Event correlation with boxed pomsets

Thomas Gazagnaire[1] and Loïc Hélouët[2]

[1] IRISA/ENS Cachan,Campus de Beaulieu, 35042 Rennes Cedex, France
[2] IRISA/INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France

Abstract. This paper proposes a diagnosis framework for distributed systems based on pomset languages. Diagnosis is performed by projecting these models on a collection of observable labels and then synchronization with an observation. This paper first proposes a new model called boxed pomset languages, which extends classical pomset-based languages as so called High-level Message Sequence Charts. It can describe infinite scenarios, and has good properties with respect to projections. We then give a solution for the event correlation problem (knowing whether two observed alarms are causally related) for pomset languages.

1 Introduction

Communication systems have become more and more complex over the recent years. Usually, several telecommunication operators share the same physical network to provide services to their clients. In this context, when a breakdown occurs, finding what really happened and who is responsible for it is becoming a major challenge.

Such kind of telecommunication breakdown happened in France in November 2004. In several towns, the whole telecommunication network was unavailable, and worse, even emergency numbers were disabled. It took a full day to restore normal communication. The cause for this trouble was made public much later: a software error in a voice over IP application had forced several equipments to switch off, and as a result, the whole network collapsed.

Similarly, the spreading of DSL (still with several operators sharing a common network) now cause some reliability problems. When a breakdown occurs the problem may be due to the physical network (at the level of the local hook-up or at upper levels), to the service provider, or even worse it may be a consequence of bad interactions between services from several providers. In this situation, finding the cause of the failure is difficult, and the time to get an explanation may be several weeks. According to France Telecom [1], several providers did not develop sufficient tools needed to detect faults in networks. Beyond technical concerns (repairing the incriminated hardware or replacing software), there are also economical reasons for diagnosis techniques: one wants to find *who* is responsible for a failure of the system - or equivalently *what is the root cause* of the failure. In such a situation, the origin of a breakdown becomes as important as the fault itself.

In order to quickly fix problems, almost every part of a modern network provides data about what it is doing: operating systems log systems and security

[1] Le monde, 01/08/2007

events, servers keep records of what they do, applications log errors, warnings and failures, firewalls and VPN gateways record suspicious traffic, routers and switches watch packets between network segments,... In a protocol such as Simple Network Management Protocol (SNMP) [7], these equipments forward alerts to a central management console. Besides monitoring their own behavior, all these agents receive and relay messages from other network components, and may in turn generate new alerts, leading to a propagation of an alarm over the whole network. A single problem can hence generate overabundant alarms, that are collected in huge log files. After a breakdown, these logs have to be searched, but they are often so big that data provided by the network can not be exploited without dedicated tools. Moreover, monitoring everything in a system is not possible because sensors cannot be placed everywhere, and thus only a subset of what occurs in the network is reported in logs. Hence to understand completely what happened during the failure, one needs to rely on partial observations, but also on his knowledge of the systems.

In practice, logs are often analyzed and simplified with the help of some simple rules such as *compression* (takes multiple occurrences of the same event, examines them for duplicate information, removes redundancies and reports them as a single event), *counting* (reports a specified number of similar events as a single one), *suppression* (associates priorities with alarms and suppresses an alarm with low priority if an event with higher-priority has occurred), *generalization* (generates a log of higher-level events from the initial log) [3], *correlation* which establishes "cause and effect" relation between events [12]. All these rules are implemented in expert systems, that read complete logs and output simplified log files. These summaries are then read by a specialist who tries to find a scenario for the failure and its root causes.

Additionally, several model-based formal techniques have been proposed recently to diagnose systems. Sampath et al [14] propose a fault detection technique from finite state machines (FSM), that distinguish safe and faulty states. Lafortune et al also propose a notion of diagnosability for their model. A system, described as a FSM, is diagnosable if for a given sequence of observable transitions, one cannot find two compatible runs of the system such that one that leads to a safe state, and the other to a faulty state. Jeron et al [10] describe a similar approach with enhanced fault models. Benveniste et al [2] propose Petri Nets based diagnosis techniques. They recover complete explanations from an incomplete observation using a Petri Net model of the monitored system. Hélouët et al [8] show how to recover explanations of a fault from a partial observation of a distributed system using High-Level Message Sequence Charts (HMSC) [9].

This paper investigates a model-based diagnosis technique using pomset languages, that are more powerful than FSM and HMSCs. Roughly speaking, such languages are automata labeled by partial orders. The major difference with HMSCs resides in the kind of pomset labeling the automata and in the sequential composition rule that can be parametrized. Using this model, we provide techniques to retrieve explanations (from a given partial observation o, provide all explanations; i.e. runs of the model, that are compatible with o) and to per-

form event correlation (infer from a partial ordering of events in an observation o whether two events should be causally related). More precisely, we show that deciding whether two observed events are ordered in all runs of a model is CoNP-complete. When the collection of possible labels is fixed and the observation has no auto-concurrency, the problem is in NLOGSPACE and we give an effective algorithm to compute the reconstructed causal order explaining the observation.

This document is organized as follows: Section 2 introduces the basic definitions of pomset languages and boxed pomset languages that will be used as models of monitored systems. Section 3 establishes the main properties of these languages. Section 4 uses these results to solve the event correlation problem. Section 5 concludes this work and gives some perspectives.

2 Pomsets, boxed pomsets, and Pomset languages

Pomsets are a very natural representation to describe runs of distributed systems. Furthermore, they avoid the well known state-space explosion problem due to interleaving. Popular languages based on partial orders such as HMSCs [9] are now standardized. This section introduces a new pomset language called *boxed pomsets*, that has nice properties with respect to projection and embeds the expressive power of HMSCs. The following definitions are mainly due to Gischer [6] and were reused later by Pratt [13].

Pomsets. A *labeled partial order* (or *lpo*) over a set E with labels Σ is a structure $(E, \leq, \lambda, \Sigma)$ where $\leq$ partially orders E and $\lambda : E \to \Sigma$ assigns an element of Σ to each element of E. When needed, we will denote by $(E_p, \leq_p, \lambda_p, \Sigma_p)$ the components of lpo p. Labels in Σ should be considered as types of actions that can be performed by a system, E as instances of these actions representing *events* in a run of a distributed system. The set of all events is denoted by $\mathbb{E}$. A lpo is *auto-concurrent* iff one can find two incomparable events $e, e' \in E$ (i.e. $e \not\leq e'$ and $e' \not\leq e$) such that $\lambda(e) = \lambda(e')$.

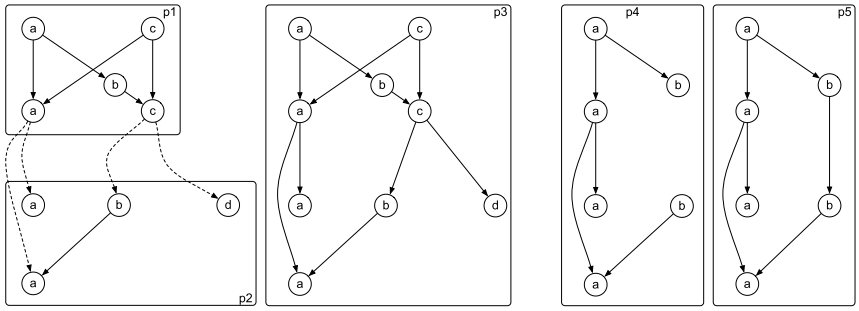
A *map* of lpos $(f, t) : (E_1, \leq_1, \lambda_1, \Sigma_1) \to (E_2, \leq_2, \lambda_2, \Sigma_2)$ consists of a monotone map f: $(E_1, \leq_1) \to (E_2, \leq_2)$ of partially ordered sets and an alphabet map t: $\Sigma_1 \to \Sigma_2$ such that for all e in E, $\lambda_2(f(e)) = t(\lambda_1(e))$. An *isomorphism* of lpos is a map (f, t) where f is an isomorphism of partially ordered sets and t is the identity function.

A *pomset* is the isomorphism class $[E, \leq, \lambda, \Sigma]$ of a lpo $(E, \leq, \lambda, \Sigma)$. More intuitively, pomsets pay attention to cardinality , labeling and ordering of events, but not to their identity. ¿From now on, we consider that the set of events $\mathbb{E}$ and its labeling function $\lambda : \mathbb{E} \to \Sigma$ are fixed. Thus we will denote a pomset p by $[E_p, \leq_p]$ instead of $[E_p, \leq_p, \lambda_p, \Sigma_p]$, because Σ_p is a subset of Σ and λ_p is the restriction of λ to the domain E_p. We will also denote by $\mathbb{P}$ the set of all possible pomsets.

A *projection* of a pomset p on an observable alphabet Σ_o is a function $\pi_{\Sigma_o} : \mathbb{P} \to \mathbb{P}$ which restricts p to observable labels, i.e. $\pi_{\Sigma_o}(p) = [E_p \cap E_{\Sigma_o}, \leq_p \cap E^2_{\Sigma_o}]$ with $E_{\Sigma_o} = \lambda^{-1}(\Sigma_o)$.

Given a predicate ψ which associates a boolean to each pair of Σ^2, we can define the *composition* of pomsets p_1 and p_2, denoted by $p_1 \circledcirc_\psi p_2$, or simply

$p_1 \circledcirc p_2$, as an operator that computes the disjoint union of two pomsets and then adds an ordering between all pairs of events $(e, e') \in E_{p_1} \times E_{p_2}$ such that $\psi(\lambda(e), \lambda(e'))$. More formally, we have $p_1 \circledcirc_\psi p_2 = (E_{p_1} \uplus E_{p_2}, (\leq_1 \uplus \leq_2 \uplus \leq_\psi)^*)$ where $\leq_\psi = \{(e, e') \in E_{p_1} \times E_{p_2} \mid \psi(\lambda(e), \lambda(e'))\}$. This composition is similar to the local composition of pomsets defined by Pratt [13]. The parameterization of ψ makes the composition law able to express several classical operators such as the parallel composition when $\psi(a, b)$ is $false$ for all $a, b \in \Sigma$, the strong concatenation, that is sometimes used to compose MSC's, when $\psi(a, b)$ is $true$ for all $a, b \in \Sigma$, and the weak sequential concatenation when Σ is decomposed into p disjoint sets $\Sigma_1, \ldots, \Sigma_p$ representing respectively all actions that can be executed by processes $1, \ldots, p$ and $\psi(a, b)$ holds when $\exists i \in 1 \ldots p$ such that $a, b \in \Sigma_i$. From now on, when ψ is clear from the context, we will only write $p_1 \circledcirc p_2$ instead of $p_1 \circledcirc_\psi p_2$. Figure 1 gives an example of pomset composition and projection. Each event e is represented by a circle labeled by $\lambda(e)$. As we do not pay attention to events themselves, they are unnamed. For clarity, we only show the transitive reduction of the partial orders. Let ψ hold only for pairs in $\{(a, a); (c, c); (c, b); (c, d)\}$. The composition of pomsets p_1 and p_2 is shown on Figure 1-a. Added causalities, corresponding to $\leq_\psi$ are depicted by dotted lines. Figure 1-b shows that projections of pomsets composition is, in general, not equal to composition of pomsets projections. Indeed, for $\Sigma_o = \{a, b\}$, $p_4 = \pi_{\Sigma_o}(p_1) \circledcirc \pi_{\Sigma_o}(p_2)$ is not isomorphic to $p_5 = \pi_{\Sigma_o}(p_1 \circledcirc p_2)$.



(a) $p_3 = p_1 \circledcirc p_2$ (b) $p_4 \neq p_5$

Fig. 1. Composition of pomsets

Boxed pomsets. In order to manipulate pomsets with projection and composition more easily, we introduce a new model called *boxed pomsets*. A *port* is the isomorphism class of a subset E of $\mathbb{E}$ where each label appears at most once, i.e for every letter a of Σ, $|\lambda^{-1}(a) \cap E| \leq 1$.

A partial order $\leq$ *plugs* a set of events E_1 to another set of events E_2 when, for every label a in Σ, every event of E_1 labeled by a precedes any event in E_2 labeled by a. More formally, we note $\leq_{E_1 \rightsquigarrow E_2} = \{(e_1, e_2) \in E_1 \times E_2 \mid \lambda(e_1) = \lambda(e_2)\}$ and we say that $\leq$ plugs E_1 to E_2 iff $\leq_{E_1 \rightsquigarrow E_2} \subseteq \leq$.

Definition 1. *A* boxed pomset *is the isomorphism class* $[E^- \uplus E \uplus E^+, \leq]$ *of structures* $(E^- \uplus E \uplus E^+, \leq)$*, where* E^- *and* E^+ *are isomorphic ports called*

respectively input port *and* output port, *E is a set of events called* inside box, *and* $\leq \subseteq (E^- \uplus E) \times (E \uplus E^+)$ *is a partial order relation. Moreover,* $\leq$ *plugs* E^- *to* $(E \uplus E^+)$ *and* $(E^- \uplus E)$ *to* E^+.

A boxed pomset b can be seen as an encapsulated pomset, with an access, for each label, to its maximal and minimal events, through respectively output and input ports. Events which occur before b will only interact with its input port, events which occur after b will only interact with its output port. When needed we will detail the components of boxed pomset b as $[E_b^- \uplus E_b \uplus E_b^+, \leq_b]$. The set of all boxed pomsets is denoted by $\mathbb{B}$. Figure 2 show three examples of boxed pomsets called b_1, b_2 and b_3. They are represented as pomsets in which separate rectangles distinguish clearly input ports, inside boxes and output ports. Input ports will always be located above inside boxes, and output ports below. Note that ports are not real executable events but rather pointers to minimal and maximal events of a pomset. Hence boxed pomset b_3 of Figure 2 and pomset p_3 of Figure 1-a have the same meaning.

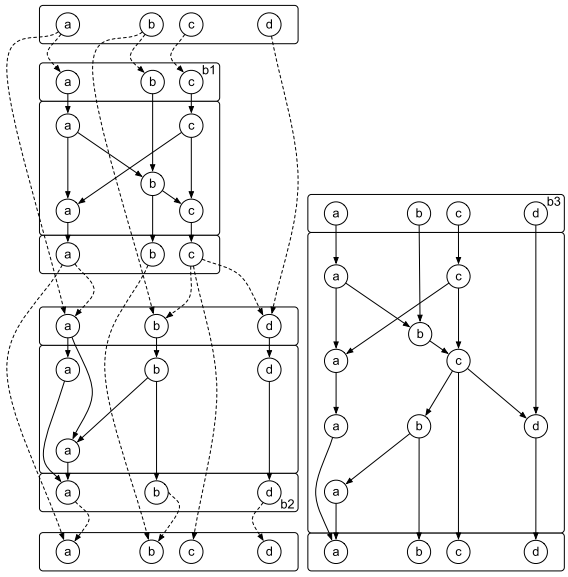


Fig. 2. Boxed pomsets, where $b_1 \boxdot b_2 = b_3$

Definition 2. *A* projection *of a boxed pomset b on an observable alphabet Σ_o is a function $\bar{\pi}_{\Sigma_o} : \mathbb{B} \to \mathbb{B}$ which restricts the inside box of b to events which are labeled by Σ_o, with no modification of the input and output ports, i.e. $\bar{\pi}_{\Sigma_o}(b) = [E_b^- \uplus (E_b \cap E_{\Sigma_o}) \uplus E_b^+, \leq_b \cap (E'_{\Sigma_o})^2]$ where $E_{\Sigma_o} = \lambda^{-1}(\Sigma_o)$ and $E'_{\Sigma_o} = E_b^- \uplus E_{\Sigma_o} \uplus E_b^+$.*

Ports show their usefulness with projections: they are not only labels, but are also used to memorize causal relations with events that may have occurred

before or after a given pomset, that disappear during projection. We extend composition over pomsets to composition over boxed pomsets. This composition does not change the global structure of boxed pomsets: an input port, an inside box, and an output port. Intuitively, the composition of boxed pomsets b_1 and b_2, denoted by $b_1 \boxdot_\psi b_2$ (or simply $b_1 \boxdot b_2$ when ψ is clear from the context), performs the composition of intermediate ports (output port of b_1 and input port of b_2) and keeps the resulting partial order over elements of inside boxes. Input and output ports are used to compute new ports that are respectively the minimal and maximal events of the new object. More formally:

Definition 3. *Let $b_i = [E_i^- \uplus E_i \uplus E_i^+, \leq_i]$ for $i \in \{1,2\}$ be two boxed pomsets, and ψ be a predicate on Σ^2. We define the* composition *of b_1 and b_2 as $b_1 \boxdot_\psi b_2 = [E^-_{1\boxdot 2} \uplus E_{1\boxdot 2} \uplus E^+_{1\boxdot 2}, \leq_{1\boxdot 2}]$, where:*

- *$E^-_{1\boxdot 2}$ is a port such that $\lambda(E^-_{1\boxdot 2}) = \lambda(E_1^-) \cup \lambda(E_2^-)$;*
- *$E_{1\boxdot 2}$ is isomorphic to $E_1 \uplus E_2$;*
- *$E^+_{1\boxdot 2}$ is a port such that $\lambda(E^+_{1\boxdot 2}) = \lambda(E_1^+) \cup \lambda(E_2^+)$;*
- *$\leq_{1\boxdot 2} = (\leq_1 \cup \leq_2 \cup \leq_\psi \cup \leq_{E^-_{1\boxdot 2} \rightsquigarrow (E_1^- \uplus E_2^-)} \cup \leq_{(E_1^+ \uplus E_2^+) \rightsquigarrow E^+_{1\boxdot 2}})^* \cap E^2$ where $E = (E^-_{1\boxdot 2} \uplus E_{1\boxdot 2} \uplus E^+_{1\boxdot 2})$, and $\leq_\psi = \{(e,e') \in E_1^+ \times E_2^- \mid \psi(\lambda(e), \lambda(e'))\}$.*

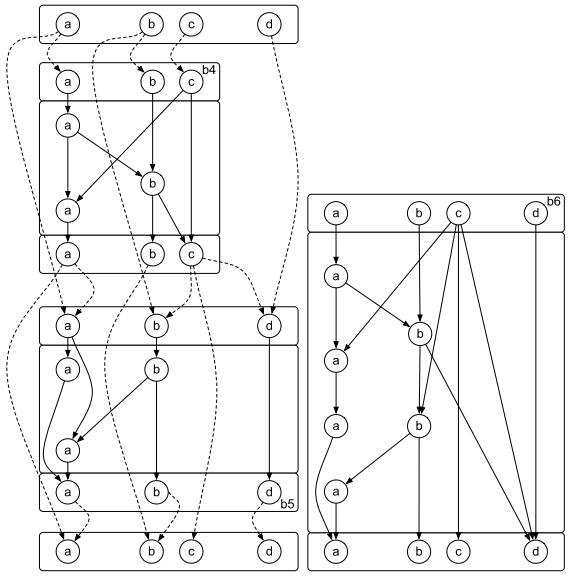


Fig. 3. Projection of boxed pomsets, where $b_4 \circledcirc b_5 = b_6$

Input and output ports of $b_1 \boxdot_\psi b_2$ contain labels of input and output ports of b_1 and b_2, inside box of $b_1 \boxdot b_2$ is isomorphic to the union of insides boxes of E_1 and E_2, and causality relation of $b_1 \boxdot b_2$ is the union of causality relations of b_1 and b_2, augmented with the composition of output port of b_1 with input port of b_2, projected on events of $E^-_{1\boxdot 2} \uplus E_{1\boxdot 2} \uplus E^+_{1\boxdot 2}$. Moreover, we also ensure that $\leq_{1\boxdot 2}$ plugs correctly $E^-_{1\boxdot 2}$, $E_{1\boxdot 2}$ and $E^+_{1\boxdot 2}$. Consider again Figure 2,

and let ψ hold only for $\{(a,a);(c,c);(c,b);(c,d)\}$. Then, boxed pomset b_3 is the composition of boxed pomsets b_1 and b_2. Added causalities, corresponding to $\leq_\psi \cup \leq_{E^-_{1\boxdot 2} \rightsquigarrow (E^-_1 \uplus E^-_2)} \cup \leq_{(E^+_1 \uplus E^+_2) \rightsquigarrow E^+_{1\boxdot 2}}$ are symbolized by dotted lines, and the created ports are symbolized by rectangles located respectively above and below b_4 and b_5.

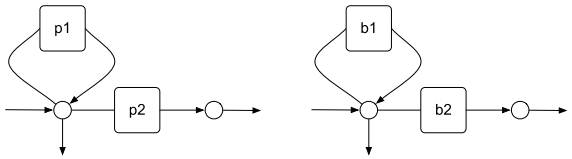
Let us consider the boxed pomsets b_4, b_5 and b_6 of Figure 3 and the examples of Figure 2. In this figure, $b_4 = \bar{\pi}_{\Sigma_o}(b_1)$, $b_5 = \bar{\pi}_{\Sigma_o}(b_2)$ and if we let ψ hold for $\{(a,a),(c,c),(c,b),(c,d)\}$, then we have $b_6 = b_4 \boxdot b_5$. Moreover, we can remark that $b_6 = \bar{\pi}_{\Sigma_o}(b_3)$. Section 3 explains more in detail the relations between pomsets and boxed pomsets, and shows that boxed pomsets have good properties with respect to projections. Hence, it will be easier to manipulate boxed pomsets than pomsets. Thus, we define morphisms to translate problems occurring in pomsets monoid $(\mathbb{P}, \circledcirc)$ to problems in boxed pomsets monoid $(\mathbb{B}, \boxdot)$, which should be solved more easily.

The *boxing operator* $B : \mathbb{P} \to \mathbb{B}$ is used to build a boxed pomset $B(p)$ from a pomset p. The boxed pomset built has an inside box, which corresponds exactly to p, and input and output ports plugged adequately, i.e. input port is plugged to inside box and output port, and inside box is plugged to output port. Thus, $B(p)$ is defined as $B(p) = [E^- \uplus E_p \uplus E^+, (\leq_p \cup \leq_{E^- \rightsquigarrow E_p} \cup \leq_{E_p \rightsquigarrow E^+})^*]$ where E^- and E^+ are ports such that $\lambda(E^-) = \lambda(E_p) = \lambda(E^+)$.

The *unboxing operator* $U : \mathbb{B} \to \mathbb{P}$ is used to extract the inside box from a boxed pomset: $U(b) = [E_b, \leq_b \cap E_b^2]$. Let us consider pomset p_1 from Figure 1, and boxed pomset b_1 from Figure 2. We have $B(p_1) = b_1$, and $U(b_1) = p_1$.

Automata and languages. Single finite pomsets are not sufficient to provide a model for systems that may produce runs of arbitrary size. A good way to design unbounded behaviors is to use an automaton to compose an arbitrary number of pomsets, as in HMSCs. We introduce now classical definitions about automata. For a given set L, a *L-automaton* $\mathcal{A}$ is a tuple $(S, \rightarrow, L, S_0, S_f)$ where S is a set of states, L a collection of labels, $\rightarrow \subseteq S \times L \times S$ a transition relation, S_0 a set of initial states and S_f a set of final states. A *path* ρ of $\mathcal{A}$ is a succession of consecutive transitions of $\mathcal{A}$ such that $\rho = n_0 \xrightarrow{l_1} n_1 \ldots \xrightarrow{l_k} n_k$ and (n_i, l_{i+1}, n_{i+1}) are in $\rightarrow$. An *accepting path* is a path starting with an initial state and ending with a final one. We define $\alpha_*(\rho) = l_1 * \ldots * l_k$, the map which assigns to each path of a L-automaton an element of the monoid $(L, *)$. We extend this definition to L-automaton: $\mathcal{L}_*(\mathcal{A})$ is the language of $\mathcal{A}$, i.e. the set of all elements of $(L, *)$ that $\mathcal{A}$ generates : $\mathcal{L}_*(\mathcal{A}) = \{\alpha_*(\rho) \mid \rho \text{ is an accepting path of } \mathcal{A}\}$. When the composition operator used is not ambiguous, we write α and $\mathcal{L}$ instead of α_* and $\mathcal{L}_*$. For instance, it shall be clear that we use $\circledcirc$ when we manipulate $\mathbb{P}$-automata, and $\boxdot$ when we manipulate $\mathbb{B}$-automata. Figure 4 gives an example of two L-automata. States are represented by circles, labels by rectangles. Initial states have an incoming arrow without source and final states have an outgoing arrow without destination. $\mathcal{A}_1$ is a $\mathbb{P}$-automaton, as p_1 and p_2 (from Figure 1-a) belong to $\mathbb{P}$. $\mathcal{A}_2$ is a $\mathbb{B}$-automaton, as b_1 and b_2 (from Figure 2) belong to $\mathbb{B}$.

We extend operators over $\mathbb{P}$ and $\mathbb{B}$ to operators over $\mathbb{P}$-automata and $\mathbb{B}$-automata. From a map $f : L_1 \to L_2$, we build a new mapping operator $\mathcal{M}_f$:



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$

Fig. 4. Examples of L-automata

L_1-automata $\rightarrow$ L_2-automata, such that, given $\mathcal{A}$ a L_1-automaton, $\mathcal{M}_f(\mathcal{A})$ is the L_2-automaton where each transition $(s, l, s') \in \rightarrow$ is replaced by a transition $(s, f(l), s')$. In this paper, we will mainly consider $\mathcal{M}_B$, $\mathcal{M}_U$, $\mathcal{M}_{\pi_{\Sigma_o}}$ and $\mathcal{M}_{\bar{\pi}_{\Sigma_o}}$ which are, respectively, the conversion of $\mathbb{P}$-automata to their corresponding $\mathbb{B}$-automata that replaces pomsets by boxed pomsets in transitions, the conversion of $\mathbb{B}$-automata to their corresponding $\mathbb{P}$-automata that replaces boxed pomsets in transitions by unboxed ones, the projection of $\mathbb{P}$-automata, that replaces labels of transitions by projected ones, and the projection of $\mathbb{B}$-automata, that replaces labels of transitions by projected boxed pomsets. For the examples of Figure 4 as $B(p_1) = b_1$ and $B(p_2) = b_2$, we have $\mathcal{M}_B(\mathcal{A}_1) = \mathcal{A}_2$, and conversely $\mathcal{A}_1 = \mathcal{M}_U(\mathcal{A}_2)$. Moreover, we can remark that for any automaton and a pair of mappings f and g, we have $\mathcal{M}_f(\mathcal{M}_g(\mathcal{A})) = \mathcal{M}_{fg}(\mathcal{A})$. $\mathbb{P}$-automata and $\mathbb{B}$-automata can not be considered as new models (they are just standard automata over peculiar alphabets). However, the composition laws on pomsets and boxed pomsets gives them more expressive power than simple HMSCs.

Finally, we naturally extend operators over $\mathbb{P}$ and $\mathbb{B}$ to sets of $\mathbb{P}$ and set of $\mathbb{B}$. For instance, we will write $\pi_{\Sigma_o}(\mathcal{L})$ instead of $\{\pi_{\Sigma_o}(p) \mid p \in \mathcal{L}\}$.

3 Properties of Boxed Pomsets

This section introduces the main properties of boxed pomsets. First, we introduce basic properties of the operators defined in Section 2. Then we show several results on pomset languages and their projections. The nice properties of boxed pomsets with respect to projection motivate the use of this new model to answer diagnosis problems, as a natural way to consider partial observation is to work with projected runs of a model. More especially, Theorem 2, gives an automaton construction for the projection of any pomset automaton.

Let us first consider basic properties of B and U operators with respect to projection and composition. We will focus essentially on pomset and boxed pomsets objects, i.e. we will not consider pomset and boxed pomset languages. Proposition 1 below states that the boxing operation is the inverse relation of the unboxing one. Note that as the unboxing operation is not injective, the converse property does not hold.

Proposition 1. *Let p be a pomset. Then $UB(p) = p$.*

The following proposition shows that boxed pomset projection is a kind of dual operation of pomset projection, used with unboxing operator.

Proposition 2. *Let b be a boxed pomset labeled by Σ, and Σ_o be a subset of Σ. Then, $\pi_{\Sigma_o} U(b) = U\bar{\pi}_{\Sigma_o}(b)$.*

Proposition 3 shows that pomset composition and boxed pomset composition are also strongly related. Unlike projections, compositions are not compatible with unboxing operator as in general $U(b_1) \circledcirc U(b_2)$ is not equal to $U(b_1 \boxdot b_2)$. Fortunately boxing operation and compositions work well together:

Proposition 3. *Let p_1 and p_2 be two pomsets. Then $B(p_1 \circledcirc p_2) = B(p_1) \boxdot B(p_2)$.*

The above propositions give us some basic tools to manipulate pomsets and boxed pomsets together with projections. Let us now focus on pomset and boxed pomset languages. It is well known (see for instance Genest et al's paper [5]) that pomset languages generated by automata are not stable under projection: given a $\mathbb{P}$-automaton $\mathcal{A}$, there is, in general, no $\mathbb{P}$-automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$.

Let us now consider the case of boxed pomsets languages. Proposition 4 shows that the boxed pomset projection is distributive over boxed composition law $\boxdot$, i.e. the projection of the composition of two boxed pomsets is exactly the composition of the projection of these boxed pomsets.

Proposition 4. *Let b_1 and b_2 be two boxed pomsets labeled by Σ, and Σ_o be a subset of Σ. Then $\bar{\pi}_{\Sigma_o}(b_1 \boxdot b_2) = (\bar{\pi}_{\Sigma_o}(b_1)) \boxdot (\bar{\pi}_{\Sigma_o}(b_2))$*

This result naturally extends to boxed pomset languages: given a $\mathbb{B}$-automaton $\mathcal{A}$, one can easily find another $\mathbb{B}$-automaton $\mathcal{A}'$ such that the projection of the boxed pomset language generated by $\mathcal{A}$ is exactly the boxed pomset language generated by $\mathcal{A}'$. Theorem 1 below shows that it is sufficient to take $\mathcal{A}' = \mathcal{M}_{\bar{\pi}_{\Sigma_o}} \mathcal{A}$. Hence, computing $\mathcal{A}'$ can be performed in linear time.

Theorem 1. *Let $\mathcal{A}$ be a $\mathbb{B}$-automaton whose events are labeled by Σ, and Σ_o be a subset of Σ. Then $\bar{\pi}_{\Sigma_o}(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A}))$.*

Proof. First, let us take a path ρ in $\mathcal{A}$. Then, we have $\alpha(\rho) = b_1 \boxdot \ldots \boxdot b_n$ where b_i are labels of transition of $\mathcal{A}$. Thus, using Proposition 4, we have $\bar{\pi}_{\Sigma_o}(\alpha(\rho)) = \bar{\pi}_{\Sigma_o}(b_1) \boxdot \ldots \boxdot \bar{\pi}_{\Sigma_o}(b_n)$. Moreover, $\bar{\pi}_{\Sigma_o}(b_1), \ldots, \bar{\pi}_{\Sigma_o}(b_n)$ can be found along a path of $\mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A})$. It means that $\bar{\pi}_{\Sigma_o}(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A}))$.

Second, let us take a path ρ in $\mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A})$. Then, we have $\alpha(\rho) = b_1 \boxdot \ldots \boxdot b_k$, where b_i are labels of transition of $\bar{\pi}_{\Sigma_o}(\mathcal{A})$, i.e. $b_i = \bar{\pi}_{\Sigma_o}(b'_i)$. Thus, using Proposition 4, we have $\alpha(\rho) = \bar{\pi}_{\Sigma_o}(b'_1 \boxdot \ldots \boxdot b'_k)$. Moreover, $b'_1, \ldots, b'_k$ can be found along a path of $\mathcal{A}$. It means that $\mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A})) \subseteq \bar{\pi}_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$. This concludes the proof of Theorem 1.□

Proposition 5 extends the result of Proposition 3 to languages. More precisely, it shows that the boxing operator can be applied equivalently to each label of the initial automaton, or to the resulting language of this automaton.

Proposition 5. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ, and Σ_o a subset of Σ. Then $\mathcal{L}(\mathcal{M}_B(\mathcal{A})) = B(\mathcal{L}(\mathcal{A}))$.*

The following theorem shows that it is possible to keep an automaton-like representation of $\mathbb{P}$-automata projections. The main idea is to consider the dual boxed pomset automaton projection to do so. Roughly speaking, Theorem 2 says that the projection of a $\mathbb{P}$-automaton language is the unboxing of the language of the corresponding $\mathbb{B}$-automaton projection.

Theorem 2. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ, and Σ_o be a subset of Σ. Then $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A})) = U(\mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o}B}(\mathcal{A})))$*

Proof. We will use the above propositions to demonstrate this main result :

$$\begin{aligned}\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A})) &= \pi_{\Sigma_o}UB(\mathcal{L}(\mathcal{A})) && \text{(Prop. 1)} = U\bar{\pi}_{\Sigma_o}(B(\mathcal{L}(\mathcal{A}))) && \text{(Prop. 2)}\\ &= U\bar{\pi}_{\Sigma_o}(\mathcal{L}(\mathcal{M}_B(\mathcal{A}))) && \text{(Prop. 5)} = U(\mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o}B}(\mathcal{A}))) && \text{(Th. 1)}\end{aligned}$$ □

This theorem shows the interest of $\mathbb{B}$-automata, and boxed pomsets. Indeed, for any $\mathbb{P}$-automaton $\mathcal{A}$, there is in general no $\mathbb{P}$-automaton that can generate $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$, but the trivial $\mathbb{B}$-automaton $\mathcal{M}_{\bar{\pi}_{\Sigma_o}B}(\mathcal{A})$ generates a language equivalent to $\pi_{\Sigma_o}(\mathcal{A})$. It seems more convenient for a designer to define the behaviors of a system with $\mathbb{P}$-automata, as one does not have to care for ports. On the other hand, $\mathbb{B}$-automata is a kind of model closed under projection. As trivial transformations allow to switch from one model to another, the framework for diagnosis seems rather clear: models of our systems will be $\mathbb{P}$-automata, and formal manipulations will be performed on $\mathbb{B}$-automata.

4 Event Correlation

$\mathbb{P}$-automata can be used to model distributed system or multi-threaded system, distributed robotics system, business work-flows,... In this paper, we will focus on telecommunication networks. These systems are composed of concurrent agents that react to their environment according to their programmed behavior, and report a part of the events occurring in their neighborhood. These events correspond to a finite subset of all the possible actions that may happen and form the finite collection Σ of event labels. Figure 5 shows a typical architecture for a monitored system. It is very similar to the SNMP architecture: each agent is equipped with a sensor (represented by a diamond), which sends observable events it monitors to the centralized log system (represented by a cylinder). Connection between agents are represented by dotted lines. The log system receives observations and records them in a log file that contains few information about causalities between recorded events.

Note that monitoring systems can not record everything that occurs in a network. The first obvious reason is that the size of log files on disk is necessarily limited, and hence designers have to choose what to record. The second reason is that some actions that one would like to record are performed by hardware, or in a part of the network that is not owned by the company which monitors the network. Hence, only a subset Σ_o of Σ is observable. Furthermore, one can not record all causal relations among events: this needs very intrusive tools, usually based on vector clocks instrumentation [4, 11] which impose a time penalty on communications and again can not be implemented in unobservable places of

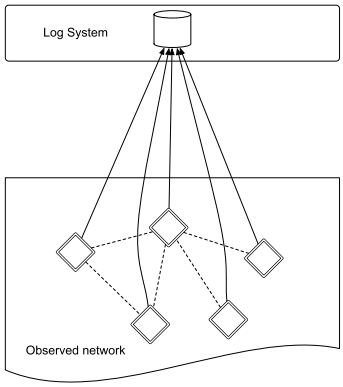


Fig. 5. A monitored system

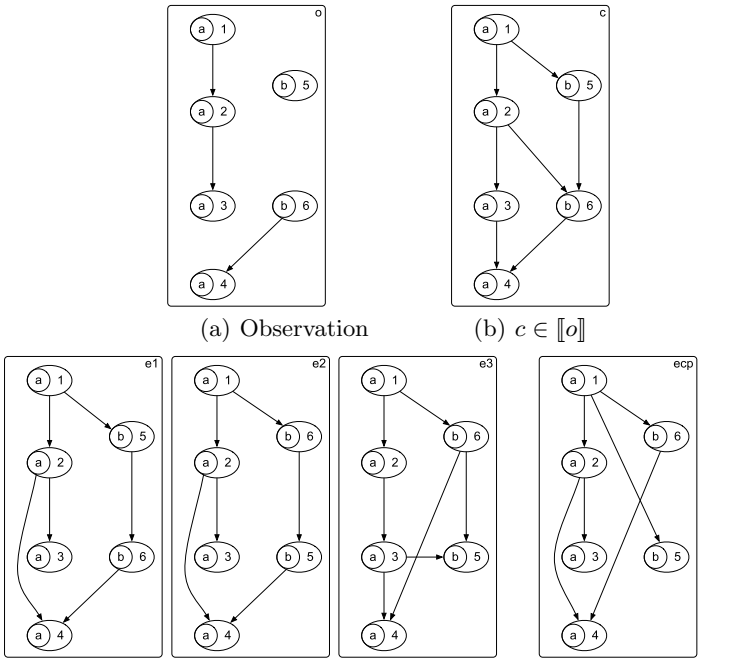
the network. Hence, most of the time logs contain incomplete information about causal relations between recorded events. The log file can then be defined as a lpo $o = (E_o, \leq_o)$, where $\lambda(E_o) \subseteq \Sigma_o$. However, we show in this section how the lost ordering between events can be reconstructed with a model-based approach.

Within this context, the problems we are interested in are *event correlation*, i.e. infer causalities lost by the observation process, and *root causes* elicitation for faulty behavior, i.e. exhibit the minimal observed events with respect to the inferred causal ordering. Note that the log file is in general not sufficient to infer all lost causal relations among observed events. We propose to use some additional information on the behavior of the monitored system. This information is provided by a model given in terms of a $\mathbb{P}$-automaton $\mathcal{A}$. $\mathcal{A}$ represents all the knowledge of experts about the system behaviors, hence $\mathcal{L}(\mathcal{A})$ is supposed to model a significant part of possible runs of the system.

Definition 4 (Explanations). *An* explanation *of an observation given as a lpo $o = (E_o, \leq_o)$ is a lpo $o' = (E_o, \leq)$ such that $\leq_o \subseteq \leq$. The set of all explanations of o is denoted by $[\![o]\!]$. Moreover, given a $\mathbb{P}$-automaton $\mathcal{A}$ the* model-based explanation *of o by $\mathcal{A}$ with observation labels Σ_o, is denoted by $[\![o]\!]_{\Sigma_o,\mathcal{A}}$. $[\![o]\!]_{\Sigma_o,\mathcal{A}}$ is the set of explanations whose isomorphism class belong to the projection of the language generated by $\mathcal{A}$ on Σ_o. More formally, $l \in [\![o]\!]_{\Sigma_o,\mathcal{A}}$ if and only if l is an instance of an element of $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$ and $l \in [\![o]\!]$.*

Note that o and its explanations partially orders the same sets of observed events E_o. Using definition 4, we can formalize the correlation problem as follows:

Definition 5 (Event Correlation Problem). *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ, $o = (E_o, \leq_o)$ be an observation labeled by $\Sigma_o \subseteq \Sigma$, and (e_1, e_2) be a pair of events in E_o^2. The* Event Correlation Problem *for the pair (e_1, e_2), is denoted by $ECP(\Sigma_o, \mathcal{A}, o, e_1, e_2)$ and can be stated as follows: decide whether $e_1 \leq e_2$ for every $p = (E_o, \leq) \in [\![o]\!]_{\Sigma_o,\mathcal{A}}$. We denote by $ecp_{\Sigma_o,\mathcal{A},o}$ the lpo $(E_o, \leq)$ where $e_1 \leq e_2$ if and only if $ECP(\Sigma_o, \mathcal{A}, o, e_1, e_2)$. The set of* root causes *of observation o is denoted by $rc_{\Sigma_o,\mathcal{A},o}$ and is the collection of minimal events with respect to $ecp_{\Sigma_o,\mathcal{A},o}$.*



(a) Observation (b) $c \in [\![o]\!]$

(c) $[\![o]\!]_{\{a,b\},\mathcal{A}_1} = \{e_1, e_2, e_3\}$ (d) $ecp_{\{a,b\},\mathcal{A}_1,o}$

Fig. 6. Observations, explanations and correlations

Intuitively, $ecp_{\Sigma_o,\mathcal{A},o}$ contains all causal orderings that are certain according to the explanations of o provided by $\mathcal{A}$. Figure 6 illustrates these definitions. The graphical representation for lpos is similar to the representation adopted for pomsets, with the slight difference that we associate an unique number to each event to differentiate distinct occurrences of the same action. We consider in this figure that $\Sigma = \{a, b, c, d\}$ and $\Sigma_o = \{a, b\}$. Figure 6-a shows the observation called o. Figure 6-b shows a possible explanation of o called c. Let us suppose that the model of the systems behaviors is the $\mathbb{P}$-automaton $\mathcal{A}_1$ of Figure 4-a. For this automaton, c is not a model-based explanation of o, as c does not belong to $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}_1))$. Lpo's e_1, e_2 and e_3 in Figure 6-c are possible members of $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$ which embed the ordering given by o. As these different explanations do not agree on the respective ordering of events 3 and 4, 3 and 5, 4 and 5, nor 5 and 6, they shall not be ordered in $ecp_{\Sigma_o,\mathcal{A}_1,o}$, as depicted in Figure 6-d. For this case, the root cause of the observation o is event 1 which is labeled by a.

Theorem 3 shows that the ECP problem is hard, but fortunately, Theorem 4 identifies one case where this problem can be solved in polynomial time. Moreover, the constructive proof leads directly to an effective algorithm.

Theorem 3. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ, $o = (E_o, \leq_o)$ be a lpo labeled by $\Sigma_o \subseteq \Sigma$ and (e, e') be in E_o^2. Then $ECP(\Sigma_o, \mathcal{A}, o, e, e')$ is CoNP-complete.*

Proof. We want to show that answering to the following question is NP-complete: Is there a lpo $l = (E_o, \leq) \in [\![o]\!]_{\Sigma_o,\mathcal{A}}$ such that $e \not\leq e'$? This can be proved with

an extension of the proof of Th. 5 in Alur et al's paper [1]. First, let us show that ECP is in NP. A *solution* is a path ρ of $\mathcal{A}$, such that $\pi_{\Sigma_o}(\alpha_{\circledcirc}(\rho)) = [E, \leq]$ is an isomorphism class that contains an explanation of o with $f(e) \not\leq f(e')$, where f is the map which assigns each event class of E to its instance in E_o. Let us consider the $\mathbb{B}$-automaton $\mathcal{A}' = \mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})$. Theorem 2 says that $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A})) = U\mathcal{L}(\mathcal{A}')$. Thus, ρ is also a path in $\mathcal{A}'$ such that $U(\alpha_{\boxdot}(\rho)) = [E, \leq]$ is an isomorphism class that contains an explanation of o with $f(e) \not\leq f(e')$. Let us assume that the size of ρ is greater than $|o||\mathcal{A}|\Sigma|^2$. Then, as ρ should have at most $|o|$ transitions in $\mathcal{A}'$ with observable events, we can find a sequence of unobservable transitions in $\mathcal{A}'$ of size at least $|\mathcal{A}||\Sigma|^2$. That means that an unobservable transition t appears more than $|\Sigma|$ times in $\mathcal{A}'$. As, for any boxed pomset $b_i = [E_i^- \cup E_i^+, \leq_i]$ we have $b_i^{|\Sigma|} = b_i^{|\Sigma|+1}$ and $b_1 \boxdot b_2 = b_2 \boxdot b_1$, we can remove some occurrences of t to build a shorter path ρ' of size bounded by $|o||\mathcal{A}||\Sigma|^2$. Finally, we found ρ' in $\mathcal{A}'$, and thus in $\mathcal{A}$, such that $|\rho|' \leq |o||\mathcal{A}||\Sigma|^2$ and ρ' is a solution. This concludes the NP part.

Second, let us show that ECP is NP-hard. We provide a reduction from the NP-complete problem ONE-IN-THREE-3SAT : given a 3-CNF formula ϕ, is there a satisfying assignment to the variables such that each clause of ϕ gets exactly one literal assigned true ? From a 3-CNF formula $\phi = C_1 \wedge \ldots \wedge C_n$ over variables $x_1 \ldots x_m$, we define a $\mathbb{P}$-automaton $\mathcal{A}$ whose events are labeled by $\Sigma = \{a_i \mid 1 \leq i \leq n\} \cup \{b_j \mid 1 \leq j \leq m\}$. $\mathcal{A}$ has only one state, which is initial and final, and has $2m$ transitions labeled by p_{x_j} and $p_{\bar{x}_j}$, for $1 \leq j \leq m$.

Each p_{x_j} contains an event b_j and an event a_i for each clause C_i where variable x_j appears positively. Similarly, each $p_{\bar{x}_j}$ contains an event b_j and an event a_i for each clause C_i where variable x_j appears negatively. Now, consider the lpo $o = (E_o, \emptyset)$ which contains exactly one event for each possible label, and no causal ordering among these events, and a predicate ψ that returns false to any entry in Σ^2. Moreover, let us simply call e_σ the event of E_o labeled by σ. Thus, for any σ, σ' in Σ, deciding $not(ECP(\Sigma, \mathcal{A}, o, e_\sigma, e_{\sigma'}))$ is equivalent to knowing if there exists $l = (E_o, \leq) \in [\![o]\!]_{\Sigma_o, \mathcal{A}}$ such that $e_\sigma \not\leq e_{\sigma'}$. As all events of o are independent, and as labeling is bijective, solving ECP for o means that there exists a valuation which answers ONE-IN-THREE-3SAT. This concludes the proof.□

Theorem 4. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ and $o = (E_o, \leq_o)$ be a lpo labeled by $\Sigma_o \subseteq \Sigma$. If o has no auto-concurrency and Σ is fixed, then for every $(e, e') \in E_o^2$, $ECP(\Sigma_o, \mathcal{A}, o, e, e')$ is NLOGSPACE. More precisely, ECP can be solved in $O(|\mathcal{A}||o|^{|\Sigma||\Sigma_o|})$.*

Proof. Let us show that the ECP problem can be translated into finding accessible states of an automaton of size $O(|\mathcal{A}||o|^{|\Sigma||\Sigma_o|})$. We will use lpos instead of pomsets when we need to recall the identity of events, which is the case for ECP. Of course, the operations and mappings defined for pomsets and boxed pomsets extend to lpos and to boxed lpos.

To complete the proof, we need to define the notion of unfolding for boxes lpos. The *unfolding* of a boxed lpo b, denoted by $\mathcal{U}_b$, is a finite boxed lpo automaton, i.e. an automaton labeled by boxed lpos. $\mathcal{U}_b = (S, \rightarrow, \mathcal{B}, s_0, s_f)$, where:

- S is the set of prefixes of $B(l)$, i.e. the set of boxed lpos $\{b' \mid \exists b'',\ b = b' \boxdot b''\}$. Note that prefixes depend also on relation ψ used for composition;
- $\mathcal{B}$ is a set of boxed lpos that are used by the transition relation;
- $s_0 = \epsilon_{\mathbb{B}}$ is the initial state, and $s_f = b$ is the final state;
- $s_1 \xrightarrow{b'} s_2$ iff $s_1 \boxdot b'$ is an explanation of s_2, i.e. if they have the same event set and $\leq_{s_2} \subseteq \leq_{s_1 \boxdot b'}$.

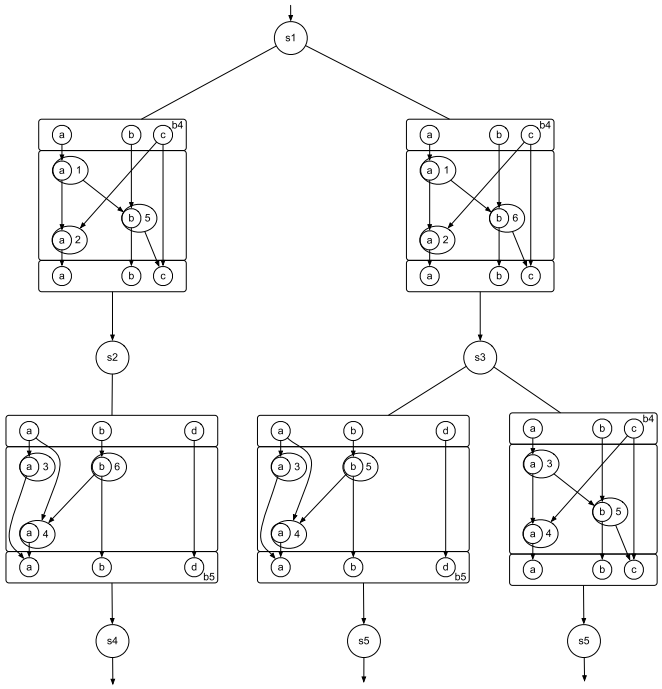


Fig. 7. Unfolding and synchronization

Note that for a boxed lpo in our unfolding, all labels in ports are not useful. For instance, consider boxed lpo b_5 in Figure 7: label d does not have to appear in the ports of this lpo as it is not connected to an event of the inside box of b_5. Hence, we can define a smaller set of labels $\mathcal{B}$ for our unfolding by considering only boxed lpo b whose ports are defined over labels that are connected to the inside box of b.

Lemma 1. *Let l be a lpo labeled by Σ_o. Then $U(\mathcal{L}(\mathcal{U}_{B(l)})) = [\![l]\!]$. If l has no auto-concurrency then the number of states of $\mathcal{U}_l$ is bounded by $|l|^{|\Sigma||\Sigma_o|}$.*

Proof. ($\Rightarrow$) If $l' \in U(\mathcal{L}(\mathcal{U}_{B(l)}))$ then there exists $b' = b'_1 \boxdot \ldots \boxdot b'_k$ such that $U(b') = l'$. Let us make an induction on k. First, let us assume that $k = 1$. Then, as b' is an explanation of $B(l)$, $U(b') = l'$ is an explanation of $UB(l) = l$ (Prop. 1). Let us then have $b'_1 \boxdot \ldots \boxdot b'_{k-1}$ an explanation of b_{k-1}, the state reached after reading $b'_1 \ldots b'_{k-1}$. By definition of $\mathcal{U}_{B(l)}$, $b_{k-1} \boxdot b'_k$ is an explanation of

b_k. Let us remind that we manipulate only boxed lpos with minimal number of events in ports. Thus $E_{b_{k-1}} = E_{b'_1 \boxdot \ldots \boxdot b'_{k-1}}$ and $\leq_{b_{k-1}} \subseteq \leq_{b'_1 \boxdot \ldots \boxdot b'_{k-1}}$, and we obtain that $b'_1 \boxdot \ldots \boxdot b'_k$ is an explanation of b_k. ($\Leftarrow$) If $l' \in [\![l]\!]$ then l' is an explanation of l and $B(l')$ is a possible transition in $\mathcal{U}_{B(l)}$ from s_0 to s_f.

About the complexity statement : If we want to count the number of possible prefix of a boxed lpo, the simplest way is to consider that each event in output ports records a prefix of the boxed lpo. Moreover, if b has no auto-concurrency, it suffices to record only one event for each observable label - corresponding to the maximal observed event for this label - in order to record a prefix. □

The *product* of a boxed lpo automaton $\mathcal{A}_1$ and a $\mathbb{B}$-automaton $\mathcal{A}_2$, denoted by $\mathcal{A}_1 \times \mathcal{A}_2$ is the boxed lpo automaton resulting on the cartesian product of states of these automata and such that $(s_1, s_2) \xrightarrow{l} (s'_1, s'_2)$ iff $s_1 \xrightarrow{l} s'_1$ and $s_2 \xrightarrow{b} s'_2$, with l an instance of the boxed pomset b.

Lemma 2. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ and l be a lpo labeled by $\Sigma_o \subseteq \Sigma$. Then $U(\mathcal{L}(\mathcal{U}_{B(l)} \times \mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A}))) = [\![l]\!]_{\Sigma_o, \mathcal{A}}$. If l has no auto-concurrency then the number of states is bounded by $|\mathcal{A}||l|^{|\Sigma||\Sigma_o|}$.*

Proof. $l' \in U(\mathcal{L}(\mathcal{U}_{B(l)} \times \mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})))$ is equivalent to $l' \in U(\mathcal{L}(\mathcal{U}_{B(l)}))$ and l' is an instance of an element of $U(\mathcal{L}(\mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})))$, using definition of $\times$. Moreover, this is also equivalent to $l' \in [\![l]\!]$ (Lemma 1) and l' is an instance of an element of $\pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$ (Theorem 2). This is also equivalent, by definition, to $l' \in [\![l]\!]_{\Sigma_o, \mathcal{A}}$. Complexity comes from Lemma 1 and the definition of $\times$.□

Corollary 1. *Let $\mathcal{A}$ be a $\mathbb{P}$-automaton whose events are labeled by Σ and $o = (E_o, \leq_o)$ be a lpo labeled by $\Sigma_o \subseteq \Sigma$.*

Then $ecp_{\Sigma_o, \mathcal{A}, o} = (E_o, (\bigcap_{(E_o, \leq_l) \in L} \leq_l)^*)$, where $L = U(\mathcal{L}(\mathcal{U}_{B(l)} \times \mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})))$.

The constructive proof of Theorem 4 and its corollary immediately provide an algorithm to find $ecp_{\Sigma_o, \mathcal{A}, o}$ for a given observation o and a model $\mathcal{A}$. The first step is to compute $\mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})$. The second step is to compute the product of the unfolding of $B(o)$ with $\mathcal{M}_{\bar{\pi}_{\Sigma_o} B}(\mathcal{A})$, and restrict this product to accessible states. Each accepting path of the product generates an explanation for o, and $ecp_{\Sigma_o, \mathcal{A}, o}$ can then be obtained by intersecting the orders given by these explanations. Figure 7 shows an example of synchronization $\mathcal{U}_{B(o)} \times \mathcal{M}_{\bar{\pi}_{\Sigma_o}}(\mathcal{A}_2)$, where o is the lpo of Figure 6-a, and $\mathcal{A}_2$ is the automaton of Figure 4-b. States of this product are boxed lpos which are prefixes of $B(o)$. Path $\rho_1 = s_1 \xrightarrow{b_4} s_2 \xrightarrow{b_5} s_4$ corresponds to e_1 (as $U\alpha_{\boxdot}(\rho_1) = e_1$), path $\rho_2 = s_1 \xrightarrow{b_4} s_3 \xrightarrow{b_5} s_5$ corresponds to e_2 and path $\rho_2 = s_1 \xrightarrow{b_4} s_3 \xrightarrow{b_4} s_6$ corresponds to e_3, where e_1, e_2, and e_3 are the model-based explanations of o given in Figure 6-c.

Theorem 4 explicitly rules out observations with autoconcurrency. This restriction is only due to complexity reasons, as considering autoconcurrency would make our algorithm exponential in the size of the observation rather than in the size of the observed labels. This should not be considered as a severe limitation of the approach, as these requirements are naturally met in an observation

framework where each sensor produces different observed labels (for instance by tagging an action name with a unique identity), and where local sequential ordering on each sensor is not lost during communication to the log system.

5 Conclusion

We have shown how to perform event correlation from an observation with a partial order model. This work opens two perspectives. The first one is to distribute computations as proposed previously [8]. Indeed, we know that the complexity of correlation is in $O(|\mathcal{A}||o|^{|\Sigma||\Sigma_o|})$. We can define distributed monitoring architectures, where local log systems observe only a subset of the network. Within this kind of architecture, Σ_o is partitioned into subsets of observable actions. Event correlation can be performed in parallel by each local log system with lower complexity. The main challenge is then to combine the local results to obtain a global view. The second perspective is to look at probabilistic models. So far, we can only answer whether a causal relation among some events is sure or not. It may be interesting to have a more qualitative answer, given as a probability.

References

1. R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In *Proc. of ICALP'01*, number 2076 in LNCS, pages 797–808, 2001.
2. A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, 2003.
3. C. Dousson and V.D. Thang. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proc. of IJCAI'99*, pages 620–626, 1999.
4. C. Fidge. Logical time in distributed computing systems. *IEEE Computer*, 24(8):28–33, 1991.
5. B. Genest, L. Hélouët, and A. Muscholl. High-level message sequence charts projection. In *Proc. of CONCUR'03*, number 2761 in LNCS, pages 308–322, 2003.
6. J.L. Gischer. The equational theory of pomsets. *TCS*, 61(2-3):199–224, 1988.
7. IETF Network Working Group. A simple network management protocol (snmp). Technical report, IETF, 1990.
8. L. Hélouët, T. Gazagnaire, and B. Genest. Diagnosis from scenarios. In *WODES'06*, 2006.
9. ITU-TS. *Recommendation Z.120: Message Sequence Chart (MSC)*. 2004.
10. T. Jéron, H. Marchand, S. Pinchinat, and M.O. Cordier. Supervision patterns in discrete event systems diagnosis. In *WODES'06*, 2006.
11. F. Mattern. Virtual time and global states of distributed systems. In *Workshop on Parallel and Distributed Algorithms*, 1989.
12. Y.A. Nygate. Event correlation using rule and object based techniques. In *Proc. of the 4^{th} Integrated network Management*, pages 278–289, 1995.
13. V. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.
14. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.