrusion detection, as well as to a more suitable Web site having fewer HTML files and more traffic. We expect to find more interesting patterns for such a site.

We are also interested in pursuing some of the complexity issues that arose in the NP-hardness proof. Specifically, The problem CLIQUE that we reduced from is actually SNP-complete [19] which is a class of languages that has some interesting properties we are investigating. In addition, we are considering application of Ramsey's Theorem to our problem domain.

7 Acknowledgments

The authors wish to thank Bud Mishra from the Computer Science Department at NYU for his general input to the contents of this chapter as well as specifically for his help with the proof of Theorem 1.

A Appendix: Proof of Theorem 1

Problem: Given a string of temporal events $V = v_1, v_2, \ldots, v_r$, does there exist an interesting pattern in V of the form $X_1 \mathbf{B_k} X_2 \mathbf{B_k} \ldots \mathbf{B_k} X_m$ for an arbitrary m? The following proof shows that this problem is NP-complete.

Proof of Theorem 1: We show that our problem is NP-hard by proving that CLIQUE $\leq_p$ INTERESTINGNESS. The reduction algorithm begins with an instance of CLIQUE. Let $G = (V, E)$ be an arbitrary graph with $|V|$ vertices and $|E|$ edges. We shall construct a string of events S such that an interesting pattern of the form $e_1 \mathbf{B_k} e_2 \ldots \mathbf{B_k} e_m$ exists if and only if G has a clique of size m. The string is constructed as follows. Each vertex $v_1, v_2, \ldots, v_{|V|}$, in the graph G will become an event in our string S, i.e. our events are $e_1, e_2, \ldots, e_{|V|}$. Additionally we will make use of $(|V| + |E|)m$ "dummy" events called $d_1, d_2, \ldots, d_{(|V|+|E|)m}$,

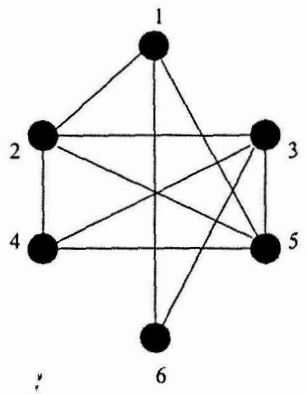

Fig. 2. The graph $G(V, E)$ with vertices $v_1, v_2, \ldots, v_6$ and a clique C of size 4. $C = \{v_2, v_3, v_4, v_5\}$

where m is the value from the CLIQUE problem. Based on each vertex $v_i \in G$ a substring will be created. The associated event e_i will be called the "generator" of this substring and the substring will be "generated" by the event. The concatenation of these substrings will be the string S. Initially, the vertices in G are arbitrarily ordered $1, 2, ...|V|$. Then for each associated event e_i, in order, we create the substring based on e_i by listing, again in sorted order, the list of vertices(actually their associated events) e_j, for which there exists an edge $(v_i, v_j) \in E$ plus the event $e_i|V|$ times. For example, the substring generated by e_2 for the graph in Figure 1 would be

$$e_1 \underbrace{e_2 e_2 \ldots e_2}_{|V|} e_3 e_4 e_5$$

since there are edges in G from v_2 to each of e_3, e_4, and e_5. Following each substring generated in this fashion we concatenate a substring of all the dummy events in sorted order. As will be seen shortly these dummy events are used to separate the substrings of events e_i and therefore no dummies are needed following the substring generated by $e_{|V|}$. Thus, for the above graph the string

$$S = \underbrace{e_1 e_1 \ldots e_1}_{|V|} e_2 e_5 e_6 d_1 \ldots d_{(|V|+|E|)m} e_1 \underbrace{e_2 \ldots e_2}_{|V|} e_3 e_4 e_5 d_1 \ldots$$
$$d_{(|V|+|E|)m} e_2 \underbrace{e_3 \ldots e_3}_{|V|} e_4 e_5 e_6 d_1 \ldots d_{(|V|+|E|)m} e_2 e_3 \underbrace{e_4 \ldots e_4}_{|V|} e_5 d_1 \ldots$$
$$d_{(|V|+|E|)m} e_1 e_2 e_3 e_4 \underbrace{e_5 \ldots e_5}_{|V|} d_1 \ldots d_{(|V|+|E|)m} e_1 e_3 \underbrace{e_6 \ldots e_6}_{|V|}$$

:otal length of S will be $2|E|+|V|^2+(|V|-1)((|V|+|E|)m)$. This can be seen lows. The substring generated by e_i will have $|V|$ occurrences of e_i plus one

occurrence of each event e_j such that $(v_i, v_j) \in E$ $(deg(v_i))$. Summing over all vertices i the total length of these substrings will equal $2|E| + |V|^2$. In addition there will be a total of $|V| - 1$ occurrences of the substring $d_1 d_2 \ldots d_{(|E|+|V|)m}$ with a total length of $(|V| - 1)((|V| + |E|)m)$. The string S can clearly be constructed in polynomial time as it is polynomial in the size of the graph.

Given that our problem allows for an exogenous assignment of probabilities we will assume that all of the events are equiprobable. That is

$$\mathsf{Pr}[\![X]\!] = \frac{1}{|V| + (|V| + |E|)m}$$

for $X = e_i$ or d_j, $i \in 1 \ldots |V|, j \in 1 \ldots (|e| + |V|)m$. Since each dummy event occurs exactly $|V| - 1$ times and each event e_i occurs $|V|$ times in the substring it generates plus an additional $deg(|V|)$ times elsewhere, these exogenous probabilities are not consistent with the actual probabilities of the events in S as the events corresponding to vertices occur more frequently than the dummy events. It is possible to define the probabilities so that the assigned probabilities of the dummy events is consistent with their actual frequencies but this requires a somewhat more complicated construction and proof and offers little insight into the problem so we have chosen to proceed as described above.
Let BEFOREK $= |V| + |E|$.

The expected number of occurrences of a pattern

$$\begin{aligned} X_1 \mathbf{B_k} X_2 \mathbf{B_k} \ldots \mathbf{B_k} X_L &= (n - K(L-1))K^{L-1}\mathsf{Pr}[\![X_1]\!]PX_2 \ldots \mathsf{Pr}[\![X_L]\!] \\ &+ \sum_{i=L-1}^{K(L-1)-1} \binom{i}{(L-1)} \mathsf{Pr}[\![X_1]\!] \ldots \mathsf{Pr}[\![X_L]\!], \quad if K > 1 \\ &= (n - K(L-1))K^{L-1}\mathsf{Pr}[\![X_1]\!]PX_2 \ldots \mathsf{Pr}[\![X_L]\!], \quad else \end{aligned}$$

where $K =$ BEFOREK and $n = |S|$. This can be derived in a manner analogous to how expectations were derived in section 3. It can be seen that in the special case of $L = 2$ this formula reduces to the one derived previously for $\mathsf{E}[\![\mathbf{B_k}]\!]$.

For the case where $K = |V| + |E|, n = 2|E| + |V|^2 + (|V| - 1)((|V| + |E|)m)$, and $L = m$ we will call the value of this expectation ϵ. Let the interestingness threshold

$$T = \frac{2|V|m - 1}{2\epsilon}$$

The relevance of this value is that if a pattern of the form $X_1 \mathbf{B_k} X_2 \ldots \mathbf{B_k} X_m$ is instantiated only with events e_i(no dummies) and it occurs at least $|V|m$ times it will be deemed interesting. If it occurs $|V|m - 1$ times it will not. This will be discussed in further detail shortly.

We must now show that this transformation from CLIQUE to INTERESTINGNESS is a reduction. First, suppose a CLIQUE $v_1, v_2, \ldots, v_m$ exists in G and therefore corresponding events $e_1, e_2, \ldots, e_m$ exist in S. Note that here the indexes of the vertices and events are not intended to suggest that the clique

must consist of the first m vertices in the original ordering but rather are used for ease of exposition. Of course these $v_1, \ldots, v_m$(and $e_1, \ldots, e_m$) could represent any collection of m vertices(events) although we will continue to assume that they are in sorted order. By construction, the substring generated by e_1 will include

$$\underbrace{e_1 e_1 \ldots e_1}_{|V|} e_2 e_3 \ldots e_m$$

For an arbitrary i the substring generated by e_i will include [9]

$$e_1 e_2 \ldots \underbrace{e_i e_i \ldots e_i}_{|V|} e_{i+1} \ldots e_m$$

Each substring will contain $|V|$ occurrences of the pattern $e_1 \mathbf{B_k} e_2 \mathbf{B_k} e_3 \mathbf{B_k} \ldots \mathbf{B_k} e_m$ and there are m such substrings so the total number of occurrences of this pattern is $|V|m$. Thus

$$\frac{\mathsf{A}[\![e_1 \mathbf{B_k} \ldots \mathbf{B_k} e_m]\!]}{\mathsf{E}[\![e_1 \mathbf{B_k} e_2 \ldots \mathbf{B_k} e_m]\!]} = \frac{|V|m}{\epsilon} > T$$

Conversely, suppose that an interesting pattern of the form $X_1 \mathbf{B_k} X_2 \ldots \mathbf{B_k} X_m$ exists. We must show that a corresponding CLIQUE of at least size m exists in G. The following lemma is the basis for our showing this.

Lemma 1. *If an interesting pattern exists then it consists only of events e_i, containing no dummy events.*

Proof: We have already seen that if a CLIQUE of size m exists in G then an interesting pattern exists in S. Thus interesting patterns are possible. What is left to show is that if

- a pattern consists only of dummy events then it can't be interesting, and
- if a pattern consists of both dummy events and events e_i it can't be interesting

Assume we instantiate the pattern $P = X_1 \mathbf{B_k} \ldots \mathbf{B_k} X_m$ with j dummy events and $m - j$ events e_i where $j = 1 \ldots m$. Note that given our definition of BEFOREK for any pattern of this form its total length, i.e. the distance in the ... S from X_1 to X_m can be at most $(|E| + |V|)m$. Therefore, if a pattern ins any dummy events these occurrences must occur only at the beginning l of the pattern since any dummy event is part of a substring of $(|E|+|V|)m$ ıy events. That is there cannot exist a dummy event d_j in the pattern such ın event e_i occurs before d_j in the pattern *and* an event e_k occurs after it. ill assume, without loss of generality, that the j dummy events all occur at d of the pattern. We will next count the maximum number of occurrences terns of this form.

[9] re may, of course be vertices that are not part of the clique that are connected some edge to e_i. These vertices would also be included.

Each of the $m - j$ events e_i generates a substring in S. In that substring the event e_i occurs $|V|$ times and all other events occur once. In addition, in the substring of dummy events immediately following this substring each event occurs once. Thus, there can be at most $|V|$ occurrences of the pattern P that include events from the substring generated for each e_i. There are a total of $m-k$ events e_i in the pattern and therefore a maximum of $(m - j)|V|$ occurrences of P that include these substrings. In addition, there exist $|V| - (m - j)$ substrings generated by events not in P. In each of these substrings P can occur at most once since each event in P occurs at most once in a substring that it did not generate. This can result in a maximum of $|V| - (m - j)$ additional instances of P for a total of $(m - j + 1)|V| - (m - j)$ occurrences of P. This expression is maximized if $j = 1$ in which case the maximum number of occurrences of $P = m|V| - m + 1$. Since

$$\frac{m|V| - m + 1}{\epsilon} < T$$

where ϵ is again the expected number of occurrences of this pattern,this pattern cannot be interesting. □

We now know that any interesting pattern can consist only of events e_i. We also know that each occurrence of an interesting pattern can include only events generated by a single e_i (since BEFOREK $< (|E| + |V|)m$, the length of the dummy substrings separating event substrings generated by each event). Furthermore, we can use an argument identical to the one used in the proof of the above lemma to show that for at least $m|V|$ occurrences of a pattern to exist at least $m|V|$ of them must include the generating event from which all the events in this instance came. In other words, if an interesting pattern $e_1 \mathbf{B_k} \ldots \mathbf{B_k} e_m$ exists then there must be at least $m|V|$ instances which include the e_i that generated the substring from which all the other events came. To see this note that each time an instance of a pattern that includes a generating event occurs, $|V|$ instances will actually occur, one for each copy of the generating event in the substring it generated. Let us assume that only $(m - 1)|V|$ instances of a pattern exist that include the generating event from which all other events in this instance came.[10] In all the other substrings generated by events not in the pattern there can be at most one instance of the pattern since each event occurs at most once in a substring it did not generate. There are $|V| - (m - 1)$ such events so the total number of instances would only be $m|V| - m + 1$. Therefore, for a pattern to occur at least $m|V|$ times and thus to be interesting there must be $m|V|$ instances that include the generator of the other events in that instance. Since each generator results in $|V|$ instances there are m generators that are part of instances. The m vertices that correspond to these m events form a clique in G. This is clearly true since for any of the e_i amongst these m generators there is an edge from itself to each of the other generators.

Finally, note that this problem is also in NP and therefore NP-complete since given a certificate(i.e. an instantiation of our pattern in this case) we can

[10] There cannot be any more than this unless there are $m|V|$ since they come in multiples of $|V|$.

check if it is interesting by simply scanning over S. This clearly can be done in polynomial time. □

Notice that we have phrased our NP-hardness problem as "Does *any* interesting pattern exist?" We could have just as easily posed the question "Do p interesting patterns exist"? Our proof can be trivially extended to accomplish this by enforcing that the dummy events always contain $p-1$ interesting patterns and that the pth interesting pattern only occur if a clique of size m exists in G. Our decision to enforce that the dummy events contain no interesting patterns and to thus pose our question as we did was rather arbitrary.

References

1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *In Proc. of the conference on foundations of data organizations and algorithms (FODO)*, October 1993.
2. R. Agrawal, T. Imielinsky, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference*, pages 207–216, 1993.
3. R. Agrawal, K-I Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *In Proc. of the 21st VLDB conference.*, 1995.
4. R. Agrawal, G. Psaila, E. Wimmers, and M. Zait. Querying shapes of histories. In *In Proc. of the 21st VLDB conference.*, 1995.
5. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the International Conference on Data Engineering.*, March 1995.
6. W.A. Ainsworth. *Speech recognition by machine.* Peter Peregrinus Ltd., London, 1998.
7. D. Berndt. AX: Searching for database regularities using concept networks. In *Proceedings of the WITS Conference.*, 1995.
8. D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* AAAI Press/ The MIT Press, 1996.
9. C. Bettini, X.S. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proceedings of PODS Symposium*, 1996.
10. J. Clifford, V. Dhar, and A. Tuzhilin. Knowledge discovery from databases: The NYU project. Technical Report IS-95-12, Stern School of Business, New York University, December 1995.

V. Dhar and A. Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6), 1993.

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *In Proceedings of the SIGMOD conference.*, 1994.

D.Q. Goldin and P.C. Kanellakis. On similarity queries for time-series data: constraint specification and implementation. In *In Proc. of the 1st Int'l Conference on the Principles and Practice of Constraint Programming. LNCS 976*, September 1995.

P. Laird. Identifying and using patterns in sequential data. In *Algorithmic Learning Theory, 4th International Workshop*, Berlin, 1993.

15. J.B. Little and L. Rhodes. *Understanding Wall Street.* Liberty Publishing Company, Cockeysville, Maryland, 1978.
16. H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, August 1996.
17. H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal,Canada, August 1995.
18. B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: A temporal logic approach. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, August 1996.
19. C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.
20. H.V. Poor. *An Introduction to signal detection and estimation.* Springer-Verlag, New York, 1988.
21. L.R. Rabiner and S.E. Levinson. *Isolated and connected word recognition: Theory and selected applications. In Readings in speech recognition.* Morgan Kaufmann Publishers, San Mateo, CA., 1990.
22. P. Seshadri, M. Livny, and R. Ramakrishnan. Design and implementation of sequence database system. In *Proceedings of ACM SIGMOD Conference*, 1996.
23. A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, August 1995.
24. A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), December 1996.
25. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the International Conference on Extending Database Technology*, 1996.
26. J. van Leeuwen. *Handbook of Theoretical Computer Science: Volume B Formal Models and Semantics.* The MIT Press/Elsevier, 1990.
27. J. T.-L. Wang, G.-W. Chirn, T. G. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1994.
28. L. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. In *Fuzzy Sets and Systems, vol. 11*, pages 199–227. 1983.