A Conceptual Framework for Network Management Event Correlation and Filtering Systems

Masum Hasan   *Binay Sugla*   *Ramesh Viswanathan*

Bell Laboratories, Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733
USA
{masumh,sugla,rv}@bell-labs.com

Abstract

Event correlation is a key functionality of a network management system that is used to determine the root cause of faults in a network, and to filter out redundant and spurious events. A number of event correlation systems have been proposed. The event correlation systems generally combine causal and temporal correlation models with the topology of a network. The power and robustness of the models used and the algorithms developed vary from system to system. However, in the absence of a simple, uniform, and precise presentation of the event-correlation problem, it is impossible to compare their relative power or even analyze them for their properties. In general, causal and temporal-based correlation models have not been rigorously presented or thoroughly investigated. In this paper we formalize the concepts of causal and temporal correlation using a single conceptual framework. We characterize various properties of the framework. We can characterize existing systems based on the formal properties of our framework, and we consider one system as an illustrative example.

Keywords

Event correlation and filtering, fault detection, root-cause analysis, correlation models, formal methods

1 Introduction

An important component of a network management (NM) system is an event correlation and filtering system (ECS). An ECS aims to: (a) diagnose root causes of network faults and performance degradations by establishing relationships between network events, (b) filter event (alarm) flood by correlating events into a single conceptual event. To be useful, an ECS must be (a) *correct*: the root causes inferred by an ECS should with high likelihood be entailed by the detected events, *i.e.*, the root causes must truly have occurred in the network, (b) *optimal*: the ECS should infer as small a set of root causes that can explain all the detected events.

Because of the importance of an ECS, a number of systems [9, 6, 10, 13, 1] have been proposed and implemented. A comparative study of the software frameworks used in these systems can be found in [7]. These systems vary widely in the underlying representation structures used to model the relationships between network events and consequently the algorithms used in their correlation engines. Often the models and algorithms are only implicitly given via the software frameworks that are used in the systems which themselves vary widely. For example, systems such as GTE IMPACT [9] and other similar ones [12, 5, 14, 2] are rule-based systems in which the network event relationships are specified using AI rule languages and the correlation engine is then defined by the execution of the rule engine on the specified set of rules. The CEDAR-HY+ system [6] uses a database-like language to specify event relationships which are then compiled to Petri net like transition nets which serve as the correlation engine. In the HP OpenView ECS [13], a circuit diagram and functional programming language is the specification framework and the correlation engine is an execution of this program specification. The SMARTS InCharge [10] system uses a CORBA-like specification framework but the correlation engine is obtained by a suitable encoding of the specification in terms of codebooks. Finally, the Seagate NerveCenter [1] requires programming the correlation engine directly as state transition diagrams and finite state machines. This wide diversity in the description and implementation of the systems makes it difficult to compare them. Moreover, in the absence of a simple definition of the event correlation problem that is independent of the software framework used, it is impossible to analyze any existing system or one proposed in the future for correctness or optimality.

In this paper, we aim to study the problem of event correlation from a *general* and *rigorous* point of view. We present general frameworks for describing causal and temporal relationships between events. For these frameworks, we define the correlation relationships that would be correct to infer from them. Finally, we can use these frameworks as a basis for understanding existing systems more precisely. We believe that our framework is compelling for a number of reasons:

- It is general enough to encompass the models used in existing systems. Due to space limitations, we consider only the SMARTS InCharge system in this paper but other systems can be understood using our framework as well.
- It is scalable in a uniform way to incorporate different notions of interest. For example, our temporal correlation framework is obtained from a causal correlation framework by incorporating time in a very natural way.
- It admits very precise characterizations of a strong nature. For example, we give a proof system for deriving causal correlations that is shown to be sound and complete.
- Finally, inspite of being fairly general, the correlation relationships definable are efficiently computable. For example, we give a linear time algorithm for computing correlations for our causal framework.

The rest of the paper is organized as follows. We illustrate and make precise some intuitions about events and event correlation in Section 2. These intuitions motivate the formal development of causal correlation in Section 3 and temporal correlation in Section 4. In Section 5, we illustrate how our formal framework can be used fruitfully to better understand existing systems by considering the example of SMARTS InCharge. We end with some concluding remarks in Section 6.

2 Events, Faults, and Correlation

An ECS correlates events to diagnose faults. In this section, the concepts of event, fault, and correlation are illustrated and made more precise. The intuitions presented here motivate the more formal development presented in later sections.

2.1 Events and Faults

An event is an instantaneous occurrence at a time point. For the purposes of this paper, it is sufficient to assume a discrete model of time, *e.g.*, we can treat the time line to be the set of natural numbers and a time instant to be a natural number. Typically, an event is associated with an *object* in which it occurs; the objects are called *managed* objects (MOs). An object is an entity with some *state* and to model the state of an object, an object can be treated as a collection of attributes whose values may change with time with the attributes bound to the state of the object at each time instant. More precisely, an object is an n-dimensional tuple whose components are functions of time. One class of events are alarms that are directly generated in an object, *i.e.*, one of the attributes of the tuple is the event itself. Another class consists of events that are considered to occur when there is a change in state of the object, *e.g.*, an event can be deemed to occur when one of the components of the tuple is below some fixed threshold value.

A *fault* is an event that is associated with an abnormal network state, *i.e.*, network behavior that deviates from expectation. This deviation can be attributed to hardware/software failures, human errors, design flaws, or a combination of the above. Network faults can be classified as being *hard* or *soft*. A hard fault occurs when MOs fail completely (*e.g.*, a router link failure, link cut, or a server crash); a soft fault occurs when MOs function in a degraded performance state. In general, a soft fault may cause a hard fault, and vice versa.

Events can be classified as being *primitive* or *composite*. Events that are directly generated in or correspond to change of state of a managed object can be considered to be primitive since they are directly *observable*. On the other hand, some events are *non-observable* and cannot be directly detected by hardware or software either because the MO where the event occurs is not monitored or the event is not reported. For example, a router link failure may not be directly reportable. A non-observable event is often a *conceptual* construct. The occurrence of a non-observable event is established by inferring from the pattern of occurrence of other events with which it is correlated. For example, a router link failure may be inferred from the connection failure at a server and client attached to the router. Some of the events with which a non-observable event is correlated may themselves be non-observable as well. One of the functions of an ECS is to establish non-observable events from the observable ones that are reported.

2.2 Event Correlation

Event correlation works by establishing relationships between network events. We say that an event e correlates a set of events e_1, e_2, …, e_k, written $e \Longrightarrow \{e_1, e_2, \ldots, e_k\}$, if $e_1, e_2, \ldots, e_k$ by entering into a relationship with each other and with e define the event pattern e. An ECS aids in the following:

- Detection or isolation of faults. For example, consider a simple network consisting of clients that are connected to servers through routers and repeaters. In this monitored network, alarms can be generated by servers as *server connection failure* (s_i), or by clients as *client connection failure* (c_i), which can be attributed to either *router link failure* (r_1) or *repeater failure* (r_2). Without this knowledge, defined as, r_1 or $r_2 \Longrightarrow c_i$ or s_i, a network operator or an automated system cannot efficiently and effectively pin-point the root cause of (possibly large number of) *connection failure* alarms.

- Filtering events. In the above example, the correlation knowledge enables the ECS to report the primary alarms (states of routers and repeaters, i.e., r_1 or r_2) while suppressing the secondary and potentially numerous connection failure alarms.

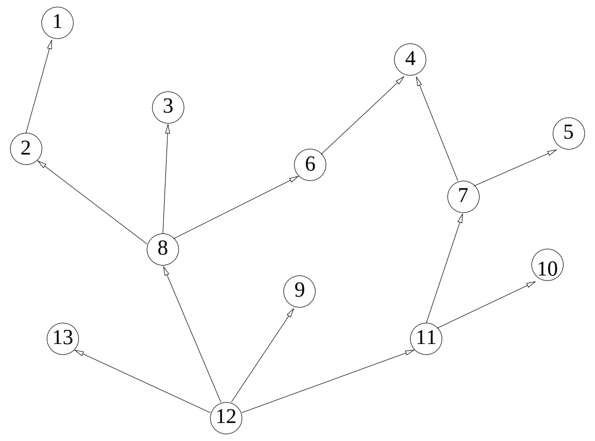

Figure 1: An example of a causality graph

- Performance tuning. When root causes of network faults are isolated efficiently and effectively, performance degradations are short-lived and MOs related to faulty network events can be tuned effectively.

The relationships between events used for event correlation and filtering can be classified as being *causal* or *temporal*. We next consider these two forms of correlation and illustrate with examples how these relationships can be combined together for more comprehensive event correlation.

3 Causal Correlation and Filtering

The simplest relationship between events that can be considered is a *cause-effect* or *causality* relation. Causal correlation and filtering is then the problem of determining root causes of faults by using this causality relationship among events. Using $e \prec e'$ to denote that the event e causes the event e', it is natural to consider the causality relation to be a *strict order*, *i.e.*, a binary relation that is irreflexive and transitive. For any strict order $\prec$, we can define its *skeleton*, denoted $\to$, as follows: $e \to e'$ if $e \prec e'$ and there is no e'' such that $e \prec e'' \prec e'$. Conversely, given the skeleton relation, we can recover the complete causality relationship by taking its transitive closure, *i.e.*, $e \prec e'$ if there are $e_1, \ldots, e_n$ such that $e \to e_1 \to \cdots \to e_n \to e'$. By irreflexivity, the skeleton of a causality relationship can be represented as a directed acyclic graph (DAG) whose nodes are the events and edges the relationship $\to$. Thus, even though the skeleton and the complete causality relationship determine the same pieces of information, the skeleton admits a simple diagrammatic presentation. It is also conceptually easier to define the correlation and filtering properties using the skeleton; we therefore prefer to work with this as our framework for representing the causal structure among events.

Definition 3.1 A *causal event structure* $\mathcal{E}$ is given by a pair $\langle E, \to \rangle$ where E is a set of events and $\to \subseteq E \times E$ is a binary relation on events such that the transitive closure of $\to$ is irreflexive.

Figure 1 shows a causal event structure on the set of events $\{1, \ldots, 12\}$ with the edges defining the $\to$ relation; the condition on $\to$ required by Definition 3.1 merely corresponds to checking that the graph does not contain any directed cycles. Given the causal information about events represented by a causal event structure, the purpose of an ECS is to deduce the correlation relationship, $e \Longrightarrow \{e_1, \ldots, e_k\}$, to be read as saying that the event e correlates all the reported events $e_1, \ldots, e_k$. Then, if the events $e_1, \ldots, e_k$ are reported, the ECS can be used to filter down to e since the presence of all the events $e_1, \ldots, e_k$ is entailed by the event e. Defining the correlation relationship $\Longrightarrow$ is not straightforward — we therefore consider some motivating examples before presenting the precise definition.

Consider again the causality relationship given by Figure 1. Clearly, it would be reasonable to consider the event 8 to correlate the events $2, 3, 6$, *i.e.*, $8 \Longrightarrow \{2, 3, 6\}$ since the event 8 causes $2, 3, 6$. We can also conclude that $8 \Longrightarrow \{1, 2, 3, 6\}$ since 8 causes event 1 transitively via 2. On the other hand, we do not want to conclude that event 8 correlates the set $\{2, 3, 6, 7\}$, since the event 7 cannot be caused by 8. Less obviously, we also do not want to define that 8 correlates the set $\{3, 6\}$, since even though 8 causes both 3 and 6, it would be incorrect to conclude the presence of 8 as the cause, since if event 8 had indeed occurred then we would have also seen event 2 which is however not reported in the set. However, because events may be non-observable, we cannot necessarily require the *explicit* presence of all the effects of an event in the set of reported events. For example, $8 \Longrightarrow \{1, 3, 6\}$, since even though 2 is not present in the set of reported events we can infer its occurrence from the reporting of event 1. Finally, note that these ideas need to be applied recursively: $11 \Longrightarrow \{4, 5, 10\}$ since the presence of 7 can be concluded from the presence of 4 and 5, but 11 does not correlate $\{5, 10\}$ since the event 5 alone is insufficient evidence to conclude the presence of 7. These examples should suggest that the correlation relationship, $\Longrightarrow$, does not admit a simple characterization such as, *e.g.*, the transitive closure of the $\rightarrow$ relationship. To define correlation, we consider a slightly more general relation of the form $S_1 \Longrightarrow S_2$, where S_1, S_2 are sets of events. In other words, we allow the left hand side of the relation to be a set rather than a single event; the original relation can be obtained by taking S_1 to be a singleton set. This more general relation is given by a proof system for proving judgements of the form $S_1 \Longrightarrow S_2$. In our proof system, we use the notation

$$\frac{J_1, \ldots, J_n}{J}$$

for an inference rule that says that if all the judgements $J_1, \ldots, J_n$ can be proved then we can prove the judgement J by application of this rule. We use J to denote an axiom that says that J is provable (without having to prove anything else).

Definition 3.2 Let $\mathcal{E} = \langle E, \rightarrow \rangle$ be a causal event structure. For any $e \in E$, we define the immediate effects of e as follows:

$$\mathit{ImmEff}(e) \quad = \quad \{e' \mid e \rightarrow e'\}$$

For any sets $S, S' \subseteq E$, we say that $S \Longrightarrow S'$ if it is provable using the following axioms and proof rules.

$$\begin{array}{rcl}
\text{(Effects)} & \{e\} \Longrightarrow \mathit{ImmEff}(e) & \text{if } \mathit{ImmEff}(e) \neq \emptyset \\[1ex]
\text{(Reflexivity)} & S \Longrightarrow S & \\[1ex]
\text{(Transitivity)} & \dfrac{S_1 \Longrightarrow S_2 \qquad S_2 \Longrightarrow S_3}{S_1 \Longrightarrow S_3} & \\[2ex]
\text{(Union)} & \dfrac{S_1 \Longrightarrow T_1 \qquad S_2 \Longrightarrow T_2}{S_1 \cup S_2 \Longrightarrow T_1 \cup T_2} &
\end{array}$$

The proof system given in Definition 3.2 consists of very few rules that are simple and natural; yet, these suffice to characterize the correlation relation we are interested in. Later, we will show in a very precise sense that this proof system is complete, but for now we illustrate its power with some examples using the causality structure given in Figure 1. We have $\{8\} \Longrightarrow \{2,3,6\}$ immediately by the axiom (Effects). We can also derive $\{8\} \Longrightarrow \{1,3,6\}$ using the proof system as follows. By (Reflexivity), we have that $\{3,6\} \Longrightarrow \{3,6\}$ and by (Effects) that $\{2\} \Longrightarrow \{1\}$. Using (Union) on these judgements, we can derive $\{2,3,6\} \Longrightarrow \{1,3,6\}$. Combining this judgement with $\{8\} \Longrightarrow \{2,3,6\}$ using (Transitivity), we can derive $\{8\} \Longrightarrow \{1,3,6\}$. If we use the rule (Union) on this latter judgement with the already derived judgement $\{8\} \Longrightarrow \{2,3,6\}$, we can derive $\{8\} \Longrightarrow \{1,2,3,6\}$.

Some remarks regarding the proof system in Definition 3.2 are in order. First, we note that there is a non trivial gap between the causal correlation relation defined by the proof system and the causal event structure from which it is defined, with the causal event structure being a far more direct specification of the properties of events in a network that one may be aware of. Thus, even in the simplest case of causal correlation, this is a concrete illustration of the utility of having a separate representation for expressing the properties of events in a network (the causal event structure) rather than directly expressing the correlation relation. Secondly, even though the proof system is defined for the specific case of causal event structures, it is very useful as a basis framework for defining other notions of interest. For example, it is implicit in a causal event structure that each of the effects of an event e are independent and are therefore all caused whenever event e happens. Referring again to Figure 1, when event 8 occurs, we have events 2,3,6 all occurring. One may want to express richer causality patterns such as saying that event 8 causes events 2 and 3 to happen or event 6 to happen. This can be incorporated in the framework of the proof system by simply modifying the (Effects) axiom and keeping all other rules intact. The last considered example can be expressed by the axioms $\{8\} \Longrightarrow \{2,3\}$ and $\{8\} \Longrightarrow \{6\}$. Finally, the axioms and rules of the proof system provide a natural structure for defining a *probabilistic* notion of event correlation whereby we can associate a probability with a correlation statement $S_1 \Longrightarrow S_2$ being true. This is applicable in a scenario where some primitive events may be "lost" or not reported. Due to space considerations, we do not detail this definition in this paper.

3.1 Causality from Propositional Relations

While a causal event structure is a fairly intuitive description of the causal relationship among events, we can also consider an alternate representation of this causal information by defining the occurrence of events in terms of propositional or boolean conditions on other events. For example, using the operators $\wedge$ and $\vee$ for logical conjunction and disjunction respectively, we can define $e = (e_1 \wedge e_2) \vee e_3$ to mean that the event e occurs when both events e_1 and e_2 occur or when e_3 occurs. There are two reasons for considering this alternate representation: (a) it helps establish in a very precise sense the soundness and completeness of the proof system given in Definition 3.2, (b) this point of view is more useful in giving a framework for temporal correlation.

Definition 3.3 A causal event system $\mathcal{C}$ is given by a set of events E, and a set of equations of the form $e = \varphi$ where $e \in E$ and φ is a expression constructed using propositional boolean operations and elements of E as propositional symbols.

Causal event structures, as given by DAGs, can be described as causal event systems defined by equations as follows.

Example 3.4 Given a causal event structure $\mathcal{E} = \langle E, \rightarrow \rangle$, we can define the corresponding causal event system $\mathcal{C}(\mathcal{E})$ to consist of the event set E and equations given by

$$e = e_1 \wedge \ldots \wedge e_n$$

where $\mathit{ImmEff}(e) = \{e_1, \ldots, e_n\}$ and is non-empty.

For example, the causal event system representation of the causal structure given in Figure 1 is the following set of equations.

$$\{2 = 1, 6 = 4, 8 = 2 \wedge 3 \wedge 6, 7 = 4 \wedge 5, 12 = 13 \wedge 8 \wedge 9 \wedge 11, 11 = 7 \wedge 10\}$$

For an event set E, we can define a run ρ to be a map from E to the set $\{\mathsf{true}, \mathsf{false}\}$. Intuitively, a run denotes an execution of the network with $\rho(e) = \mathsf{true}$ denoting that event e occurred and $\rho(e) = \mathsf{false}$ denoting that event e did not occur. Given a run ρ and a boolean expression φ constructed from events in E, we use $[\![\varphi]\!]\rho$ to denote the truth value of φ under the truth assignment given by ρ; for example, if $\rho(1) = \mathsf{true}, \rho(2) = \mathsf{false}$, we have that $[\![1 \wedge 2]\!]\rho = \mathsf{false}$. The equations given in a causal system describe certain properties among events that we know to hold in all possible executions of a network: this is formally captured by the following definition.

Definition 3.5 For a causal event system $\mathcal{C}$ with event set E, a *feasible run* is a map from E to $\{\mathsf{true}, \mathsf{false}\}$ such that for any equation $e = \varphi$ in $\mathcal{C}$, we have that $\rho(e) = [\![\varphi]\!]\rho$.

We can now define event correlation with respect a causal event system given by equations as follows. We define the more general notion where the left hand side of the correlation can be a set; the specific case when this set is a singleton gives us the definition for when it is a single event.

Definition 3.6 Suppose $\mathcal{C}$ is a causal event system with event set E. For $S_1, S_2 \subseteq E$, we say that the correlation judgement $S_1 \Longrightarrow S_2$ holds if for any feasible run ρ, we have that:

(a) If $\rho(e) = \mathsf{true}$ for every $e \in S_1$, then $\rho(e) = \mathsf{true}$ for every $e \in S_2$

(b) If $\rho(e) = \mathsf{true}$ for every $e \in S_2$, then $\rho(e) = \mathsf{true}$ for every $e \in S_1$

Intuitively, condition (a) captures the notion that the events in S_1 entail all the events in S_2 that are reported, and condition (b) captures the notion that it is correct to conclude from the reported events in S_2 that the events in S_1 did occur. For example, referring to the equations generated from the causality graph given in Figure 1, it is condition (a) that invalidates the correlation $\{8\} \Longrightarrow \{2, 3, 6, 7\}$, while it is condition (b) that invalidates the correlation $\{8\} \Longrightarrow \{3, 6\}$. Remarkably, we can now show that the proof system given in Definition 3.2 can be used to derive exactly all the correlations that are said to hold by Definition 3.6.

Theorem 3.7 *Let $\mathcal{E}$ be a causal event structure and $\mathcal{C}(\mathcal{E})$ be its corresponding causal system as defined in Example 3.4. For any S_1, S_2 that are subsets of the event set, we have that $S_1 \Longrightarrow S_2$ is provable in the proof system of Definition 3.2 if and only if $S_1 \Longrightarrow S_2$ holds with respect to $\mathcal{C}(\mathcal{E})$ by Definition 3.6.*

3.2 Deducing Event Correlation

Using the characterization given by Theorem 3.7, we are now ready to describe an algorithm to deduce event correlation properties for a given causal event structure. More precisely, given a causal event structure and a set of reported events $\{e_1, \ldots, e_n\}$, we will describe an algorithm for computing a set S such that $S \Longrightarrow \{e_1, \ldots, e_n\}$ with S optimal in the sense of being the "most conceptual" possible.

Given a causal event structure $\mathcal{E} = \langle E, \rightarrow \rangle$, we first define the rank of any event $e \in E$ as follows:

$$Rnk(e) \quad = \quad \begin{cases} 0 & \text{if } ImmEff(e) = \emptyset \\ Max(\{Rnk(e') \mid e' \in ImmEff(e)\}) + 1 & \text{otherwise} \end{cases}$$

For example, for the causality structure given in Figure 1, we have that $Rnk(1) = 0$, $Rnk(2) = 1$, $Rnk(8) = 2$, and $Rnk(12) = 3$. Intuitively, the rank of an event corresponds to its "degree of conceptuality". Events with rank 0 do not have any effects and can therefore be thought of as primitive events that can only be directly observable, and the higher the rank of an event, the more conceptual it is in the sense that there is a potentially larger set of events that can be inferred to have occurred from its presence. We define the rank of a set of events by taking the maximum of the ranks of its elements, *i.e.*, for a set $S \subseteq E$,

$$Rnk(S) \quad = \quad Max(\{Rnk(e) \mid e \in S\})$$

Assume now that we are given a set $S \subseteq E$ of reported events. Using Definition 3.5, we will now define a feasible run ρ_S that assigns all the reported events in S to be true and assigns minimally additionally events to be true. In other words, ρ_S assigns events to be true only if it is "forced" to, given the assumption that the events in S have been reported. The value of ρ_S for an event e is defined by induction on $Rnk(e)$ and can therefore be computed in stages, where at Stage i, we compute $\rho_S(e)$ for all events e with $Rnk(e) = i$.

Base Case: If $Rnk(e) = 0$, then we take $\rho_S(e) = \mathsf{true}$ if $e \in S$ and $\rho_S(e) = \mathsf{false}$ otherwise.

Induction Step: If $Rnk(e) = i + 1$ then

$$\rho_S(e) \quad = \quad \begin{cases} \mathsf{true} & \text{if } e \in S \\ \mathsf{true} & \text{if } \forall e' \in ImmEff(e).\rho_S(e') = \mathsf{true} \\ \mathsf{false} & \text{otherwise} \end{cases}$$

Referring again to the example given in Figure 1, suppose that we are reported the set $S = \{1, 3, 6\}$. At the base case, we compute $\rho_S(1) = \rho_S(3) = \mathsf{true}$ and ρ_S for events 4,5,9,10,13 to be false. At the first induction step, we compute $\rho_S(6) = \mathsf{true}$ (since it is present in S), $\rho_S(2) = \mathsf{true}$ (since $\rho_S(1) = \mathsf{true}$), and $\rho_S(7) = \mathsf{false}$. At the next induction step, $\rho_S(8) = \mathsf{true}$ (since $\rho_S(2), \rho_S(3), \rho_S(6)$ are true), $\rho_S(11) = \mathsf{false}$. At the final induction step, $\rho_S(12) = \mathsf{false}$.

Given a set of reported events S, we can then compute its minimal cause to be the events that are assigned true by ρ_S and such that they do not have any cause that is also assigned true by ρ_S:

$$MinCause(S) \quad = \quad \{e \mid \rho_S(e) = \mathsf{true} \text{ and } \neg\exists e'.(e' \rightarrow e \text{ and } \rho_S(e') = \mathsf{true})\}$$

Continuing with the example of $S = \{1, 3, 6\}$, we had calculated ρ_S to be true only for the events 1,2,3,6,8. Among these, events 1,2,3,6 have causes which have been assigned true, and thus we get that

$$MinCause(\{1, 3, 6\}) = \{8\}.$$

In general, we can show that for any causal event structure, *MinCause* computes a set that is a correct correlate and that is of the maximum possible degree of conceptuality.

Theorem 3.8 *Suppose that $\mathcal{E} = \langle E, \rightarrow \rangle$ is a causal event structure, and $S \subseteq E$ is a set of reported events. Then $MinCause(S)$ has the following two properties.*

1. Correctness: $\mathit{MinCause}(S) \Longrightarrow S$

2. Optimality: *For any S' such that $S' \Longrightarrow S$, we have that $\mathit{Rnk}(S') \leq \mathit{Rnk}(\mathit{MinCause}(S))$*

Finally, we analyze the running time of the algorithm for computing *MinCause* for a set S and causal structure $\mathcal{E} = \langle E, \rightarrow \rangle$. We take the size of the causal structure $|\mathcal{E}|$ to be the maximum of $|E|, |\rightarrow|$, *i.e.*, the maximum of the number of nodes and edges in the DAG representation of $\mathcal{E}$. We need to first compute the rank of all the events in E which can be done in $O(|\mathcal{E}|)$ time using a standard topological sort algorithm. In doing all the stages of calculating ρ_S, we process each event $e \in E$ at most once and each edge of $\rightarrow$ at most once. Thus calculating ρ_S can be done in $O(|\mathcal{E}|)$ time. Finally, to calculate $\mathit{MinCause}(S)$ from ρ_S, we need to look at the value of $\rho_S(e)$ for each event once and each edge below it once which therefore also takes $O(|\mathcal{E}|)$ time. We thus have the following theorem that the algorithm runs in time linear in the size of the event structure.

Theorem 3.9 *There is an algorithm that computes for any causal event structure $\mathcal{E}$ and set S of reported events, the set $\mathit{MinCause}(S)$ and whose running time is $O(|\mathcal{E}|)$.*

4 Temporal correlation and filtering

Events happen at particular time instants. We can establish richer correlations between events based on the specific instants of their occurrence by defining temporal relationships between them. Some examples of temporal relationships are: e_1 *followed by* e_2, *first* e_1 event *since* the *recent* e_2 event, e_1 *follows* e_2 *within* 2 *minutes*, e_1 *not within* interval I (e_1 was not observed in the interval I), etc.

Our development of the temporal correlation framework mirrors the development of Section 3.1 extended now to allow temporal operators on events. For the purposes of discussion here, it is not necessary to specify the exact temporal operators that we will be considering. We assume that they are given by some chosen temporal event specification language (TESL) which supports a number of temporal operators (such as followed by, within, etc.).

Definition 4.1 A temporal event system $\mathcal{T}$ is given by a set of events E, and a set of equations of the form $e = \varphi$ where $e \in E$ and φ is an expression constructed using temporal operators, given in a TESL, on elements of E.

Conceptually, the causal event framework of Section 3.1 was obtained by interpreting events to be propositions that have values true or false indicating their occurrence or non-occurrence. To represent temporal relationship, we refine the interpretation of events to include information about the time instants at which they occur. Formally, we interpret time to be some set $\mathbb{T}$ with a linear ordering. We can then interpret an event to be a function from $\mathbb{T}$ to $\{\mathsf{true}, \mathsf{false}\}$ with its instants of occurrence defined by the arguments (time instants) at which it is true. For an event set E, we therefore now define a run ρ to be a map from E to the function space $\mathbb{T} \rightarrow \{\mathsf{true}, \mathsf{false}\}$. As mentioned before, for an event $e \in E$, we think of event e as occurring at time t in the run ρ if $\rho(e)(t) = \mathsf{true}$. For a run ρ and a temporal expression φ constructed from events in E, we can define $[\![\varphi]\!]\rho$ to be another event, *i.e.*, a function from $\mathbb{T}$ to $\{\mathsf{true}, \mathsf{false}\}$. This can be done compositionally in terms of the operations appearing in φ. For example, suppose that φ is the temporal expression $\varphi_1 \mathtt{preceded\_by} \varphi_2$ where φ_1 and φ_2 are some other temporal expressions. Intuitively, the event φ occurs whenever the event φ_1 occurs and has been preceded by an occurrence of the event φ_2. Thus, we take $[\![\varphi]\!]\rho$ to be the function f that is defined by $f(t) = \mathsf{true}$ if $[\![\varphi_1]\!]\rho(t) = \mathsf{true}$ and there is some $t' < t$ such that $[\![\varphi_2]\!]\rho(t') = \mathsf{true}$ and $f(t) = \mathsf{false}$ otherwise. Given the definition of a composite event in a temporal event system $\mathcal{T}$ and some run ρ, we can thus evaluate the value of the composite event as a boolean function of time. A feasible run can then be defined by extending Definition 3.5 to take time into account.

Definition 4.2 For a temporal event system $\mathcal{T}$ with event set E, a *feasible run* is a map from E to $\mathbb{T} \to \{\mathsf{true}, \mathsf{false}\}$ such that for any equation $e = \varphi$ in $\mathcal{C}$, we have that at any time instant $t \in \mathbb{T}$, $\rho(e)(t) = [\![\varphi]\!]\rho(t)$.

We are now ready to define the temporal correlation relation. Intuitively, the entity on the right hand side of the relation, $\Longrightarrow$, corresponds to the monitored observations that are input to the ECS. In the case of causal correlation, this was the set of events that have been reported to have occurred. To be able to make use of the temporal relationships between events, the ECS has to be reported more than just the names of the events that have occurred — it should also be informed about the specific time instants of their occurrence. In the temporal case, we therefore take the correlation relation to be between *histories* of events timestamped with their instants of occurrence. Formally, a history H is a subset of $E \times \mathbb{T}$, *i.e.*, a set of pairs of the form (e, t) indicating that event e occurred at time t.

Definition 4.3 Suppose $\mathcal{T}$ is a causal event system with event set E. For $H_1, H_2 \subseteq E \times \mathbb{T}$, we say that the correlation judgement $H_1 \Longrightarrow H_2$ holds if for any feasible run ρ, we have that:

(a) If $\rho(e)(t) = \mathsf{true}$ for every $(e, t) \in H_1$, then $\rho(e)(t) = \mathsf{true}$ for every $(e, t) \in H_2$

(b) If $\rho(e)(t) = \mathsf{true}$ for every $(e, t) \in H_2$, then $\rho(e)(t) = \mathsf{true}$ for every $(e, t) \in H_1$

While we have not been specific about the choice of the operators in the TESL, it is important from the point of view of the correlation to be computable in *real-time*, that there are no temporal operators whose evaluation requires "looking to the future". For example, if we had the eventually operation $\Diamond$ with the event $\Diamond\varphi$ true at time t if φ is true at some time $t' > t$, then it would not be possible to evaluate the occurrence or non-occurrence of a composite event defined using this operation since we only have access to the history of occurrence of events so far. On the other hand a past version of this operator ∇ with the event $\nabla\varphi$ true at time t if φ is true at some time $t' < t$ would be reasonable to allow in the system.

5 SMARTS InCharge: An Example ECS

InCharge is an ECS developed by SMARTS supporting an object-oriented network modeling language called MODEL and a correlation engine based on an approach called the *codebook* approach [10]. In Section 5.1, we give an overview description of the system. In Section 5.2, we show how the algorithm used in **InCharge** can be obtained as a specific instance of our general causal framework of Section 3 but that it is less powerful than our framework.

5.1 Overview

Event relationships and network configuration information are encoded in a language called MODEL (Managed Object Definition Language) [11]. The MODEL language is an extension of CORBA IDL [3]. It adds new syntactic constructs to specify semantics that cannot be specified in CORBA IDL, such as *relationship*, *events*, *problems*, and *causal propagation*. Given a MODEL specification,

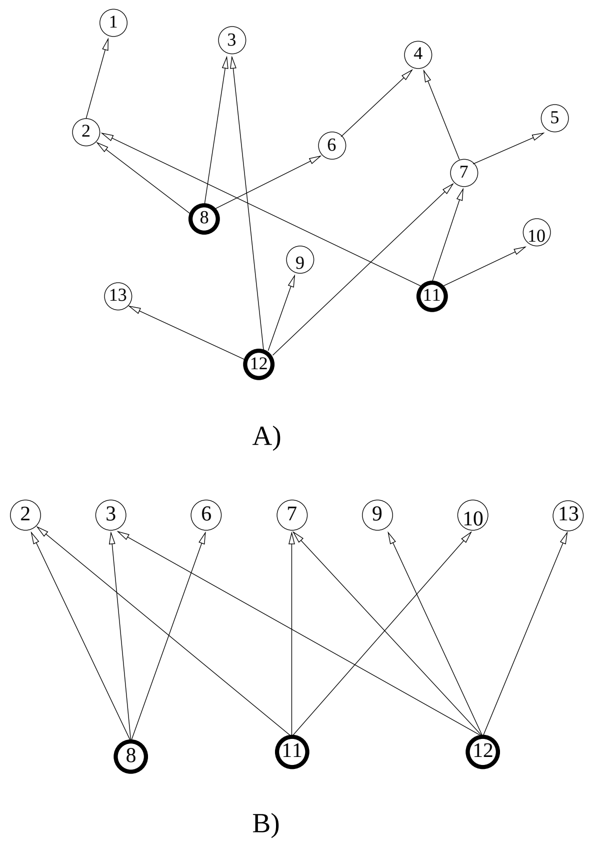

Figure 2: Example event correlation causality graph for InCharge model

InCharge then extracts a causality graph from it, converts it into a *codebook*, and uses the codebook to perform correlation.

In the causal model supported by this system two types of events are distinguished: *problem* (p) and *symptom* (s) events. The problem events manifest themselves as symptom events. In the terminology of Section 2, symptom events correspond to observable events and problem events to non-observable events. A causality graph is converted into a *correlation graph* which is a *bipartite graph* and is obtained as follows. If $p \to e_1 \ldots \to e_n$ in the causality graph, then in the correlation graph there is an edge from p to the first of $e_1, \ldots, e_n$ that is a symptom, *i.e.*, there is an edge $p \to e_i$ where $e_1, \ldots, e_{i-1}$ are not symptom events. A version of Figure 1 is shown in Figure 2(a), where modifications are necessary since causality between problems is not supported in the InCharge model. The correlation graph corresponding to the causality graph shown in Figure 2(a) is shown in Figure 2(b). The set of symptoms connected to a problem p in the correlation graph is called its *code vector*. The code vectors (together called the *correlation matrix*) for the problems 8, 11, and 12 are as follows:

$$\begin{array}{rl} 8: & [1,1,1,0,0,0,0] \\ 11: & [1,0,0,1,0,1,0] \\ 12: & [0,1,0,1,1,0,1] \end{array}$$

assuming the order of symptoms is from left to right as shown in the figure. The occurred alarms (symptoms) emitted from a network constitute an *alarm vector*.

The alarm correlation problem is then to find problems whose codes optimally match an alarm vector. Borrowing techniques and concepts from *information coding theory*, the authors of InCharge suggest a method to reduce the correlation matrix size so that each code vector still uniquely identifies a problem even in the presence of *lost* or *spurious* events. The reduced matrix is called a *codebook*. The reduction problem can be stated as follows: given a set of code vectors each of length L, *minimum Hamming distance* d between code vectors, compress the code vectors to a length l so that d is maintained. For example, if we want to maintain minimum Hamming distance of 2 in the code vector above, we can prune the last three symptoms (9, 10, 13), giving the code book:

$$\begin{array}{rl} 8: & [1,1,1,0] \\ 11: & [1,0,0,1] \\ 12: & [0,1,0,1] \end{array}$$

The minimum Hamming distance for this codebook is min(d(8,11), d(8, 12), d(11, 12)) = min(3, 3, 2) = 2. The decoder will still match a problem against an alarm vector, even in the presence of lost or spurious events. But exactly one symptom can be lost or spurious. For example, if the alarm vector is [0, 1, 1, 0] (first symptom lost), or [1, 1, ,1, 1] (last one spurious) problem 8 will be detected.

5.2 Discussion

InCharge does not support temporal correlation, as discussed in Section 4. Even for causal correlation, the correlation model supported in InCharge is clearly less general than the one presented in Section 3. In particular, problem to problem causality is not supported, *i.e.*, in this model a problem cannot be a symptom of another problem. Hence, it would not be able to handle the situation where a problem (fault) does not exhibit any symptom but is known to be caused by another problem, and where both of the problems may require handling.

Ignoring the issue of lack of support for problem to problem causality, we next compare the correlation properties that can be inferred by InCharge with our general framework for the specific causality graphs that are supported in InCharge. First, we note that the transformation of the causality graph performed by InCharge described in Section 5.1 to obtain a bipartite graph can be characterized in terms of our proof system of Definition 3.2 as follows. It merely corresponds to restricting the use of (Effects) axiom $\{e\} \Longrightarrow \mathit{ImmEff}(e)$ to the case that e is a problem event. Thus, any correlation inferred by InCharge is provable by the proof system of Definition 3.2. On the other hand, the transformation to a bipartite graph results in loss of information, *i.e.*, there are correlations that are inferrable in the original graph that cannot be obtained in InCharge. Referring to our example of Figure 2, assume that the events $\{1, 3, 4\}$ are reported. Our correlation framework used on the graph of Figure 2(a), would then be able to conclude 8 as the cause event. On the other hand, InCharge using the bipartite graph of Figure 2(b) would not infer this correlation since the events $2, 6$ have not been reported in this set.

Finally, the codebook approach may not be able to quickly adapt to changes in network configuration information. This is because the addition of a new event will require an appropriate entry in the correlation matrix, and because of the cause-effect relations of the new event with the already existing events, the new entry may violate the minimum Hamming distance property in the current codebook. Hence the costly procedure of computing a new codebook may be required. On the other hand, our algorithm of Section 3.2 can easily incorporate such a network change because the addition of a new event only requires calculating its rank which is a purely local computation.

6 Conclusion

We have presented simple models for describing causal and temporal relationships between network events and given precise formulations of which event correlations are valid for these models. This study distinguishes itself from previous work on event correlation systems in addressing the identification of precise criteria for the correctness of inferred event correlations. Although the correctness definitions have been given in the context of the specific models considered in this paper, the generality and simplicity of the models allows different models used in other systems to be mapped to them (as was illustrated in Section 5). Thus, our work can be seen as presenting a general specification of the event correlation problem that allows a formal analysis of the correctness of any existing or future systems. On the other hand, it is important to point out that the very simplicity of the models makes them unrealistic as mechanisms for directly expressing network event relationships in any practical system — the intended audience of this work is not the *user* but the *developer* of an ECS, for whom we believe that it provides a useful conceptual intermediate medium for understanding the correctness and relative expressive power of their system.

There are many directions for further work. In the case of causal correlation, we presented an algorithm for inferring a causal correlation and identified a class of causal relations (those satisfying Definition 3.1) for which the algorithm was provably correct and optimal. On the other hand, in the temporal case, while we defined the correctness condition for correlation we did not discuss an algorithm for temporal correlation. Such a study would have to be done in the context of specific temporal operators, and an analysis of the different algorithmic complexity for different temporal operators would shed light on what kinds of temporal relations are feasible to include in any practical system. This study has also not addressed the inclusion of network-topology information. In this regard, a good framework would describe how to smoothly and automatically integrate the causality and temporal information at individual nodes in a network to obtain the causality and temporal relationships for the whole network, using its topology. More generally, the work presented here uses as its starting point the causality graph or temporal relationship between events — further work should address the automatic generation of these structures from information that is more succinct and directly expressible for a network. Most of these issues are addressed in the Java based distributed ECS called *ECLite* [8, 4] developed at Lucent Bell Labs. We plan to address the formal issues omitted in this paper with a characterization of ECLite in a future report.

References

[1] http://www.seagatesoftware.com/nervcpro/. Seagate NervCenter Pro Web site, 1997.

[2] http://www.cabletron.com/products/items/sa-csi1016/. Cabletron Web site, 1997.

[3] CORBA, the common object request broker: Architecture and specification. Object Management Group, July 1995.

[4] http://www.bell-labs.com/~masumh/projects.html. ECLite Fault Correlation System Web Site, 1998.

[5] http://www.gensym.com. Gensym Web site, 1997.

[6] Masum Hasan. An active temporal model for network management databases. In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Proceedings of the IEEE/IFIP Fourth Interna-*

tional Symposium on Integrated Network Management, pages 524–535. Chapman and Hall, May 1995.

[7] Masum Hasan, Lawrence Ho, Frank Feather, and Binay Sugla. The software frameworks for network management event correlation and filtering systems. Technical report, Lucent Bell Labs Research, Network and Service Management Research Department, August 1998.

[8] Masum Hasan and Binay Sugla. The ECLite fault correlation system. Technical Report BL0113540-980904-04TM, Lucent Bell Labs Research, Network and Service Management Research Department, August 1998.

[9] G. Jakobson and M. Weissman. Real-time telecommunication network management: extending event correlation with temporal constraints. In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Proceedings of the IEEE/IFIP Fourth International Symposium on Integrated Network Management*, pages 290–301. Chapman and Hall, May 1995.

[10] S. Klinger, S. Yemini, Y. Yemini, D. Oshe, and S. Stolfo. A coding approach to event correlation. In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Proceedings of the Fourth IEEE/IFIP International Symposium on Integrated Network Management*, pages 266–277. Chapman and Hall, May 1995.

[11] A. Mayer, S. Kliger, D. Ohsie, and S. Yemini. Event modeling with the MODEL language. In Aurel A. Lazar, Roberto Saracco, and Rolf Stadler, editors, *Proceedings of the Fifth IEEE/IFIP International Symposium on Integrated Network Management*, pages 625–637. Chapman and Hall, May 1997.

[12] Y. A. Nygate. Event correlation using rule and object based techniques. In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, *Proceedings of the IEEE/IFIP Fourth International Symposium on Integrated Network Management*, pages 278–289. Chapman and Hall, May 1995.

[13] Kenneth R. Sheers. HP OpenView Event Correlation Services. *Hewlett-Packard Journal*, October 1996.

[14] http://www.tivoli.com/o_products/html/datasheet_listing.html. Tivoli Web site, 1997.