Unfoldings of Networks of Automata and their Application in Supervision

Bartosz Grabiec
INRIA, IRISA
Campus de Beaulieu, F-35042 Rennes, France
Bartosz.Grabiec@irisa.fr

Claude Jard
ENS Cachan Bretagne, IRISA
Université Européenne de Bretagne,
Campus de Ker-Lann, F-35170 Bruz,
France
Claude.Jard@bretagne.ens-cachan.fr

ABSTRACT

In this article we present techniques of unfoldings of networks of automata. This type of techniques allows capturing the causal relations of partial order between events of a model of a distributed system. They are particularly tailored to address the issue of supervision. Given a sequence of actions observed during an execution of a distributed application it is possible, using a model of the system, to determine transitions of the model that produced these actions and to infer causal relationships or independence between events. We also introduce a method of partial observations into unfoldings. Using this method we can state some actions or transitions as unobservable and under some termination constraints we can infer possible explanations.

We explore unfoldings in the original context of supervision systems which are modeled by a network of timed automata and we show how the possible dates of occurrences of actions can be inferred by using symbolic constraints.

1. INTRODUCTION

The notion of unfolding used to model behaviours of distributed systems [?] was introduced to equip these models with a semantics known as *partial order* [?, ?]. In contrast to the usual sequential semantics given in terms of transition systems [?] and traditionally used in verification activities, control and tests, the semantics of partial order defines executions as partial orders on events. It gives a possibility to place in an execution the causal dependencies between events and additionally it can identify creation of independent or concurrent events. Superposition of sets of events for all possible executions may constitute what is called an *event structure* [?] or an unfolding. An event structure is a set of events structured by a partial order relation and a relation of conflict which allows finding valid executions. This notion was essentially developed in the context of Petri nets [?]. Here we present it in the context of networks of automata [?, ?].

The question of supervision is a practical problem which arises when we want to monitor on line activity of a real system and collegiate events that are produced by the system [?, ?]. For this purpose, usually some probes are placed in the system and they produce events which carry useful information for supervisor. These events are then transmitted (often over a network in the case of distributed systems) to be processed by a module called the supervisor. It is assumed that the order of arrival of events to the supervisor is not significant since it is in general the result of an asynchronous process of collection. The supervisor must then rebuild the dependences which can not be found directly using the local information associated with the events. For this reason, we consider a model-based approach in which the supervisor uses a model of the system. This model comprises of transitions labelled by information which can be observed.

Our approach is summarized in Figure 1. The sequence of observations is a series of symbols. The supervisor produces a set of explanations since several trajectories of the model can produce the same set of symbols. An explanation is a partial order of events. An event corresponds to the execution of a transition in the model. Such a transition is labeled by the symbol that is observed and which should be then explained. The supervisor therefore infers from the model possible dependencies between observed symbols. We extend the approach by adding time constraints [?] to the model (the model of networks of timed automata), which makes it possible moreover to infer the dates of production of the observed symbols.

In our previous works [?, ?, ?] we have developed a supervision approach based on unfoldings of Petri nets [?], mainly in the untimed framework [?], with a particular motivation for the distribution of calculations used in the supervision [?], and more recently by tackling the questions of symbolic constraints [?, ?].

One of the main new issues presented in the article is a unified way of processing the model of networks of timed automata [?]. The explicit use of networks of automata is original and quite different from the usual approach in which we first calculate the corresponding single automaton representing all the components before proceeding to an analysis. The algorithmic complexity is higher but it is, however, highly compensated by the more compact (in terms of memory size) representation as we do not calculate the set of interleavings of actions which are parallel. Above all it is a natural way to explain the causal relationships that we

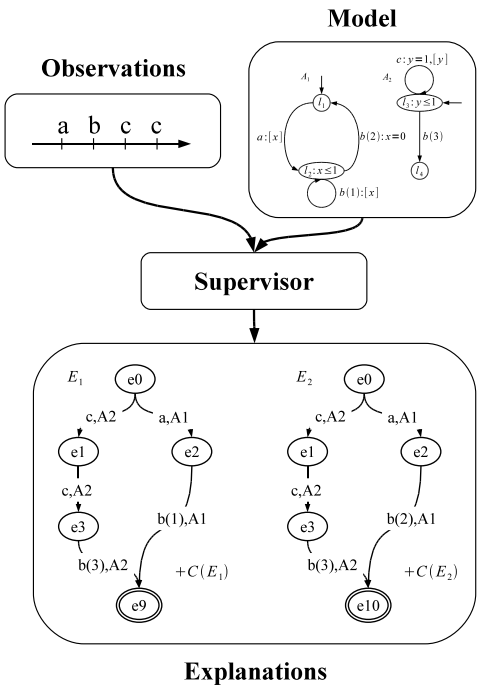

Figure 1: Schema of the supervision system.

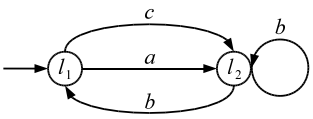

Figure 2: A finite state automaton. l_1 is the initial state, $\Sigma = \{a, b, c\}$.

believe are central to the work of supervision and diagnosis.

The rest of the article is structured as follows. We start with a presentation of the concept of unfolding for the well-known model of the finite state automaton. Then we present the unfoldings for networks of finite state automata. Next we describe the question of supervision and a method which allows producing explanations of a sequence of observations for model without time constraints. In the next parts the timed model is introduced and a method for producing timed explanations is described.

2. FINITE AUTOMATA

Definition 1. (Finite automaton) *A finite state automaton A is defined by a tuple $A = (Q, q_0, \Sigma, T)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is the alphabet (a finite set of symbols) and $T \subseteq Q \times \Sigma \times Q$ is the set of transitions (or edges).*

An example of graphic depiction of a finite state automaton is given in Figure 2.

All executions of such an automaton can be represented as labeled paths in a tree graph with the initial state as the root. They can be formally defined with the use of notion of unfolding. A transition is denoted by $t = (\alpha(t), \lambda(t), \beta(t))$.

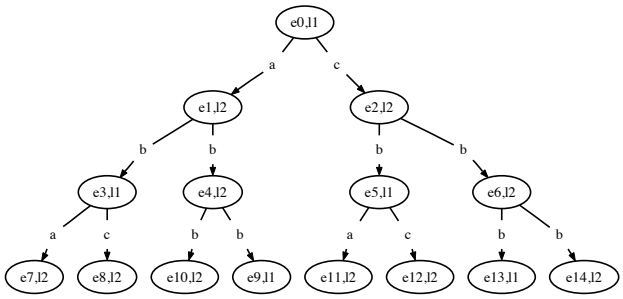

Figure 3: A prefix of the unfolding of the automaton in Figure 2.

Unfolding of an automaton A, denoted by $\mathcal{U}(A)$ is given by a set of events. Each event represents an occurrence of a transition. It is defined by a pair $e = (\pi(e), \tau(e))$, where $\pi(e)$ is the event which precedes e in the unfolding, and $\tau(e)$ is the transition assigned to e. There is also a fictitious initial event $\perp$ for which $\beta(\tau(\perp)) = q_0$.

$\mathcal{U}(A)$ can be defined inductively as follows.

Definition 2. (Unfolding of a single automaton) *Given a finite state automaton $A = (Q, q_0, \Sigma, T)$, the unfolding of A, denoted by $\mathcal{U}(A)$, is the smallest set such that:*

- $\perp \in \mathcal{U}(A)$,
- $\{\exists \pi(e) \in \mathcal{U}(A) \wedge \exists t \in T \wedge \alpha(t) = \beta(\tau(e))\}$
  $\implies (e, t) \in \mathcal{U}(A)$

Having two events e and e' of $\mathcal{U}(A)$, e immediately precedes e' (denoted by $e \rightarrow e'$) if $\pi(e') = e$. Causality between two events is defined as a reflexive and transitive closure of the relation $\rightarrow$ (denoted by $\rightarrow^*$). For an event e, its set of causal predecessors is denoted by $\downarrow e = \{f \mid f \rightarrow^* e\}$. This notation is extended to sets: $\downarrow E = \bigcup_{e \in E} \downarrow e$.

In general unfoldings are infinite sets e.g. unfoldings of automata with loops. In Figure 3 there is a finite subset of the unfolding of the automaton in Figure 2. The subset is closed by precedence relation and is called a prefix of the unfolding. Graphically, an event $(\pi(e), \tau(e))$ is represented by an arc labeled by $\lambda(\tau(e))$ which starts from the node $\pi(e)$ and is ended by the node e. The events are drawn as ellipses which contain the name of the event and the reached state.

The prefixes of unfoldings of finite state automata are trees with bounded degree.

3. NETWORK OF AUTOMATA

We use the similar approach as in the previous section to define the notion of unfolding of a network of automata.

Definition 3. (Network of finite state automata) *A network of finite state automata is a set $\{A_1, ..., A_n\}$ of n finite state automata with $A_i = (Q_i, q_{0i}, \Sigma_i, T_i)$. A* global state *of the network is a tuple of $Q_1 \times Q_2 ... \times Q_n$. The initial global state is $(q_{01}, ..., q_{0n})$. The activity of the automata is synchronized on transitions having the same label. We define the set of synchronizations Sync as the set of $(t_1, ..., t_n) \in (T_1 \cup \{\epsilon\}) \times ... \times (T_n \cup \{\epsilon\})$ such that $(t_1, ..., t_n) \neq (\epsilon, ..., \epsilon)$ and there exists a such that $\forall t_i \neq \epsilon . \lambda(t_i) = a \wedge \forall t_i = \epsilon . a \notin \Sigma_i$.*

An example of such a network is given in Figure 4.

As in the case of a single automaton, unfolding of a network N, denoted by $\mathcal{U}(N)$ is given as a set of events. An event is a vector $e = (e_1, \ldots, e_n)$, where $e_i = (\pi_i(e), \tau_i(e))$ in which $\pi_i(e)$ denotes the predecessor of e considering the automaton A_i, and $(\tau_1(e), \ldots, \tau_n(e)) \in Sync$. In the case where the automaton A_i is not considered by the transition we define $\pi_i(e) = \epsilon$.

Given two events e and e', e precedes immediately e' in the automaton A_i (denoted by $e \rightarrow_i e'$) if $\pi_i(e') = e$. A set of events is in conflict iff $\exists e, e' \in E, i \in [1, n]$ such that $\pi_i(e) = \pi_i(e')$. From the causality point of view none of the events can have conflicts in its causal history as it would mean that there are two exclusive events. Such events are the result of a local choice of a single automaton.

Definition 4. (Unfolding of a network) *Given a network N, $\mathcal{U}(N)$ is the smallest set satisfying:*

- $\perp \in \mathcal{U}(N)$
- $$\left\{\begin{array}{l} (\tau_1(e), ..., \tau_n(e)) \in Sync \\ \forall i \in [1, n] \left\{\begin{array}{l} \tau_i(e) = \epsilon \Rightarrow \pi_i(e) = \epsilon \\ \tau_i(e) \neq \epsilon \Rightarrow \left\{\begin{array}{l} \pi_i(e) \in \mathcal{U}(N) \\ \alpha(\tau_i(e)) = \beta(\tau_i(\pi_i(e))) \end{array}\right. \end{array}\right. \\ \downarrow e \text{ is conflict free} \end{array}\right.$$
  $$\Longrightarrow e \in \mathcal{U}(N)$$

Figure 4 presents an example of a prefix of unfolding of the network figured on the left.

Graphically, an event e is represented by an arc going from the node $\pi_i(e)$ to the node e, and labeled by $\langle \lambda(\tau_i(e)), A_i \rangle$. Each node is represented by an ellipse, labeled with the name of an event e and the local states reached by the corresponding transition.

In general the prefixes of the unfoldings of the networks are acyclic graphs with an unbounded degree. The inductive definition can be directly used to construct an algorithm in which the events are placed one by one in the unfolding if they are not already there.

4. SUPERVISION WITH NETWORK OF AUTOMATA

4.1 Guided unfolding

Having the notion of unfolding, the question of supervision can be addressed. We consider a sequence of observations $\sigma \in \Sigma^*$. The problem is to construct all possible executions of a network which are correct with respect to the observation. In other words we search for the explanations of what is observed. The sequence of observations is finite, and so is the unfolding.

The idea is that the order given by the sequence of observations σ does not necessarily correspond to the real order of the corresponding events. The observation is just the result of the observation process in the distributed system that is be observed. It is the role of the supervisor, using the model, to propose the possible causal relationship between observations. It is clear that the actions of the same type are totally ordered as they correspond to a local action of an automaton or a synchronization. A good method to guide a construction of an unfolding is to use the Parrikh function of the sequence σ. The Parrikh function $\varpi : \Sigma^* \rightarrow \mathbb{N}^{|\Sigma|}$ counts the number of occurrences of each symbol of the sequence. We extend the information of an event e by the Parrikh vector $\varsigma(e)$ of the sequence recognized by the set of causal predecessors of e. By comparison of the Parrikh vector of an event with the Parrikh vector of an observation, we ensure that the latter does not exceed the former. For an action $a \in \Sigma$, we denote by χ_a the Parrikh vector which has all its components set to 0 except the one which corresponds to the action a and equals 1. The events are denoted by $e = \left((\pi_i, \tau_i)_{i \in [1,n]}, \varsigma(e)\right)$.

The unfolding guided by the observation, denoted as $\mathcal{E}(N, \sigma)$ may be therefore defined as:

Definition 5. (Supervision of a network) *Given a network N and a sequence of observations σ, $\varepsilon(N, \sigma)$ is the smallest set satisfying:*

- $\perp \in \mathcal{E}(N, \sigma)$ *with* $\varsigma(\perp) = \mathbf{0}$
- $$\left\{\begin{array}{l} t = (\tau_1(e), ..., \tau_n(e)) \in Sync \\ \forall i \in [1, n] \left\{\begin{array}{l} \tau_i(e) = \epsilon \Rightarrow \pi_i(e) = \epsilon \\ \tau_i(e) \neq \epsilon \Rightarrow \left\{\begin{array}{l} \pi_i(e) \in \mathcal{E}(N) \\ \alpha(\tau_i(e)) = \beta(\tau_i(\pi_i(e))) \end{array}\right. \end{array}\right. \\ \downarrow e \text{ is conflict free} \\ \varsigma(e) = \sum_{f \in \downarrow e} \chi_{\lambda(t)} \leq \varpi(\sigma) \end{array}\right.$$
  $$\Longrightarrow e \in \mathcal{E}(N, \sigma)$$

Figure 5 shows supervision obtained for the sequence of observations *acbc*. The result of this supervision actually represents the two following explanations: the actions a and two c are carried out independently and then the action b is fired. In the case of the action b there are two possibilities as there are two actions labeled by b in the first automaton. These explanations are obtained by:

- extracting executions of the unfolding. An execution is defined by a subset of E of the unfolding $\mathcal{E}(N, \sigma)$ which is closed by causality ($\downarrow E = E$) and without conflict;
- requiring that the executions explain the whole sequence of observations i.e. $\varsigma(e) = \varpi(\sigma)$.

Not only we infer the trajectories of automata but also the possible causal links between observations. This is what gives a real meaning of the method in the context of supervision, thanks to which

we get the extra information to the sequence of observations.

4.2 Partial observations

In many cases we cannot or we do not want to observe all of transitions of the model. For this reason we introduce a construction of an unfolding using only a sequence of partial observations. To construct an such an unfolding we assume that we observe only some chosen transitions of the network. They can be any transitions of the model. We can observe both some local transitions or some global transitions which involve several automata.

For the construction of an unfolding based on some partial observation σ we use a slightly modified technique which we used in the section 4.1. To mark unobservable transitions we use a boolean function $\nu(x)$, where x is an event or a transition. An event e is observable (i.e. $\nu(e)$ is true) if at least

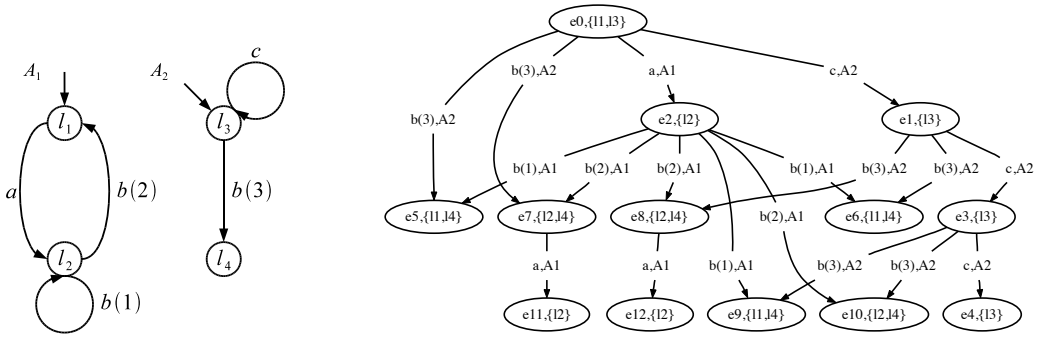

Figure 4: A network of finite state automata on the left. $Sync = \{(a, \epsilon), (b(1), b(3)), (c, \epsilon), (b(2), b(3))\}$**. A prefix (on the right) of the unfolding of the network. The initial event** $\perp$ **is represented as** e_0 **in the picture. Inside each ellipse which denotes an event apart from its name there is a set of locations which was reached after firing the corresponding transitions of an event. For example for the event** e_6 **there are two local transitions** $b(1)$ **and** $b(3)$ **which take part in the action** b**. After this action the network reaches the locations** l_1 **and** l_4**.**

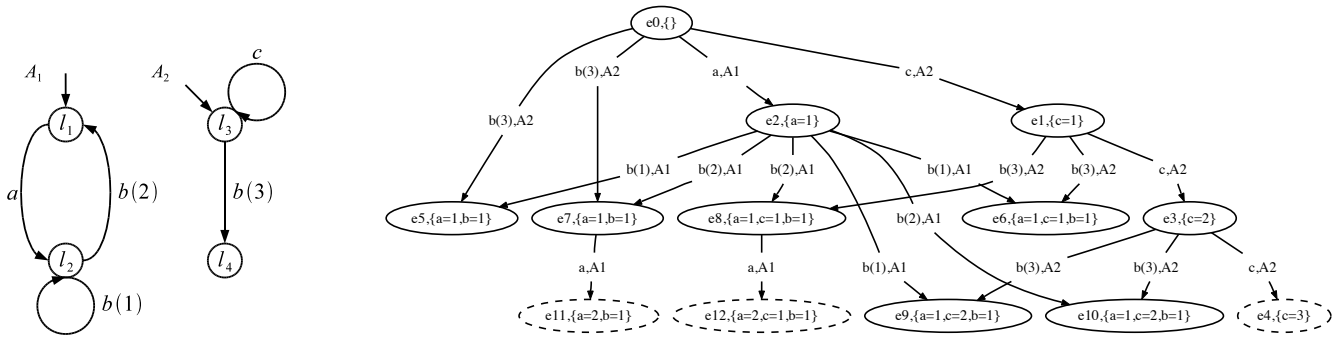

Figure 5: Network of automata (on the left) and its unfolding (on the right) constructed on the basis of the sequence of observations $\sigma = acbc$**. The vector** $\varsigma(e)$ **is shown next to each event. For the convenience of understanding it is represented as a set of equalities of the form** $x = y$**, where** x **stands for the number of transitions labeled by "x" and** y **is the corresponding value. In the picture we distinguished using dashed lines three events** e_4, e_{11}, e_{12} **for which** $\varsigma(e) \not\leq \varpi(\sigma)$**, where** $(e \in \{e_4, e_{11}, e_{12}\})$**. For example for** e_{12} **the number of transitions labeled by** a **is greater than** 1 **which is the case in** σ**. In the picture** $\perp \equiv e_0$**.**

one of its underlying transitions is observable. Formally we can write it as follows: $\nu(e) \iff (\exists \tau_i \in \tau(e), \nu(\tau_i)) \vee \nu(\tau(e))$. Let us notice that in this definition we distinguished two situations: one when a local transition is observable or not ($(\exists \tau_i \in \tau(e), \nu(\tau_i))$ is true or false), and the other one when the global transition is observable or not ($\nu(\tau(e))$ is true or false). We have made a distinction in order to use the modified version of the method with the Parrikh vector. Namely, in the vector we put an extra position which is a number of events which cannot be observed during the considered execution. Thus during construction of the unfolding every time an event which is not observable is produced the value of the mentioned position is increased. On the contrary whenever there is an observable event e we have to increase the value which corresponds to the transition $\tau(e)$. However, in the latter case we may not observe the whole global transition if it consists of several local transitions and some of them are not observable. Then we increase only the values of the vector which are assigned to the local observable transitions. This way when we observe a whole global transition (i.e. all its underlying local transition), only one position of the vector is modified. Such approach let us easily to compare a sequence of observations and a set of events produced by an execution in the unfolding.

It is important to mention that one of the main problems when we want to infer some information about a system which is only partially observable is that there may be some infinite loops which are unobservable. For this reason in our solution we bound for each possible execution E of the system the number of unobservable events i.e. $|\{e \in E \mid \neg\nu(e)\}| \leq M$.

In Figure 6 we present an example of the unfolding of the system in Figure 4. In the example we observe two transitions: a and $b(3)$. It is worth noticing that we do not observe the whole action b but just one of the local transitions which take part in it. Thus we do not care and we cannot distinguish which of the two possible synchronizations takes place. Let us take one of the successful events (i.e. a maximal event of a valid explanation) from Figure 6 e.g. e_{12}. As an input we have a sequence of observations aba. We can notice that among predecessors of e_9 there are two observable events $\{e_2, e_9\}$ (we do not count the initial event $e_0 \equiv \perp$) and two unobservable events $\{e_1, e_3\}$. When we take all the transitions of the events $\{e_1, e_2, e_3, e_9, e_{12}\}$ we get the vector in which we have two unobservable events (in the picture $? = 2$), two transitions a, and one $b(3)$.

5. TIMED AUTOMATON

Currently we will explore an extended version of the previous models. Namely timed automata which have additionally time constraints. This allows us to restrict the behavior of the underlying model which is without time. Thus we add to the underlying finite automaton clocks, invariants on the states, guards and resets on transitions [**?**].

Definition 6. (Timed automaton) *A timed automaton A is a tuple $(Q, q_0, \Sigma, X, T, Inv)$ where*

- Q is a finite set of states, $q_0 \in Q$ the initial state;
- Σ is a finite alphabet of actions;
- X is a finite set of clocks;

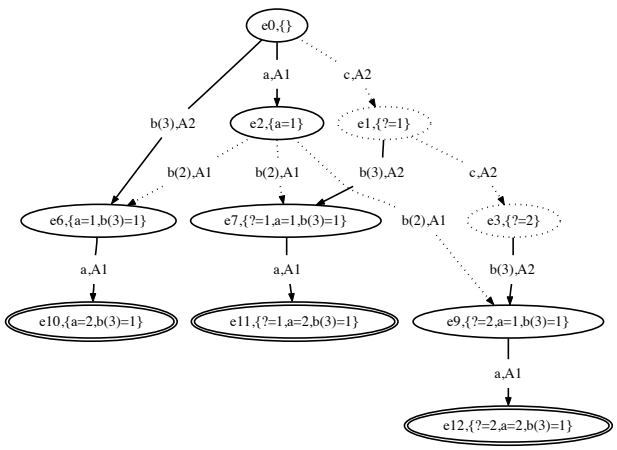

Figure 6: A prefix of unfolding of the network in Figure 4 guided by the partial observation aba. For the construction an extra parameter was used which is the maximal number of unobservable events and is equal to 2. The description of events is the same as in Figure 5. ? denotes the number of unobservable events in the set of predecessors of the considered event. The dotted ellipses and arrows denote events and transitions which are unobservable. The events e_{10}, e_{11}, e_{12} are all possible explanations for the given sequence of observations.

- $T \subseteq Q \times \mathcal{C}(X) \times \Sigma \times 2^X \times Q$ is a finite set of transitions. $t = \langle \alpha(t), \gamma(t), \lambda(t), \rho(t), \beta(t) \rangle$ represents a transition from the state $\alpha(t)$ to the state $\beta(t)$, labeled by the action $\lambda(t)$, with guard $\gamma(t)$ and a set of resets $\rho(t)$. $\mathcal{C}(X)$ is a set of conjunctions of constraints of the form $x \bowtie c$, where $x \in X$, $c \in \mathbb{R}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. $\mathcal{C}_<(X)$ is a set of conjunctions of constraints of the form $x < c$ or $x \leq c$;
- $I : Q \to \mathcal{C}_<(X)$ assigns an invariant to any state.

In the similar manner to the previous definition of the network we can define a network of timed automata. Thus we consider each synchronization as a tuple with transitions which share an action.

Definition 7. (Network of timed automata) *A network of timed automata (NTA) is a set $\mathcal{N} = \{A_1, \ldots, A_n\}$ of n timed automata with $A_i = (Q_i, q_{0i}, \Sigma_i, T_i, X_i, I_i)$. We make the assumption that clocks are not shared between automata ($\forall A_i, A_j \in \mathcal{N}, X_i \cap X_j \neq \emptyset$). The set of synchronizations Sync is defined as in the untimed case.*

Figure 7 presents an example of NTA.

The operational semantics of such a system is defined as follows. The state of the system is composed of the local state of each automaton and a global date. We denote such a state by $\left((q_i, dor_i)_{i \in [1,n]}, \theta\right)$. The local state (l_i, dor_i) of an automaton is given by its location l_i and the dates of last resets of clocks of the automaton given by the function $dor_i : X_i \to \mathbb{R}$. To fire a global transition $(t_i)_{i \in [1,n]}$ at time $\theta' \geq \theta$, where θ is the current date, we require that the invariants of all locations of all the automata are still valid at the time θ' (they are true for all the values of the

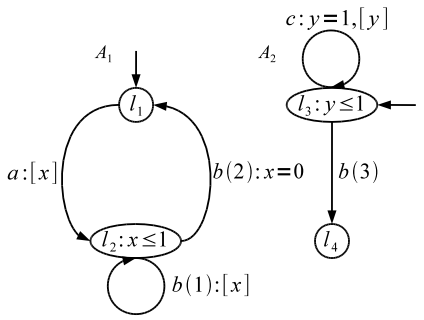

Figure 7: Timed automata with two clocks x and y. Each of the locations is marked with a label and invariants. The transitions are labeled with their action (a, b or c), the guard on clocks and the set of resets (in square brackets).

clock $\theta' - dor$) and all the automata that need to fire a local transition t can do this. In other words this means that the initial location of the transition t has to be the current location of the automaton and the guard has to be satisfied at the moment θ. The resulting overall state is obtained by changing the locations of the automata in accordance with the considered local transitions and by reseting clocks mentioned in the transitions. We must also ensure that the invariants of the new locations are true at the time θ'.

Definition 8. (Operational semantics) *The operational semantics of NTA is defined by the following transition system:*

- *The initial global state is* $\left((q_{0i}, X_i \times \{0\})_{i\in[1,n]}, 0\right)$;
- *The transitions are defined by the following SOS rule:*
  - *for each* $i \in [1,n]$

$$\frac{\left\{\begin{array}{l} I_i(q_i)(\theta' - dor_i) \wedge \\ (t_i \neq \epsilon) \implies \left\{\begin{array}{l} \alpha(t_i) = q_i \wedge \gamma(t_i)(\theta' - dor_i) \\ q_i' = \beta(t_i) \wedge I_i(q_i')(\theta' - dor_i') \\ dor_i'(x) = \begin{cases} \theta' & \text{if } x \in \rho(t_i) \\ dor_i(x) & \text{otherwise} \end{cases} \end{array}\right. \end{array}\right.}{\left((q_i, dor_i)_{i\in[1,n]}, \theta\right) \xrightarrow{(t_i)_{i\in[1,n]}\in \text{Sync}} \left((q_i', dor_i')_{i\in[1,n]}, \theta'\right)}$$

In the example presented in Figure 7 we have a network of automata which produces four actions a, b and two actions c. The possible explanations of these actions are shown in Figure 8. What we want to know are the possible dates of these actions (denoted $\delta(a), \delta(b)$ and $\delta(c)$). As we can see the action c can be only produced at time 1. After this action we can reset the clock and repeat it at time 2. After these two actions automaton A_1 fires action b but to do this it has to be synchronized with the second automaton A_2. Thus automaton A_2 hast to go to the location l_2 before the action c and then it can choose between two transitions labeled by b. If the automaton executes the action b which produces the state in which automaton A_1 is in location 1_1 we know that $\delta(a) = \delta(b)$ because of the guard $x = 0$. $\delta(b) \leq 3$ because otherwise we would have another action c. Moreover, since we know that the global time progresses we can infer that

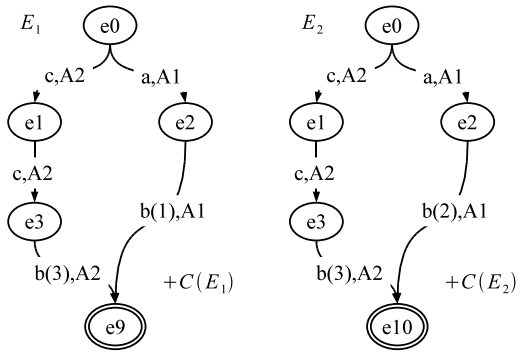

Figure 8: Two possible explanations E_1 and E_2 with time constraints $C(E_1), C(E_2)$ on dates of occurrences of events.

$\delta(b) \in [2,3]$. In the other case where the transition b was the effect of the synchronization of $b(1)$ with $b(3)$, we have $\delta(b) \leq 3$ like previously but also $\delta(b) \leq \delta(a) + 1$ because otherwise another action c would be produced. We can deduce in fact that $\max(\delta(a), 2) \leq \delta(b) \leq \min(\delta(a) + 1, 3)$. This is the type of information we will try to infer automatically during the supervision.

6. SUPERVISION OF TIMED SYSTEMS

The question of supervision of timed systems can be placed in the same context as in the case of systems without time constraints. Let us imagine a sequence of observations made only by a series of symbols of the alphabet Σ. The problem is to find executions that can explain this sequence. What is especially interesting is to infer the causal relationships induced by the model.

The first idea is to proceed as in the untimed case and to define the notion of unfolding of a NTA. The sequence of observations can be also naturally seen as a TA without time constraints. The notion of timed unfoldings has recently been introduced in [**?**]. Its calculation is complex and produces a structure of events in which each of the events has assigned to it a symbolic expression which gives the possible dates of firing transitions. The concept of conflict also has to be weakened in the notion of asymmetrical conflict in order to keep concurrency.

We propose here a simpler approach to answer the question of supervision. By definition, the introduction of time constraints limits the possible executions of the model. Thus we can consider the explanations produced for the underlying untimed model (i.e. the unfolding guided by the observations) and then take into account the time constraints. We will see furthermore that this phase of post-selection will be able to infer the possible dates of observation. This is a potentially rich information for supervision activities.

Let us thus consider a NTA and its untimed underlying network N and a sequence of observations σ and let us take the set of possible untimed explanations, i.e. all the sets of events $E \subseteq \mathcal{E}(N, \sigma)$ such that $\downarrow E = E$ is without conflict and $\varsigma(e) = \varpi(\sigma)$.

We consider an explanation E. We denote by E_i the set of events concerned by the automaton A_i (i.e. the events e such that $\tau_i(e) \neq \epsilon$). We know that the closure $\rightarrow_i^*$ is a total order on E_i since the processes are sequential. We will denote by $\uparrow_i E$ the maximal event for this relation. For

each event, we will denote by $\delta(e)$ a date of the event and by $dor_i(e)$ a date of reseting clocks (X_i) of the automaton A_i after the event occurred. $dor_i(e)$ is defined as follows:

$$\forall x \in X_i, \forall e \in E, dor_i(e)(x) \stackrel{def}{=} \begin{cases} \delta(e) & \text{if } x \in \rho(\tau_i(e)) \\ dor_i(\pi_i(e))(x) & \text{otherwise} \end{cases}$$

Definition 9. (Time validity) *An explanation E which does not take time into consideration is valid according to the time constraints of the network iff:*

$$\forall i \in [1,n] \begin{cases} \forall e \neq \perp \in E_i \begin{cases} \delta(\pi_i(e)) \leq \delta(e) \\ I_i(\alpha(\tau_i(e)))(\delta(e) - dor_i(\pi_i(e))) \\ \gamma(\tau_i(e))(\delta(e) - dor_i(\pi_i(e))) \end{cases} \\ I_i(\beta(\tau_i(\uparrow_i E)))(max_{f \in E}\delta(f) - dor_i(\uparrow_i E)) \end{cases}$$

The definition of time validity incorporates the definition of sequential semantics. We explain the formula line by line:

- the time cannot go back between the causal predecessor of an event and itself;
- the invariants of the initial locations of the transition are satisfied at the moment of firing the transition;
- the guards of the local transitions which form the transition of the considered event are also satisfied at the time of firing;
- the invariants of the final locations of the transition are satisfied at the end of the explanation.

These conditions can therefore reject the explanations which are invalid if time constraints are taken into consideration. Beyond that, we can pass to the symbolic representation constraints having in mind that for each event e, $\delta(e)$ is a real variable assigning the possible dates of firing the corresponding transition of the event e.

For our example in Figure 8, the two possible explanations are described with the following constraints:

- $$C(E_1) \equiv \begin{cases} \delta(\perp) \leq \delta(e_1) \\ \delta(e_1) \leq \delta(e_3) \wedge \delta(e_3) \leq \delta(e_9) \\ \delta(\perp) \leq \delta(e_2) \wedge \delta(e_2) \leq \delta(e_9) \\ \delta(e_9) - \delta(e_3) \leq 1 \wedge \delta(e_9) - \delta(e_2) \leq 1 \\ \delta(e_3) - \delta(e_1) \leq 1 \wedge \delta(e_3) - \delta(e_1) = 1 \\ \delta(e_1) - \delta(\perp) \leq 1 \wedge \delta(e_1) - \delta(\perp) = 1 \\ [\max(\delta(\perp), \delta(e_1), \delta(e_2), \delta(e_3), \delta(e_9)) + \\ -\delta(e_2) \leq 1] \end{cases}$$
- $$C(E_2) \equiv \begin{cases} \delta(\perp) \leq \delta(e_1) \\ \delta(e_1) \leq \delta(e_3) \wedge \delta(e_3) \leq \delta(e_{10}) \\ \delta(\perp) \leq \delta(e_2) \wedge \delta(e_2) \leq \delta(e_{10}) \\ \delta(e_{10}) - \delta(e_2) \leq 1 \wedge \delta(e_{10}) - \delta(e_2) = 0 \\ \delta(e_{10}) - \delta(e_3) \leq 1 \wedge \delta(e_3) - \delta(e_1) \leq 1 \\ \delta(e_3) - \delta(e_1) = 1 \wedge \delta(e_1) - \delta(\perp) \leq 1 \\ \delta(e_1) - \delta(\perp) = 1 \end{cases}$$

After reduction and with assumption that $\delta(\perp) = 0$, we obtain

$$C(E_1) \equiv (e_1 = 1) \wedge (e_3 = 2) \wedge (2 \leq e_9 \leq 3) \wedge (0 \leq e_9 - e_2 \leq 1)$$

and

$$C(E_2) \equiv (e_1 = 1) \wedge (e_3 = 2) \wedge (2 \leq e_{10} \leq 3) \wedge (e_{10} - e_2 = 0)$$

which confirms very well the previous informal analysis of the executions of the timed model.

7. CONCLUSION

In this article we presented an original method using the model of networks of timed automata to produce timed explanations of a sequence of actions produced by a distributed system under surveillance. Many possible applications can be considered such as the correlation of alarms and the detection of errors, monitoring behavior patterns to detect for example intrusions, surveillance of non-functional time properties, etc. A detailed algorithimc analysis is still necessary. Several extensions can be also naturally explored such as the real distribution of supervision and the use of different and more realistic models of time.

During our research we have developed a prototype tool which implements the solutions presented in this paper. We used this tool to prepare a more realistic example which is described in the appendix.

8. ACKNOWLEDGEMENTS

This works has been done in the Distribcom research group of IRISA-INRIA. It has been partly founded by the the national french project ANR DOTS under the reference ANR-06-SETI-003.

APPENDIX

A. EXAMPLE 1

In the following example we present a network of timed automata which models the alternating bit protocol. The whole model consists of four automata presented in Figure 9. We assume that a user of the protocol can observe only four transitions of the model i.e. $!m_0(0)$ which is the first emission of a message with bit 0, $!m_1(2)$ which is the first emission of a message with bit 1, $?m_0(0)$ and $?m_1(1)$ which are the first reception of bit 0 and 1 respectively. It is worth noticing that the user cannot observe the retransmissions ($!m_0(2)$ and $!m_1(0)$). This is a task of the protocol to retransmit and manage all of the unobservable transitions. Let us also mention that the user do not observe any transitions of the automata which model medium.

In our example we want to answer the folowing questions.

- Given the sequence of observations $\sigma = !m_0(0), ?m_0(0), ?m_1(1), !m_1(2), ?m_0(0)$ is there an execution of the given model which satisfies σ?

  In other words we check if there is a possibility of sending a message once (in our example $!m_0(0)$) and receiving it twice by the receiver ($?m_0(2)$).
- How do such possible executions look like?
- How can we avoid such executions when we take into account time constraints of the model?

To address the first and the second question we can use the method proposed in the article and we can construct the prefix of the unfolding of the network. Because there

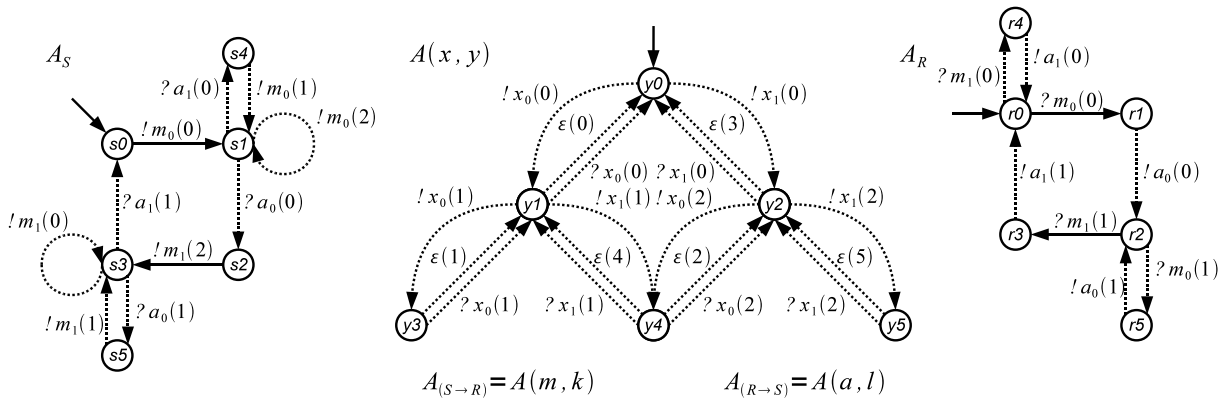

Figure 9: The alternating bit protocol. In the picture there are three different automata each of which has its role in the protocol. Thus there is the sender A_S, the receiver A_R, and an automaton $A(x,y)$ which is used to simulate a medium for the communication in one direction. $A_{(S\to R)}$ denotes a one-way communication channel from the sender to the receiver, whereas $A_{R\to S}$ is used for the opposite direction. Dotted lines denote unobservable transitions. We can observe only transitions which are drawn with a solid line.

are some unobservable events and we want to terminate the construction of the prefix at some point we give a limit of maximum five unobservable events in all executions. All the possible explanations with respect to the given sequence of observations and the termination condition are presented in Figure 10. There are eight successful executions each with a different maximal event (drawn using double ellipses). In fact during construction of the prefix the actual number of events was much higher (307 events were generated for this example).

Before we answer the last question below we give the constraints associated to the automata in Figure 9:

- for the sender A_S there we have the following constraints:

$$\gamma(!m_0(2)) \stackrel{def}{=} c_S \geq \beta$$
$$\gamma(!m_1(0)) \stackrel{def}{=} c_S \geq \beta$$
$$\rho(!m_0(0)) \stackrel{def}{=} \{c_S\}$$
$$\rho(!m_1(2)) \stackrel{def}{=} \{c_S\}$$

- for the two automata $A_{S\to R}$ and $A_{R\to S}$:

$$\rho(!x_0(0)) \stackrel{def}{=} \rho(?x_0(1)) \stackrel{def}{=} \rho(?x_0(2)) \stackrel{def}{=} \{c_A\}$$
$$\rho(!x_1(0)) \stackrel{def}{=} \rho(?x_1(1)) \stackrel{def}{=} \rho(?x_1(2)) \stackrel{def}{=} \{c_A\}$$
$$\rho(\varepsilon(1)) \stackrel{def}{=} \rho(\varepsilon(4)) \stackrel{def}{=} \rho(\varepsilon(2)) \stackrel{def}{=} \rho(\varepsilon(5)) \stackrel{def}{=} \{c_A\}$$
$$\forall_{i\in[1,5]} I(y_i) \stackrel{def}{=} c_A \leq \alpha$$

In the example we assume that each of the automata has its own non-shared clock. In the time constraints above we use two parameters α and β. With the parameter α we can control the time which is spent to transmit a message through the medium (the two automata $A_{S\to R}$ and $A_{R\to S}$). Whereas β is a threshold value of the minimal time amount after which a message can be retransmitted by the sender.

To present our method we show on one of the successful explanations how we can avoid the corresponding execution just by adjusting the two variables α and β. For the following example we choose the execution E shown in Figure 10 (on the right) ending with the event e_{301}. Using Definition 9 we get the following system of constraints for this execution:

$$\begin{cases}
\delta(\bot) \leq \delta(e_1) \wedge \delta(e_1) \leq \delta(e_4) & \\
\delta(\bot) \leq \delta(e_{11}) \wedge \delta(e_4) \leq \delta(e_{11}) & \\
\delta(\bot) \leq \delta(e_{27}) \wedge \delta(e_{11}) \leq \delta(e_{27}) & \\
\delta(e_{27}) \leq \delta(e_{71}) \wedge \delta(e_4) \leq \delta(e_{71}) & \\
\delta(e_{11}) \leq \delta(e_{152}) \wedge \delta(e_{71}) \leq \delta(e_{152}) & \\
\delta(e_{27}) \leq \delta(e_{208}) \wedge \delta(e_{152}) \leq \delta(e_{208}) & \\
\delta(e_{208}) \leq \delta(e_{263}) \wedge \delta(e_{71}) \leq \delta(e_{263}) & \\
\delta(e_{208}) \leq \delta(e_{301}) \wedge \delta(e_{263}) \leq \delta(e_{301}) & \\
\delta(e_{301}) - \delta(e_{208}) \leq \alpha \wedge \delta(e_{208}) - \delta(e_{11}) \leq \alpha & \\
\delta(e_{71}) - \delta(e_{27}) \leq \alpha \wedge \delta(e_{152}) - \delta(e_{11}) \leq \alpha & \\
\delta(e_4) - \delta(e_1) \geq \beta \wedge \delta(e_4) - \delta(e_1) \leq \alpha & (1)\wedge(2) \\
\delta(e_{11}) - \delta(e_1) \leq \alpha & (3) \\
\max_{e\in E} \delta(e) - \delta(e_{263}) \leq \alpha &
\end{cases}$$

When we take and reduce the three formulas (1), (2) and (3) we can simply deduce that $\beta \leq 2\alpha$. This means that it is impossible to have the execution if we set the retransmission time which is more then the double time of transmission by the medium.

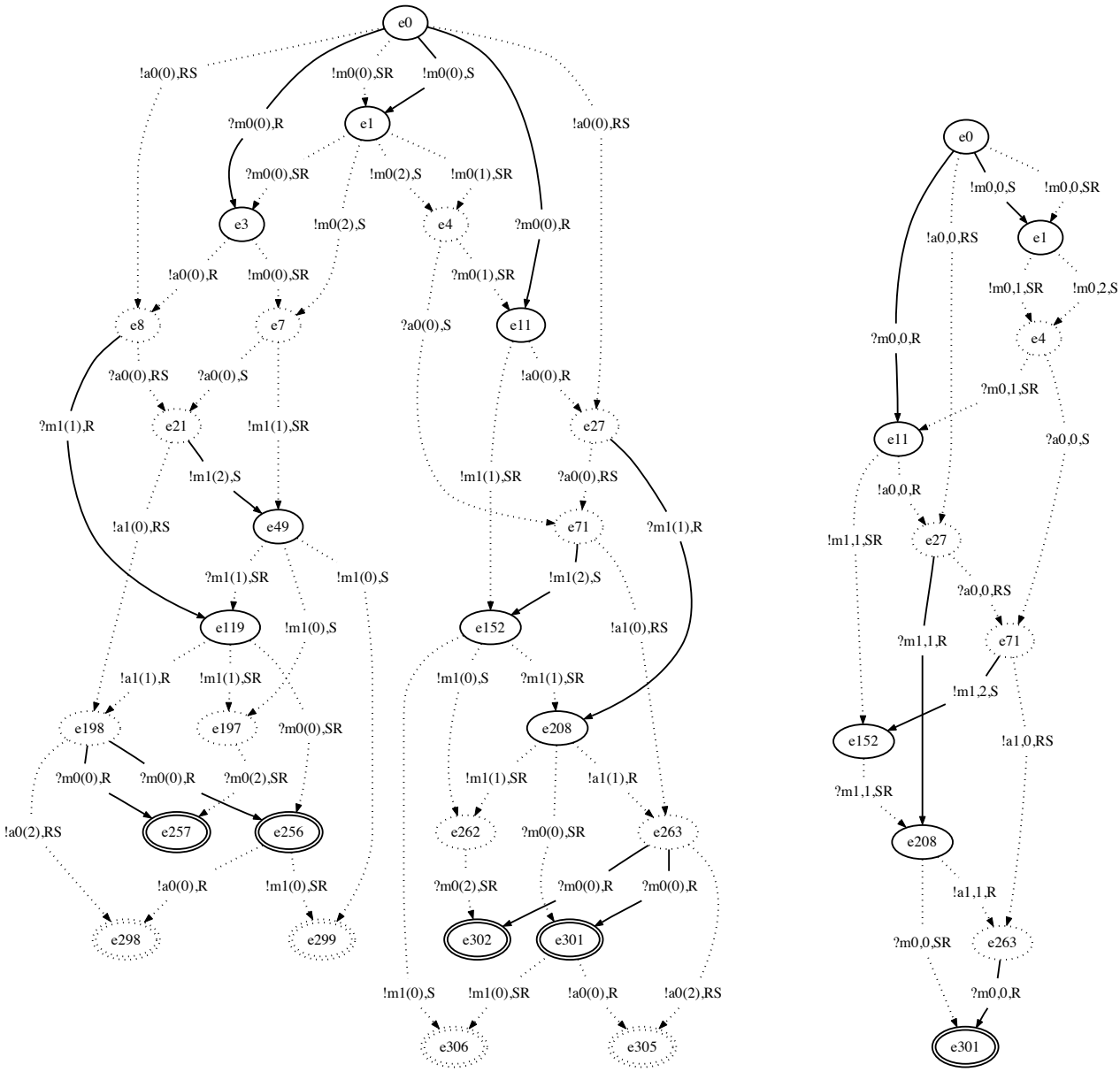

Figure 10: The preifx of the unfolding of the model in Figure 9 (on the left) constructed in accordance with the described search criteria. In the picture there are only events which were successful and their causal predecessors. The rest of the events produced during the search procedure are removed. Dotted arrows are unobservable transitions, dotted ellipses unobservable events. The events drawn using double ellipses are the ones which satisfiy the sequence of observations. It is worth noticing that in some explanations the final events are unobservable e.g. the event e_{305}. On the right a subset of the prefix used as an example in the text.