DEFINING NETWORK ACTIVITY PATTERNS USING FIRST ORDER TEMPORAL LOGICS

Lubomir Stanchev

Abstract: Part of network management is collecting information about the activities that go on around a distributed system and analyzing it in real time, at a deferred moment, or both. The reason such information may be stored in log files and analyzed later is to data-mine it so that interesting, unusual, or abnormal patterns can be discovered. In this paper we propose defining patterns in network activity logs using a dialect of First Order Temporal Logics (FOTL), called First Order Temporal Logic with Duration Constrains (FOTLDC). This logic is powerful enough to describe most network activity patterns because it can handle both causal *and* temporal *correlations. Existing results for data-mining patterns with similar structure give us the confidence that discovering DFOTL patterns in network activity logs can be done efficiently.*

Keywords: network management, temporal logics

ACM Classification Keywords: C.2.3 Network Operations – network management; F.4.1 Mathematical Logic – temporal logic

1. Introduction

The rapid growth of the Internet has increased the need for recording and analyzing network activity patterns. The vast amount of information that is collected makes its manual processing unfeasible. Fortunately, data-mining algorithms have been developed for the automatic and semi-automatic processing of big chunks of data. In this paper we explore how useful information can be extracted from network activity logs in an efficient way through the use of semi-automatic data exploration procedures.

Network activity logs are important for network management. Related research (e.g., [LeFa93]) shows that log analyzing activities can be helpful for all five categories of network management. For example, fault management network logs can be analyzed to derive the patterns that correlate alarms with actual faults. Such patterns can then be used by the network management station to feed to the network administrator the faults that actually occurred, rather then their symptoms. In configuration management, the network log can be used to deduce what system components are active, how they are configured, and how they are connected. In security management, historical logs can be analyzed in order to discover patterns of irregularities that can point to a hacker attempt to break in the system or to an actual breach of security. In performance management, patterns that are data-mined from log files can show the network administrators which system components are under-utilized and which are over-utilized so that appropriate adjustments can be made. Finally, in accounting management, patterns extracted from network logs can help the system administrators get a better understanding of which groups of users rely on which network resources. Serious abuse of the system can also be detected from analyzing network logs.

There are two known types of procedures for acquiring network activity information. The first method presumes that the network management station has traps set at different locations throughout the system. An example of a trap for a network link is: "If the traffic over the link reaches 90% of the link capacity, then send an alarm". An example of a practical implementation of this approach is Remote Monitoring (RMON) with Simpler Network Management Protocol (SNMP) (see [GaTa99]). The second approach assumes that the network management station will periodically poll managed objects or network monitoring components. An example of a system which uses both techniques is Network Flight Recorders (NFR) - see [WeChSt99].

The first contribution of this paper is the introduction of a dialect of First Order Temporal Logics (FOTL) that we call *First Order Temporal Logics with Duration Constraints (FOTLDC)*. This dialect can be used to express patterns in network activity logs. FOTLDC is more expressive than many existing languages for network activity pattern definition (e.g., [HaSuVi99], [MaToVe95],[LePaSt99], and [LePaSt99]). Unlike the enumerated languages, FOTLDC can be used to express not only *causal*, but also *temporal* patterns. It also allows for the expression of patterns that reference attribute values of managed objects over time periods, which is not supported even in rich temporal-based models, such as the one described in [LiMoYa99]. To summarize, FTOLDC can capture useful patterns in network activity logs that cannot be captured by existing proposals.

The second contribution of this paper is the architectural design of a system that analyzes network activity logs based on FOTLDC patterns. The most challenging part of implementing such a system is detecting which patterns are normal and which are abnormal. Part of this information can be provided by experts and inferences from this expert knowledge can also be used. However, a good log-analyzer system should be able to classify on its own the patterns as normal and abnormal. This can be done by observing the frequency of the different events in a network activity log and classifying the more frequently occurring patterns as normal and the less frequently occurring patterns as abnormal. The actual thresholds can be set by a domain expert. Note that, in order for this approach to work, the data in the network activity log must be representative. For example, if the data that is analyzed is taken during a hacker's attack, then the data-mining engine may deduce that patterns of unauthorized network intrusion are normal.

To summarize, we propose the creation of a process for analyzing network activity logs based on discovering FOTLDC patterns. The novelties in our paper are:

- extending first order temporal logics with duration constraints,
- representing network activity patterns as temporal logic formulas, and
- presenting an architectural design of a system that analyzes network activity logs using the proposed temporal logic.

In what follows, in Section 2 we define the syntax and semantics of the FOTLDC. We also explore how this logic is different from existing temporal logics and why it is suitable for describing network log patterns. Next, in Section 3 we present existing data-mining approaches for discovering causal and temporal patterns and discus how they can be applied to FOTLDC patterns. In Section 4, we describe the network activity log analyzing process. Section 5 contains a brief summary of the paper and direction for future research.

2. First Order Temporal Logic with Duration Constraints

We next present the syntax of FOTLDC. This logic is an extension of the temporal logic proposed in [Chom95] with duration constraints. The elements of a formula in FOTLDC are:

- a finite set of predicate symbols p_i over Δ,
- the logical connectives $\wedge, \vee, \neg$, and $\Rightarrow$,
- a countable infinite set of variables V over Δ,
- a finite set of constant symbols over Δ,
- the operations +,-,*,/,
- the equality and inequality symbols: =, ≠, >, <, ≥, and ≤,
- the quantifiers $\forall$ and $\exists$, and
- the past and future temporal connectives $\blacklozenge_k$, $\lozenge_k$, $\blacksquare_k$, and $\square_k$, where $k \in T \cup \{\infty\}$.

In above definition, T is the time domain, which we assume to be isomorphic to the set of natural numbers, and Δ is the domain of discourse. The semantic of a FOTLDC formula is defined relative to a time instance t and a finite-time temporal database $D = (D_0, D_1, ..., D_k)$. Such a database represents the different instances of a database trough time, where D_t denotes the database state at time instance t (for a formal definition of a temporal database see [ChDa98]). A valuation v that interprets a FOTLDC formula is defined as a mapping from V to Δ, where V is the set of variables. Note that we assume that value of a variable does not change with time. Informally, the meaning of the newly introduced connective $\blacklozenge_k$ ($\lozenge_k$) is: in at least one point of time in the past (future) k time units the formula that follows was (will be) true. The definition is relative to the current time moment t. Similarly, the connective $\blacksquare_k$ ($\square_k$) means that always in the past k time units (always in the next k time units), relative to the current time moment t, the formula that follows was (will be) true. A formal definition follows.

$D, v, t \models \blacklozenge_k \varphi$ iff $\exists t' \in T$ such that $0 < t - t' \leq k$ and $D, v, t' \models \varphi$

$D, v, t \models \lozenge_k \varphi$ iff $\exists t' \in T$ such that $0 < t' - t \leq k$ and $D, v, t' \models \varphi$

$D, v, t \models \blacksquare_k \varphi$ iff $\forall t' \in T$ such that $0 < t - t' \leq k$, it is the case that $D, v, t' \models \varphi$

$D, v, t \models \square_k \varphi$ iff $\forall t' \in T$ such that $0 < t' - t \leq k$, it is the case that $D, v, t' \models \varphi$

In order to define patterns with FOTLDC, we need to represent the network activity logs as finite-time temporal databases. Fortunately, temporal databases are a natural way of representing data that is collected as a result of polling managed objects periodically. Data collected in the management station as a result of alarms, for example, SNMP traps, can also be represented in a temporal fashion.

We next present an example of a FOTLDC pattern for network activity. Suppose we want to define the pattern that if $host_1$ goes down and stays down for more than two minutes, then the utilization of $link_1$ will be reduced to bellow fifty percent in the next five minutes. This can be expressed using the following FOTLDC expression.

$\square_{120}$ [host_status($host_1$,off)]=>$\lozenge_{300}$ [($\exists Z$, $\exists W$) (link_throughput($link_1$, Z, W) $\wedge$ (z/w<0.5))]

Note that time is represented with granularity of one second. The predicates host_status corresponds to the temporal relations host_status, which holds exactly when the host specified by the first argument has the state specified by the second argument. The predicate link_throughput correspond to the temporal relations link_throughput, which holds exactly when the link specified by the first attribute has the link usage specified by the second attribute and the capacity specified by the third attribute. In order to determine if the example FOTLDC expression is true, one has to check that the FOTLDC formula holds for any time instance t for which there exists a log entry.

3. Data Mining Temporal Data

The activity logs for a distributed system contain huge amount of data about the different activities over time. Useful information from such logs can be extracted, for example, using data mining techniques such at the ones described in [AgImSw93]. Note however that the goal of rule extraction is to extract rules of the form: X->Y, where X and Y are first order formulas over a database. For example, when applied to a supermarket database, a data-mining algorithm may discover that people who purchase dippers also purchase beer in eighty percent of the cases. In data-mining literature, this kind of dependency is usually referred to as a *casual* dependency. Although causal dependencies are important in analyzing network activity logs, they are not powerful enough to describe all interesting correlations between events. For example, an interesting pattern that one can hope to discover in a network log for the purpose of fault management is the occurrence of several events in a particular order in a short span of time. In order to describe such a pattern, one needs to use *temporal* dependencies.

Little research has been done in the area of data-mining temporal dependencies - [MaToVe95] and [Chom95] are two exceptions. The paper [MaToVe95] proposes an algorithm for data-mining *sequential* and *parallel* episodes from a sequence of events. A sequential episode specifies that a set of events occur in particular order in the same time window. A parallel episode specifies that a set of events occur within a time window in any order. The paper shows that any pattern of event occurrences can be broken down into a composition of sequential and serial episodes. Moreover, it shows an efficient algorithm that discovers frequent patterns that occur in windows of specified size that belong to a specified, closed with respect to the operation sub-episode, set of patterns.

The [Chom95] paper presents an efficient way for storing a temporal database. As well, it describes efficient algorithms for discovering and verifying FOTL patterns. It is our hope that the algorithms from the two papers can be combined to produce an efficient algorithm for data-mining FOTLDC formulas. Designing such an algorithm remains an area for future research.

4. The Network Activity Log Analyzing Process

Figure 1 shows the overall process of analyzing a network activities log file. First, the network management station collects information about the state of the network over time trough SNMP traps or by directly polling managed objects. This information is then passed to a temporal database and stored there. Parallel to this process, a domain expert enters data patterns in the pattern database that are known to be normal or that are know to be abnormal.

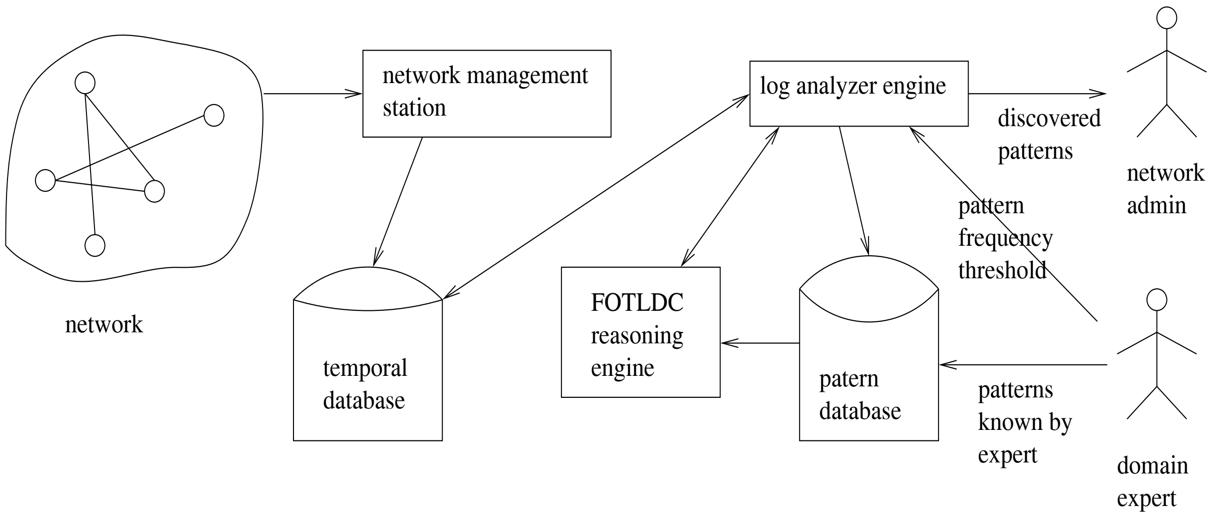

Figure 1. Network activities log analyzing process

The key part of the process is the log analyzer engine. It has two functions. The first function is to find if patterns that are known to be normal or abnormal (the FOTLDC reasoning engine can provide this information) can be discovered in the temporal database. Once a pattern is discovered, the corresponding data in the temporal database is marked as used so it is not analyzed further. As well, if the pattern that is found to be true for the temporal database is an abnormal one, then it is reported to the network administrator. The second function of the log analyzer is to discover new patterns. Once a pattern is found, the log analyzer engine checks with the FOTLDC reasoning engine to see if the pattern is indeed new, that is, the FOTLDC reasoning engine cannot classify it as normal or abnormal based on the information in the pattern database. If the pattern is indeed new, then it is classified as normal or abnormal based on its frequency. The pattern frequency threshold is set by an expert and it specifies how often a pattern has to occur in order to be classified as frequent (i.e., normal) or infrequent (i.e., abnormal). New patterns are found by data-mining the temporal database. Whenever new patterns are discovered, they are stored in the pattern database.

The FOTLDC reasoning engine is used by the log analyzer engine to determine which patterns are known to be normal and which to be abnormal. Since patterns are expressed as FOTLDC, it is also possible to reason with them. In other words, the pattern database may be thought as a knowledge database that knows which patterns are normal and which are abnormal. The FOTLDC reasoning engine can be thought as a theorem prover which can decide if there is enough information to classify a given pattern as normal or abnormal.

5. Summary and Future Research

In this paper we propose architecture for analyzing network activity logs using activity patterns expressed via FOTLDC. What remains to be done is the low-level design of the system. This includes:

- the design of an efficient data-mining algorithm that discovers FOTLDC patterns,
- the design of an algorithm that implements an efficient FOTLDC reasoning engine, and
- the design for the interface between the different components in Figure 1.

After these low-level designs have been performed, an actual implementation can be coded. Then, the components from Figure 1 can be connected to an existing network. The network activity log analyzing process can then be use in a test situation to collect experimental data. The data can then be analyzed in order to determine how successful the proposed system will be in a real-world situation.

Note that the proposed methodology for examining activity logs is more or less automatic - that is, it allows for little human interaction. However, experimental results have shown that such algorithms are not very successful. Therefore, it is worth exploring how human experts can participate in the different stages of pattern discovery. For example, when a new pattern is discovered, it can be examined by a domain expert before it is recorded in the pattern database.

To summarize, we propose a methodology for network activity log processing. We believe that the theoretical background behind the proposed methodology is sound and that when applied in practice the proposed network activity analyzer will produce results better than most existing network monitoring systems. However, the only way to find this for certain is to implement the proposed network activity analyzer and compare its performance and effectiveness to existing network monitoring systems.

Bibliography

[AgImSw93] Agrawal R., Imielinski T., Swami A., Mining association rules between sets of items in large databases. In *Proceeding of the ACM SIGMOD Conference on Management of Data*, pp. 207-216, 1993

[ChDa98] Chomicki, J., Toman D. Temporal Logic in Information Systems. In *Logics for Databases and Information Systems*, Kluwer Academic publishers, pp. 31-70, 1998

[Chom95] Chomicki, Jan. Efficient Checking of Temporal Integrity Constraints Using Bounded History Incoding. In *ACM Transactions on Database Systems*, Vol 20, No. 2, pp. 149-186, June 1995

[HaSuVi99] Hasan M., Sugla B., Viswanathan R. A Conceptual Framework for Network Management Event Correlation and Filtering Systems. In *Sixth IEEE/IFIP International Conference on Integrated Network Management*, May 1999

[LePaSt99] Lee W, Park C., Salvatore J., Automated intrusion detection using NFR: Methods and experiences. *Workshop on Intrusion Detection and Monitoring*, 1999

[LeFa93] Leinwand A, Fang K,. Network Management - A Practical Perspective. Addison Weley, 1993

[LiMoYa99] Liu G., Mok A.K., Yang E.J., Composite Events for Network Event Correlation, In *Sixth IFIP/IEEE International Symposium on Integrated Network Management*, May 1999

[MaToVe95] Mannila H., Toivonen H., Verkamo A., In *Proceedings of The First International Conference on Knowledge Discovery and Data Minning*, Montreal, Canada, pp. 20-21, 1995

Authors' Information

Lubomir Stanchev - Indiana University - Purdue University Fort Wayne, 2101 E. Coliseum Blvd., Fort Wayne, IN 46805 USA e-mail: stanchel@ipfw.edu

ON THE ERROR-FREE COMPUTATION OF FAST COSINE TRANSFORM

Vassil Dimitrov, Khan Wahid

Abstract: We extend our previous work into error-free representations of transform basis functions by presenting a novel error-free encoding scheme for the fast implementation of a Linzer-Feig Fast Cosine Transform (FCT) and its inverse. We discuss an 8x8 L-F scaled Discrete Cosine Transform where the architecture uses a new algebraic integer quantization of the 1-D radix-8 DCT that allows the separable computation of a 2-D DCT without any intermediate number representation conversions. The resulting architecture is very regular and reduces latency by 50% compared to a previous error-free design, with virtually the same hardware cost.

Keywords: DCT, Image Compression, Algebraic Integers, Error-Free Computation.

ACM Classification Keywords: I.4.2 Compression (Coding), I.1.2 Algorithms, F.2.1 Numerical Algorithms.