Symbolic Unfoldings For Networks of Timed Automata

Franck Cassez[1]*, Thomas Chatain[2] and Claude Jard[3]

[1] CNRS/IRCCyN, Nantes, France – franck.cassez@cnrs.irccyn.fr

[2] IRISA/INRIA, Campus de Beaulieu, Rennes, France – Thomas.Chatain@irisa.fr

[3] IRISA/ENS Cachan, Campus de Kerlann, Bruz, France – Claude.Jard@irisa.fr

Abstract In this paper we give a symbolic concurrent semantics for network of timed automata (NTA) in terms of *extended symbolic nets*. Extended symbolic nets are standard occurrence nets extended with *read arcs* and *symbolic constraints* on places and transitions. We prove that there is a *complete finite prefix* for any NTA that contains at least the information of the simulation graph of the NTA but keep explicit the notions of concurrency and causality of the network.

1 Introduction

Concurrent Semantics for Finite State Systems. The analysis of *distributed* or *concurrent* finite state systems has been dramatically improved thanks to *partial-order* methods (see e.g. [21]) that take advantage of the *independence* between actions, and to the *unfolding* based methods [11,16] that improve the partial order methods by taking advantage of the *locality* of actions.

Timed Systems. The main models that include timing information and are used to specify distributed timed systems are networks of timed automata (NTA) [1], and time Petri nets (TPN) [17]. There are a number of theoretical results about NTA and TPN and efficient tools to analyze them have been developed. Nevertheless the analysis of these models is always based on the exploration of a graph which is a single large automaton that produces the same behaviours as the NTA or the TPN; this induces an exponential blow up in the size of the system to be analysed.

Related Work. In [13,18], the authors define an alternative semantics for NTA based on local time elapsing. The efficiency of this method depends on two opposite factors: local time semantics generate more states but the independence relation restricts the exploration. In [15] (a generalization of [22]), the independence between transitions in a TA is exploited in a different way: the key observation is that the occurrences of two independent transitions do no need to be ordered and consequently nor do the occurrences of the clock resets. The relative drawback of the method is that, before their exploration, the symbolic states include more variables than the clock variables. Partial order methods for TPNs are studied in [20], where the authors generalize the concept

* Work supported by the project CORTOS, a program of the French government.

of *stubborn set* to time Petri nets, calling it a *ready set*. They apply it to the *state class graph* construction of [5]. The efficiency of the method depends on whether the (dynamical) timing coupling between transitions is weak or not. Unfortunately the urgent semantics of this model entails a strong timing coupling. The previous *partial order* methods only take advantage of the independence of actions and not of any locality property. We are interested in a true concurrent semantics for NTA and this has not been developed in the aforementioned work.
Process semantics for time Petri nets which is a generalization of the unfolding semantics for time Petri nets has been developed by different researchers. From a semantical point of view, Aura and Lilius have studied in [19] the *realizability problem* of a non branching process in a TPN. They build an unfolding of the untimed Petri net underlying a safe TPN, and add constraints on the dates of occurrence of the events. It is then possible to check that a timed configuration is valid or not. In [12] the authors consider bounded TPN and a discrete time domain: the elapsing of one time unit is a special transition of the net. Thus the global synchronization related to this transition heavily decreases the locality property of the unfolding. Furthermore, when the intervals associated with the transitions involve large integers, this method suffers the usual combinatorial explosion related to the discrete time approach.

Section 3 of this paper can be viewed as the counterpart of the work of Aura and Lilius [19] in the framework of NTA: we define similar notions for NTA and build a *symbolic unfolding* which is a *symbolic net*. We have to extend the results of Aura and Lilius because there is no urgency for firing a transition[1] in a NTA. As stated in [19] those unfoldings are satisfactory for *free choice nets* which are a strict subclass of TPN. Our NTA are not free choice nets and in section 4 we refine our symbolic unfolding to obtain an *extended symbolic unfolding* which is a symbolic net with *read arcs*.

Following our recent approach [9] using the notion of symbolic unfolding to capture the partial order behaviors of TPN, we propose in this paper a similar notion for NTA, but we cannot directly apply the framework of [9]. Indeed TA and TPN have different expressive powers [4,8] and as stated earlier NTA do not have the nice *urgency* features that TPN have.

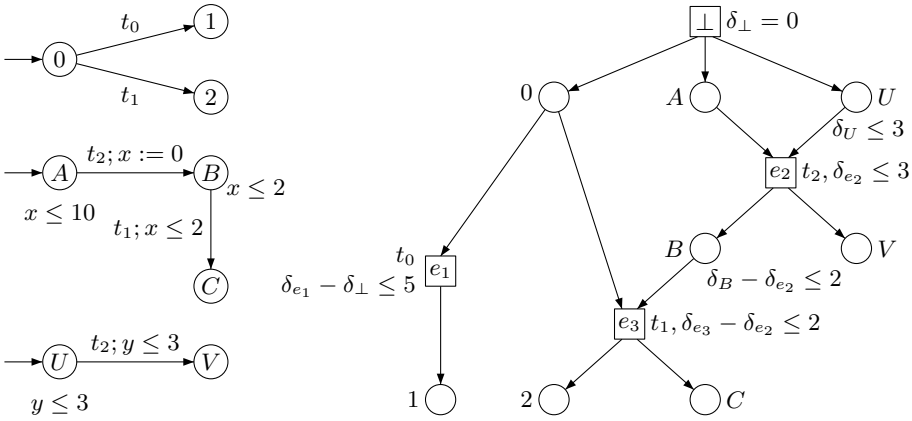
Up to our knowledge, this is the first attempt to equip NTA with a concurrent semantics, which can be finitely represented by a prefix of an unfolding. In this paper we answer the following questions:

1. What can be a good model for a *concurrent semantics* of NTA? The result is an extension of the model of symbolic nets we have proposed in [9];
2. How to define a *concurrent semantics* for NTA, *i.e.* how to define a *symbolic unfolding* that captures the essential properties of a NTA while preserving concurrency information? This is achieved in two steps: first build a *symbolic unfolding* and use this object to build a proper *extended symbolic unfolding* of the NTA. By *proper* unfolding, we mean a symbolic Petri net on which we can check that a *local configuration* is valid using only the *extended causal* past of an event.
3. Is there a *complete finite prefix* for NTA? This result is rather easy to obtain on the symbolic unfolding object and carries over to the extended symbolic unfolding.

[1] *invariants* and *guards* can be independent and a transition is not bound to fire before its deadline given by the *guard*.

About point 3 above, we are not addressing the problem of building such a prefix efficiently but our work is concerned with identifying the key issues in the construction of a prefix for NTA. The solution proposed in [9] builds a complete finite prefix for safe TPNs, but with no guarantee that this prefix is one of the smallest, which is a very difficult problem to solve. Based on this work, we address more basic questions about NTA, which are in a sense easier to study than safe TPNs because the concurrent structure is explicit.

Key Issues. In this section we present informally the problem and the key issues raised by the three previous questions. In the case of networks of finite automata, *finite complete prefixes* exist. For example, for the network[2] of Fig. 1(a), a finite complete prefix is given on Fig. 1(b). Finite complete prefixes contain full information about the reachable states of the network and about the set of events that are *feasible* in the network. A set of events (labels) is feasible iff it is a word that can be generated by the network. For example, $\{t_1\}$ is not a feasible set of events in the network $\mathcal{N}_1$, because t_1 must be



(a) The NTA $\mathcal{N}_1$ (b) Symbolic unfolding for the network $\mathcal{N}_1$

Figure 1. A NTA and its Symbolic Unfolding

preceded by t_2. And this appears in the unfolding as event e_3 (labelled by t_1) must be preceded by e_2 (labelled by t_2). In an unfolding, a set of events K is a *configuration* if there is a reachable marking obtained by firing each event in K. For example $\{\perp, e_1\}$ is a configuration, $\{\perp, e_1, e_2\}$ as well, but $\{\perp, e_3\}$ is not as e_3 must be preceded by e_2 before it occurs. The minimal set of events necessary for an event e to occur is called the *causal past* (or *local configuration*) of e. Note that by definition a configuration contains the causal past of each of its event. A *complete* prefix is an unfolding that satisfies property (P): a set of events is feasible in the NTA iff it is a configuration of the

[2] The automata synchronize on common labels. Labels of the events and places represent the corresponding location and transition in the network of automata. The constraints appearing near each node are explained later and can be ignored at this stage.

unfolding[3]. This property of unfoldings is the key point in the untimed case and allows one to do model-checking on the complete finite prefix. This unfolding can also be used for *fault diagnosis* purposes which is a very important application area.

In the case of networks of timed automata, we deal with *timed events* which are pairs (e, δ) where $\delta \in \mathbb{R}_{\geq 0}$. A set of timed events E is feasible iff there is a run in the NTA that generates a timed word that contains all the timed events in E. To decide whether a set of timed events is feasible in a network of timed automata, we can build a *symbolic unfolding*. For this, we add a symbolic timing constraint $g(e)$ to each event of the previous unfolding. For example, with e_1 we can associate the constraint $g(e_1) \stackrel{def}{=} \delta_{e_1} - \delta_\perp \leq 5$, where δ_e is the variable that represents the date of occurrence of e. A set of timed events $\{(e_1, d_1), \cdots, (e_k, d_k)\}$ is a *timed configuration* if $\{e_1, e_2, \cdots, e_k\}$ is a configuration and the constraint $g(e_1) \wedge \cdots \wedge g(e_k)$ is satisfied when replacing each δ_{e_i} by d_i. For example $\{(\perp, 0), (e_1, 4)\}$ is a timed configuration with $g(\perp) \stackrel{def}{=} \delta_\perp = 0$. Thus the property we would like to have for symbolic unfoldings is (P'): $\{(e_1, d_1), \cdots, (e_k, d_k)\}$ is a timed configuration iff there is a run $(e_{f(1)}, d_{f(1)}), \cdots, (e_{f(k)}, d_{f(k)})$ in the NTA with f a one-to-one mapping from $1..k$ to $1..k$. In the untimed case, one can check that an event is fireable in the unfolding using only the causal past of the event. We want this property to hold for the timed unfoldings as well and then a formula associated with an event e should only involve variables that are associated with events in the causal past of e (the local configuration of e). Now assume we want to decide whether $\{(\perp, 0), (e_1, d_1), (e_2, d_2)\}$ is a timed configuration. It is actually if $d_1 - d_2 \leq 2$. But this cannot be captured by any conjunction $g(\perp) \wedge g(e_1) \wedge g(e_2)$ because e_1 is not in the causal past of e_2 and e_2 not in the causal past of e_1. A symbolic unfolding built by associating constraints with each event e, with the property that each constraint $g(e)$ uses only variables in the causal past of e, does not always contain enough information for property (P') to hold. In this paper we show 1) how to build an unfolding that contains enough information so that (P') holds; 2) how to build a finite and complete prefix of the unfolding satisfying (P').

Organization of the Paper. The paper is organized as follows. Section 2 presents the model of NTA and its usual sequential semantics. Section 3 gives a concurrent semantics for NTA in terms of *symbolic branching processes* (SBP) and proves the existence of complete finite prefixes. The SBP is a first step towards a complete finite prefix having property (P'). In section 4, we show how to build an *extended* SBP, using *read-arcs*, which is a complete finite prefix satisfying property (P'). Section 5 gives a summary of the paper and directions for future work. The proofs of the theorems are omitted and can be found in the extended version of the paper [7].

2 Networks of timed automata

Notations . Given a set B we use B^ε for the set $B \cup \{\varepsilon\}$ (assuming $\varepsilon \notin B$). Let $X = \{x_1, \cdots, x_n\}$ be a finite set of *clock* variables. A *valuation* ν is a mapping from X to $\mathbb{R}_{\geq 0}$. Let $X' \subseteq X$. The valuation $\nu[X']$ is defined by: $\nu[X'](x) = 0$ if $x \in X'$

[3] Actually we should write "it is a labeling" of a configuration of the unfolding.

and $\nu[X'](x) = \nu(x)$ otherwise. $\nu_{|X'}$ is the restriction (projection) of ν to X' and is defined by $\nu_{|X'}(x) = \nu(x)$ for $x \in X'$. We denote $\mathbf{0}$ the valuation defined by $\mathbf{0}(x) = 0$ for each $x \in X$. For $\delta \in \mathbb{R}$, $\nu + \delta$ is the valuation defined by $(\nu + \delta)(x) = \nu(x) + \delta$. $\mathcal{C}(X)$ is defined to be the set of conjunctions of terms of the form $x - x' \bowtie c$ or $x \bowtie c$ for $x, x' \in X$ and $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. $\mathcal{C}(X)$ is called the set of *diagonal constraints* over X. The set of *rectangular* constraints, $\mathcal{C}_r(X)$ is the subset of $\mathcal{C}(X)$ where only constraints of the form $x \bowtie c$ appear. Given a formula $\varphi \in \mathcal{C}(X)$ and a valuation $\nu \in \mathbb{R}_{\geq 0}^X$, we use $\varphi[x/\nu(x)]$ for φ where x is replaced by $\nu(x)$. we denote $\varphi(\nu) \in \{\mathsf{tt}, \mathsf{ff}\}$ the truth value of $\varphi[x/\nu(x)]$. We let $[\![\varphi]\!] = \{\nu \in \mathbb{R}_{\geq 0} \mid \varphi(\nu) = \mathsf{tt}\}$. A subset Z of $\mathbb{R}_{\geq 0}^X$ is a zone if $Z = [\![\varphi_Z]\!]$ for some $\varphi_Z \in \mathcal{C}(X)$. Note that the intersection of two zones is a zone. Two operators are defined on zones: the *time successor* operator, $Z^{\nearrow} = \{v + \delta \mid v \in Z, \delta \in \mathbb{R}_{\geq 0}\}$ and the *R-reset* operator, $Z[R] = \{v \mid \exists v' \in Z \text{ s.t. } v = v'[R]\}$. Both $Z^{\nearrow}$ and $Z[R]$ are zones if Z is a zone.

Timed Automata. *Timed automata* were introduced in [1] to model systems which combine *discrete* and *continuous* evolutions.

Definition 1. *A* timed automaton $\mathcal{A}$ *is a tuple* $(L, \ell_0, \Sigma, X, T, \mathrm{Inv})$ *where:* L *is a finite set of* locations*;* ℓ_0 *is the* initial *location;* Σ *is a finite set of* discrete *actions;* $X = \{x_1, \cdots, x_n\}$ *is a finite set of (positive real-valued)* clocks*;* $T \subseteq L \times \mathcal{C}_r(X) \times \Sigma \times 2^X \times L$ *is a finite set of* transitions*:* $(\ell, g, a, R, \ell') \in T$ *represents a transition from the location* ℓ *to* ℓ'*, labeled by* a*, with the guard* g *and the reset set* $R \subseteq X$*; we write* $\mathrm{SRC}(t) = \ell$*,* $\mathrm{TGT}(t) = \ell'$*,* $\mathrm{G}(t) = g$*,* $\lambda(t) = a$ *and* $\mathrm{R}(t) = R$*.* $\mathrm{Inv} \in \mathcal{C}_r(X)^L$ *assigns an* invariant *to any location. We require that* Inv *be a conjunction of terms of the form* $x \bowtie c$ *with* $\bowtie \in \{<, \leq\}$ *and* $c \in \mathbb{N}$*.*

A *state* of a timed automaton is a pair $(\ell, v) \in L \times \mathbb{R}_{\geq 0}^X$. A timed automaton is *bounded* if there exists a constant $k \in \mathbb{N}$ s.t. for each $\ell \in L$, $\mathrm{Inv}(\ell) \subseteq [\![0 \leq x_1 \leq k \wedge \cdots \wedge 0 \leq x_n \leq k]\!]$. Examples of timed automata are given in Fig. 1(a). In the sequel we require that for any valuation v and any transition $t = (\ell, g, a, R, \ell')$, $g(v) \implies \mathrm{Inv}(\ell')(v[R])$.

Definition 2. *The semantics of a timed automaton* $\mathcal{A} = (L, \ell_0, \Sigma, X, T, Inv)$ *is a labeled timed transition system (TTS)* $S_{\mathcal{A}} = (Q, q_0, T \cup \mathbb{R}_{\geq 0}, \rightarrow)$ *with* $Q = L \times (\mathbb{R}_{\leq 0})^X$*,* $q_0 = (\ell_0, \mathbf{0})$ *is the initial state and* $\rightarrow$ *consists of the discrete and continuous transition relations: i) the discrete transition relation is defined for all* $t \in T$ *by:* $(\ell, v) \xrightarrow{t} (\ell', v') \iff \exists t = (\ell, g, a, R, \ell') \in T$ *s.t.* $g(v) = \mathsf{tt}$*,* $v' = v[R \mapsto 0]$*; ii) the continuous transition relation is defined for all* $\delta \in \mathbb{R}_{\geq 0}$ *by:* $(\ell, v) \xrightarrow{\delta} (\ell', v')$ *iff* $\ell = \ell'$*,* $v' = v + \delta$ *and* $\forall 0 \leq \delta' \leq \delta,\ Inv(\ell)(v + \delta') = \mathsf{tt}$*. A* run *of a timed automaton* $\mathcal{A}$ *is a path in* $S_{\mathcal{A}}$ *starting in* q_0 *where continuous and discrete transitions alternate*[4]*. The set of runs of* $\mathcal{A}$ *is denoted by* $[\![\mathcal{A}]\!]$*. A state* q *is* reachable *in* $\mathcal{A}$ *if there is a run from* q_0 *to* q*.* $\mathrm{REACH}(\mathcal{A})$ *is the set of reachable states of* $\mathcal{A}$*. A timed word* $w \in (T \times \mathbb{R}_{\geq 0})^*$ *is* accepted *by* $\mathcal{A}$ *if there is a run* $\rho \in [\![\mathcal{A}]\!]$ *s.t. the trace of* ρ *is* w*.*

The analysis of timed automata is based on the exploration of a (finite) graph, the *simulation graph*, where the nodes are *symbolic states*. A symbolic state is a pair (ℓ, Z) where ℓ is a location and Z a *zone* over the set $\mathbb{R}_{\geq 0}^X$.

[4] In our definition runs are labeled by transitions.

Definition 3. *The* simulation graph $SG(\mathcal{A})$ *of a timed automaton* $\mathcal{A}$ *is given by: i) the set of states is the set of symbolic states of the form* (ℓ, Z) *where* Z *is a zone; ii) the initial state is* (ℓ_0, Z_0) *with* $Z_0 = \mathbf{0}^{\nearrow} \cap [\![\text{Inv}(\ell_0)]\!]$*; iii)* $(\ell, Z) \xrightarrow{a} (\ell', Z')$ *if there is a transition* (ℓ, g, a, R, ℓ') *in* $\mathcal{A}$ *s.t.* $Z \cap [\![g]\!] \neq \emptyset$ *(this ensures* Z' *is not empty) and* $Z' = \big((Z \cap [\![g]\!])[R]\big)^{\nearrow} \cap [\![\text{Inv}(\ell')]\!]$.

We assume that the timed automata are bounded *i.e.* in each location ℓ, $\text{Inv}(\ell)$ is bounded[5]. In this case the number of zones of the simulation graph is finite [14,6].

Network of Timed Automata. We use the classical composition notion based on a *synchronization function*. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n timed automata with $\mathcal{A}_i = (L_i, l_{i,0}, \Sigma_i, X_i, T_i, Inv_i)$. We assume that for each $i \neq j$, $L_i \cap L_j = \emptyset$ and $X_i \cap X_j = \emptyset$ (clocks are not shared). A *synchronization constraint* I is a subset of $\Sigma_1^\varepsilon \times \Sigma_2^\varepsilon \cdots \times \Sigma_n^\varepsilon \backslash (\varepsilon, \cdots, \varepsilon)$. The (synchronization) vectors of a synchronization constraint I indicate which actions synchronize. For $(t_1, \cdots, t_n) \in T_1^\varepsilon \times \cdots T_n^\varepsilon$ we write $\lambda(t_1, \cdots, t_n) = (\lambda_1(t_1), \cdots, \lambda_n(t_n))$ with $\lambda_i(\varepsilon) = \varepsilon$. $\lambda^{-1}(I) \subseteq T_1^\varepsilon \times \cdots T_n^\varepsilon$ indicates how the transitions synchronize. For $t \in \lambda^{-1}(I)$, we let: $\textsc{src}^*(t) = \{l \in \textsc{src}(t[i]) \,|\, t[i] \neq \varepsilon\}$, $\textsc{tgt}^*(t) = \{l \in \textsc{tgt}(t[i]) \,|\, t[i] \neq \varepsilon\}$, $\text{R}(t) = \{x \,|\, x \in \text{R}(t[i])$ *and* $t[i] \neq \varepsilon\}$, $\text{G}(t) = \wedge_{t[i] \neq \varepsilon} \text{G}(t[i])$.

Definition 4. *The network of timed automata (NTA)* $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$ *is the timed automaton* $\mathcal{B} = (L, \mathbf{l}_0, \Sigma, X, T, \text{Inv})$ *defined by:* $L = L_1 \times \cdots \times L_n$, $\mathbf{l}_0 = (\ell_{1,0}, \cdots, \ell_{n,0})$, $\Sigma = \Sigma_1 \times \cdots \times \Sigma_n$, $X = \cup_{i=1}^n X_i$*;* $(\mathbf{l}, g, a, R, \mathbf{l}') \in T$ *iff* $\exists t \in \lambda^{-1}(I)$ *s.t.: (1) if* $t[i] \neq \varepsilon$ *then* $\mathbf{l}_i = \textsc{src}(t[i])$ *and otherwise* $\mathbf{l}'_i = \textsc{tgt}(t[i])$*, (2)* $a = \lambda(t)$, $g = \text{G}(t)$ *and* $R = \text{R}(t)$ *and* $\text{Inv}(\mathbf{l}) = \wedge_{i=1}^n \text{Inv}_i(\ell_i)$ *if* $\mathbf{l} = (\ell_1, \cdots, \ell_n)$.

This definition implies that if each $\mathcal{A}_i$ is bounded (resp. simple) then the NTA is bounded (resp. simple).

3 Symbolic Unfolding for Network of Timed Automata

In this section we define the symbolic semantics of a NTA in terms of *symbolic branching processes*. Those processes contain timing constraints both on places and events. We do not recall the definitions of *occurrence nets*, *branching processes (BP)* for untimed network of automata. The reader is referred to [10] for a detailed presentation of these notions.

Let $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$ be a synchronous product of TA. In a first step, we build the *untimed branching processes* (UBPs) of $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$. For each timed automaton $\mathcal{A}_i$ we let $\textsc{Untime}(\mathcal{A}_i)$ be the automaton obtained by removing all the timing constraints and clocks in $\mathcal{A}_i$. An UBP of a NTA is a BP of the network of untimed automata $(\textsc{Untime}(\mathcal{A}_1) | \ldots | \textsc{Untime}(\mathcal{A}_n))_I$ in the sense of [10]. The set of UBPs is defined inductively over two sets $\mathcal{E}$ and $\mathcal{P}$ by: i) $\perp \in \mathcal{E}$, ii) if $e \in \mathcal{E}$ and $s \in L$ then $(e, s) \in \mathcal{P}$, iii) if $S \subseteq \mathcal{P}$ and $t \in \lambda^{-1}(I)$ then $(S, t) \in \mathcal{E}$. On those two sets we define the mappings ${}^\bullet()$, ${}()^\bullet$:

[5] Any timed automaton can be transformed into an equivalent (behaviours) bounded automaton [2].

– for $\mathcal{E}$, ${}^\bullet\bot = \emptyset$, and if $e = (S, t)$, ${}^\bullet e = S$; and $e^\bullet = \{s \,|\, (e, s) \in \mathcal{P}\}$;
– for $\mathcal{P}$: ${}^\bullet(e, s) = e$ and $(e, s)^\bullet = \{e \,|\, {}^\bullet e \cap s \neq \emptyset\}$.

By definition of E and P a SBP is completely determined by E ans P as ${}^\bullet()$ and $()^\bullet$ are implicitly defined. Let x, y be two nodes (place or transitions). If $x \in {}^\bullet y$ or $y \in x^\bullet$ there is an *arc from* x to y and we write $x \to y$. This enables us to refer to the *directed graph of a net* which is simply the graph $(E \cup P, \to)$. The reflexive and transitive closure of $\to$ is denoted $\preceq$. x, y are *causally related* if either $x \preceq y$ or $y \preceq x$. x is in the (strict) *causal past* of y if $x \preceq y$ and $x \neq y$, *i.e.* $x \prec y$. x, y are in *conflict*, noted $x\#y$, if there is a place $p \in P$ such that $p \to w \preceq x$ and $p \to u \preceq y$ with $u \neq w$. x and y are *concurrent* if x and y are neither causally related nor in conflict. If J is a set of events then $\uparrow J = \big(\cup_{e \in J} e^\bullet\big) \backslash \big(\cup_{e \in J} {}^\bullet e\big)$. For a set $J \subseteq E \cup P$ $\lceil J \rceil = \{e' \in E \cup P \,|\, e' \preceq e$ for some $e \in J\}$. A set of events J is *causally closed* if $\lceil J \rceil = J$. A *configuration* of a BP is a set of events $K \subseteq E$ which is causally closed and conflict-free. A set A is a *co-set* iff $A \subseteq \uparrow K$ where K is a configuration. A *cut* $S \subseteq P$ is a set of places which is a maximal co-set. To each configuration K, we can associate a unique cut $\uparrow K$ which is denoted $\textsc{Cut}(K)$. A place $p = (e, s) \in \mathcal{P}$ is a i-place if $s \in L_i$. We can define the union of two branching processes (E_1, P_1) and (E_2, P_2) component-wise on events and places. BPs are closed under countable union and the *unfolding* of $(\textsc{Untime}(\mathcal{A}_1)| \ldots |\textsc{Untime}(\mathcal{A}_n))_I$ is be the maximal branching process. The next two properties are taken from [10]:

Proposition 1. *Two i-places of a UBP are either causally related or in conflict.*

Proposition 2. *Let C be a cut of a UBP. C contains one i-place for each $1 \leq i \leq n$.*

Thus given a configuration K, $\textsc{Cut}(K)$ corresponds to a unique state of the product of untimed automata.

The *symbolic* branching processes of a NTA are built from the UBP. The intuition is that we associate with places and events a time variable. For an event e, the variable δ_e stands for the (global) time at which event e fired. For a place p, δ_p stands for the most recent (global) time for which a token was in p. We define $\delta(E \cup P)$ to be the set of variables $\{\delta_x \,|\, x \in E \cup P\}$. A *symbolic branching process (SBP)* (E, P, γ) of $(\mathcal{A}_1| \ldots |\mathcal{A}_n)_I$ is a UBP (E, P) of $(\textsc{Untime}(\mathcal{A}_1)| \ldots |\textsc{Untime}(\mathcal{A}_n))_I$ with $\gamma : E \cup P \to \mathcal{C}(\delta(E \cup P))$ a mapping that associates to each node a timing constraint. The constraint on a node x should only refer to variables in $\lceil x \rceil$.

The constraint $\gamma(x)$ is computed by rewriting the timing constraints of the NTA in terms of the variables δ_y for $y \in \lceil x \rceil$. For the event $\bot$ we just set $\delta_\bot = 0$ stating that the system started at time 0. On the example of Fig. 1(b), to compute the timing constraint $\gamma(U)$ we just rewrite the invariant $y \leq 3$ in terms of the firing times of the events in the past of place U: if the current (global) time at which a token is in U is δ_U we must have $x = \delta_U - \delta_\bot \leq 3$ *i.e.* $\delta_U \leq 3$. For event e_3, we must have $x \leq 2$ and the value of x is given by $\delta_{e_3} - \delta_{e_2}$ which yields $\delta_{e_3} - \delta_{e_2} \leq 2$. The result for the NTA of Fig. 1(a) is depicted on Fig. 1(b). The important point is that each constraint $\gamma(x)$ is entirely determined by x. Hence to each UBP (E, P) we can associate a unique SBP (E, P, γ). We can thus define the *symbolic unfolding* $\textsc{tbp}(\mathcal{A}_1| \ldots |\mathcal{A}_n)_I$ of $(\mathcal{A}_1| \ldots |\mathcal{A}_n)_I$ to be the symbolic branching process associated with the unfolding of $(\textsc{Untime}(\mathcal{A}_1)| \ldots |\textsc{Untime}(\mathcal{A}_n))_I$.

To define cuts for SBP we need to take into account the timing constraints: for instance in Fig. 1(b), $(0, A, U)$ is a cut iff $\delta_0 = \delta_A = \delta_U \leq 3$ meaning that the global time in each place is the same and the constraints on the places are satisfied. For an event the same strategy applies. We can define a formula that characterizes all the timed cuts of a SBP:

Definition 5. *(M, Φ) is a* symbolic co-set *of (E, P, γ) if: 1) M is a co-set of (E, P), 2) $\Phi = \Phi_1(M) \wedge \Phi_2(M) \wedge \Phi_3(M) \wedge \Phi_4(M)$ with:*

$$\Phi_1(M) = \bigwedge_{x \in \lceil M \rceil} \gamma(x) \quad (1) \qquad \Phi_3(M) = \bigwedge_{p \in M} \big(\delta_{\bullet p} \leq \delta_p\big) \quad (3)$$

$$\Phi_2(M) = \bigwedge_{e \in \lceil M \rceil \cap E} \big(\wedge_{p \in \bullet e} \delta_p = \delta_e\big) \quad (2) \qquad \Phi_4(M) = \big(\bigwedge_{p, p' \in M} \delta_p = \delta_{p'}\big) \quad (4)$$

If M is a cut of (E, P), (M, Φ) is a symbolic cut. The meaning of formula (2) is that the last date δ_p at which a token was in p is the time at which an event removed a token in p. (3) imposes that if a token is in p and p is in a co-set, the current time in p which is δ_p is larger than the date of occurrence of the event that put a token in p. Finally (4) requires that all the places in the co-set have reached the same global time. The reason why we need to use variables associated with places is because there is no urgency in NTA. Notice that the formula Φ of a symbolic co-set is entirely determined by the co-set M and unique; we denote it by Φ_M. Moreover the form of the constraints on $\delta(E \cup P)$ in the SBP is such that Φ_M is a zone for each symbolic cut M:

Theorem 1. *For each symbolic cut (M, Φ_M) Φ_M is a zone.*

Given a SBP (E, P, γ), a set $M \subseteq P$, and a mapping $\Theta : \delta(\lceil M \rceil) \to \mathbb{R}_{\geq 0}$ that associates with each node a date, (M, Θ) is a *timed cut* iff (M, Φ_M) is a symbolic cut and $\Theta \in [\![\Phi_M]\!]$. Given a timed cut (M, Θ) we can associate a unique state of the NTA $\mathrm{GS}(M, \Theta)$: it suffices to compute the values of each clock variables in X from the values of the nodes variables in the SBP. Conversely, given a state $(\mathbf{l}, v)$ of the product, we can associate a timed cut to $(\mathbf{l}, v)$ as stated by Theorem 3 below.

Theorem 2. *If K is a configuration of $\mathrm{TBP}(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$ and $\Theta \in [\![\Phi_{\mathrm{CUT}(K)}]\!]$ then a) $\mathrm{GS}(\mathrm{CUT}(K), \Theta) = (\mathbf{l}, v)$ for some $(\mathbf{l}, v)$ reachable in $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$, and b) if $K \cup \{e\}$ is a configuration and $\Theta \in [\![\Phi_{\mathrm{CUT}(K)} \wedge \gamma(e) \bigwedge \big(\wedge_{p \in \bullet e} \delta_p = \delta_e\big)]\!]$ then $(\mathbf{l}, v) \xrightarrow{\lambda(e)} (\mathbf{l}', v')$ with $\mathrm{GS}(K \cup \{e\}, \Theta') = (\mathbf{l}', v')$ and $\Theta'_{|\mathrm{CUT}(K)} = \Theta$ and $\Theta'(x) = \Theta(p)$ for some $p \in \mathrm{CUT}(K)$ otherwise.*

The formula $\Phi_{\mathrm{CUT}(K)} \wedge \gamma(e) \bigwedge \big(\wedge_{p \in \bullet e} \delta_p = \delta_e\big)$ asserts that the global time is the same in every automata which is also equal to the firing time of e and that the guard of the transition t holds.

Theorem 3. *Let $(\mathbf{l}, v)$ be a reachable state in $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$. There is a configuration K of $\mathrm{TBP}(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_I$ and $\Theta \in [\![\Phi_{\mathrm{CUT}(K)}]\!]$ s.t.: a) $\mathrm{GS}(\mathrm{CUT}(K), \Theta) = (\mathbf{l}, v)$, and b) if $(\mathbf{l}, v) \xrightarrow{t} (\mathbf{l}', v')$ there is a configuration $K \cup \{e\}$ s.t. $\lambda(e) = t$ and a valuation $\Theta' \in [\![\Phi_{\mathrm{CUT}(K \cup \{e\})}]\!]$ s.t. $\mathrm{GS}(K \cup \{e\}, \theta') = (\mathbf{l}', v')$.*

If a TBP $\mathcal{T}$ satisfies the conditions of Theorem 3, we say that $\mathcal{T}$ is *complete*. Theorem 2, corresponds to a *correctness* property. For network of finite untimed automata, complete and correct finite branching processes exist, and are called *complete finite prefixes* [16,10]. In the case of network of timed automata we can construct a finite complete prefix that preserves the reachability information of the simulation graph.

Theorems 3 and 2 have two consequences. They follow from the fact that each $\Phi_{\text{CUT}(K)}$ is a zone for a configuration K. This means that the set of valuations reachable by all the linearizations of the events in K defines a zone as well. In the symbolic unfolding we construct, we obtain one zone for all the linearizations of the events in K whereas in $SG((\mathcal{A}_1|\dots|\mathcal{A}_n)_I)$ they could be two distinct states for two different linearizations. The first consequence is that the union of the zones reachable by all the linearization in $SG((\mathcal{A}_1|\dots|\mathcal{A}_n)_I)$ is a zone. Indeed computing global states preserves zones. This result was obtained recently by Ramzi Ben Salah, Marius Bozga and Oded Maler in [3] and has useful consequences. Our framework gives an alternative proof of this result and accounts for it in terms of partial order. The second consequence is that finite complete prefixes exist for NTA.

Assume the two configurations K_1 and K_2 lead to the same symbolic state $\text{GS}(\text{CUT}(K_1), \Phi_{\text{CUT}(K_1)}) = \text{GS}(\text{CUT}(K_2), \Phi_{\text{CUT}(K_2)})$, then they have the same future. Thus we can discard the events that extend one of them, for instance the smallest w.r.t. the order $\ll$ defined as: $K_1 \ll K_2$ iff $\text{GS}(\text{CUT}(K_1), \Phi_{\text{CUT}(K_1)}) = \text{GS}(\text{CUT}(K_2), \Phi_{\text{CUT}(K_2)}) \wedge |K_1| < |K_2|$. As the simulation graph contains a finite number of (union of) zones and because each $\Phi_{\text{CUT}(K)}$ is a union of zones, we can not have an infinite number of different symbolic states. This allows us to construct a complete finite prefix by keeping only the events e such that there exists a configuration K that enables e and is minimal w.r.t. $\ll$. We let $\text{PREF}((\mathcal{A}_1|\dots|\mathcal{A}_n)_I)$ be the complete symbolic finite prefix obtained from $(\mathcal{A}_1|\dots|\mathcal{A}_n)_I$. So far we are able to answer the question whether a set of timed events is a timed configuration: given the set of events K and the valuation Θ we can check whether $\Theta \in [\![\Phi_{\text{CUT}(K)}]\!]$. What we would like to do is to check whether a set of events K can be extended to a configuration *i.e.* if $\uparrow(K)$ is a co-set. We cannot do this directly with the SBP we have constructed so far. In the next section we refine our unfolding so that we do not need to look at the global state of the system to decide whether a set of events can be extended to a timed configuration.

4 Extended Finite Complete Prefixes

In the case of finite automata, any cut containing a co-set that enables an event, still enables the same event. This is not the case for network of timed automata as can be seen on the example of Fig. 1(b). If e_2 has not fired, e_1 can fire because nothing can prevent it from doing so (e_3 is not enabled). The fact that e_2 has not fired can be inferred from the fact that either place A or U contains a token. But this implies that the date δ_{e_1} at which e_1 fires satisfies $\delta_{e_1} \leq 3$. If e_2 has fired at δ_{e_2}, e_3 and e_1 are in conflict. Thus e_1 can only occur at a date when a token can be in B, *i.e.* to fire we must have $\delta_B = \delta_{e_1}$ and the constraint on the date at which a token can be in B which is $\delta_B - \delta_{e_2} \leq 2$. This implies $\delta_{e_1} - \delta_{e_2} \leq 2$. Thus the timing constraints associated with e_1 are not the same in the cuts $(0, A, U)$ and $(0, B, V)$ although they are both cuts that contain ${}^\bullet e_1$.

To encode this timing dependency structurally we can use symbolic occurrence nets with *read arcs*. For instance the symbolic net of Fig. 1(b) can be "transformed" into the symbolic extended net of Fig. 2 (a read arc is a dash line). Read arcs enable us to point to the missing timing information in the net that is needed to ensure an event can fire. This also means that we duplicate the event e_1 into e_1 and e_1' because the constraints are different depending on whether e_2 has occurred or not. Read arcs enlarge the *causal past* of the events. In the extended occurrence net, the constraint between the dates of occurrence of e_1 and e_2 can be inferred from the past of e_1: indeed, to fire, we must have $\delta_{e_1} = \delta_B$ and thus $\delta_{e_1} - \delta_{e_2} \leq 2$. Read arcs enable us to differentiate the two cuts $(0, A, U)$ and $(0, B, V)$ that generate different timing constraints on e_1 and e_2.

Extended Branching Processes. An *extended net* $\mathcal{N}$ is a tuple $(E, P, {}^\bullet(), ()^\bullet, {}^\circ())$ where $(E, P, {}^\bullet(), ()^\bullet)$ is a net, and ${}^\circ() : E \to 2^P$. If ${}^\circ e = \emptyset$ for each $e \in E$ then $\mathcal{N}$ is a net. The set ${}^\circ e$ represents the input places of an event that are to be read without removing a token. The *Extended* symbolic branching processes (ESBP) of a network are defined as in section 3: the only change we need to do is to define the set of events so that it includes the *read-only* places of an event denoted ${}^\circ e$. To this end, if $S, S' \subseteq \mathcal{P}$ and $t \in T$, (S, S', t) is in $\mathcal{E}$ and if $e = (S, S', t)$, ${}^\circ e = S'$.

The causality relation is now defined by: $x \to y$ if $x \in {}^\bullet y \cup {}^\circ y$ or $y \in x^\bullet$. $\preceq$ is the reflexive and transitive closure of $\to$. The *weak* causality relation $\dashrightarrow$ is given by: $x \dashrightarrow y$ if either $x \to y$ or ${}^\circ x \cap {}^\bullet y \neq \emptyset$ (if x needs a token in one of the input place of y this implies a causality relation, even if x is not in the past of y in the sense of $\to$.). We let $\trianglelefteq$ the reflexive and transitive closure of $\dashrightarrow$. Two nodes x and y are *weakly causally related* if either $x \trianglelefteq y$ or $y \trianglelefteq x$. x and y are in conflict, $x \# y$, if there is a place p s.t. there exist w and u, $w \neq u$, $p \in {}^\bullet u \cap {}^\bullet w$ and $w \trianglelefteq x$ and $u \trianglelefteq y$. x and y are concurrent if they are not weakly causally related nor in conflict. For $J \subseteq E \cup P$, the definitions of $\uparrow J$ and $\lceil J \rceil$ are unchanged (we use the new $\preceq$). A set of events is now causally closed if $\lceil J \rceil = J$. Co-sets, configurations and cuts are defined as before.

Safe Co-sets. Let $\textsc{Enable}(e)$ denote the *enabling cuts* of $e \neq \perp$ in a finite symbolic branching process $\mathcal{N}$: $\textsc{Enable}(e) = \{C \mid {}^\bullet e \subseteq C \textit{ and } C \textit{ is a cut of } \mathcal{N}\}$. As a running example we take the prefix $\mathcal{N}_1$ built in Fig. 1(b) and $\delta_\perp$ is always replaced by 0 (zero). For this example the enabling cuts are: $\textsc{Enable}(e_1) = \{(0, A, U), (0, B, V)\}$, $\textsc{Enable}(e_2) = \{(0, A, U), (1, A, U)\}$, $\textsc{Enable}(e_3) = \{(0, B, V)\}$.

Now assume an event e is in conflict with another event e' in the symbolic unfolding. As we pointed out at the end of section 3, the timing constraints given by $\lceil {}^\bullet e \rceil$ on the firing time of e do not always contain enough information to ensure event e can fire: event e_1 in $\mathcal{N}_1$ can fire if a) e_2 has not fired (this must be at time $\delta \leq 3$), or b) e_2 has fired, and the time elapsed since it has occurred is less than 2 time units (*i.e.* at time δ with $\delta - \delta_{e_2} \leq 2$), or c) e_2 has been disabled by another event in conflict with it and cannot occur in the future. To ensure e can fire, we should add to the conditions in ${}^\bullet e$ some information about the events in conflict with e. This is the purpose of *safe co-sets*. They extend the co-sets of the symbolic unfolding with some information about the conflicting events. In terms of occurrence nets, a safe co-set for an event e will be the set of places ${}^\bullet e$, extended with a set a *read only* places, ${}^\circ e$. The information contained

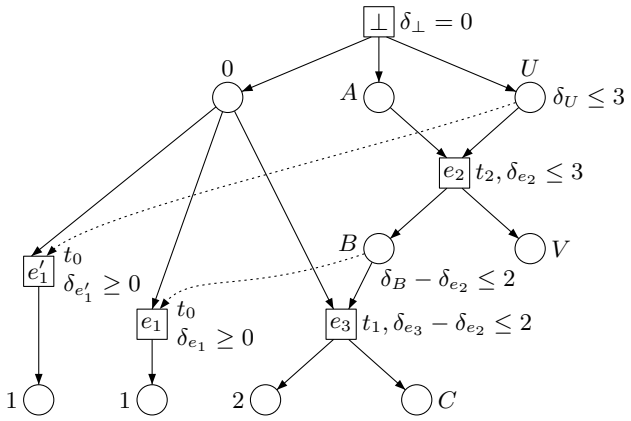


Figure 2. Extended symbolic unfolding for the example of Fig. 1(a)

in a safe co-set should be such that, if the timing constraints obtained by $\Phi_{\bullet e \cup \circ e}$ are satisfied, then there is a cut $C \supseteq {}^\bullet e \cup {}^\circ e$ s.t. Φ_C is satisfied.

For any cut C, the formula Φ_C (Def. 5, equations (1)–(4)) is a formula over $\delta(C \cup (\lceil C \rceil \cap E))$. Indeed all the intermediate places p, not in the cut, are constrained by a formula of the form $\delta_e = \delta_p$ because of equation 2 of Def. 5. For instance $\Phi_{(0,B,V)} = \delta_B - \delta_{e_2} \leq 2 \wedge 0 \leq \delta_{e_2} \leq 3 \wedge \delta_B \geq \delta_{e_2} \wedge \delta_0 = \delta_B = \delta_V$.

Because of the term Φ_4, if we use an extra variable δ and the formula $(\delta = \delta_p) \wedge \Phi_C$ for any[6] $p \in C$, we obtain a formula over $\delta(\lceil C \rceil \cap E) \cup \{\delta\}$: δ stands for the current global time (since the system started) and the constraint on δ in Φ_C defines the set of instants for which the cut C is *reachable i.e.* there are tokens in each place $p \in C$. We write Φ_C^δ for the projection on $(\lceil C \rceil \cap E) \cup \{\delta\}$ of the formula $\Phi_C \wedge (\delta = \delta_p)$. In our example, $\Phi^\delta_{(0,A,U)} = \delta \leq 3$, $\Phi^\delta_{(1,A,U)} = \delta_{e_1} \leq 3 \wedge \delta_{e_1} \leq \delta \leq 3$ and $\Phi^\delta_{(0,B,V)} = \delta - \delta_{e_2} \leq 2 \wedge 0 \leq \delta_{e_2} \leq 3 \wedge \delta \geq \delta_{e_2}$. This last example is interesting because it shows that the set of dates s.t. a token is in $(0, B, V)$ depends on the time at which e_2 occurred. Finally we let $\Theta(e) = \{\Phi_C^\delta \mid C \in \textsc{Enable}(e)\}$ and in the previous example, we obtain: $\Theta(e_1) = \{\Phi^\delta_{(0,A,U)}, \Phi^\delta_{(0,B,V)}\}$, $\Theta(e_2) = \{\Phi^\delta_{(0,A,U)}, \Phi^\delta_{(1,A,U)}\}$, $\Theta(e_3) = \{\Phi^\delta_{(0,B,V)}\}$ $\Theta(e)$ represents the set of different constraints that can be generated by all the enabling cuts of event e.

Definition 6. *A set of places S is a* safe representative *of a pair (e, C) where $e \in E$ and $C \in \textsc{Enable}(e)$ if 1) ${}^\bullet e \subseteq S \subseteq C$ and 2) for all $\nu : \delta(\lceil C \rceil \cap E) \cup \{\delta\} \to \mathbb{R}_{\geq 0}$ if $\nu_{|\delta(\lceil S \rceil \cap E) \cup \{\delta\}} \in [\![\Phi_S^\delta]\!]$ and $\nu_{|\delta(\lceil C \setminus S \rceil \cap E) \cup \{\delta\}} \in [\![\Phi^\delta_{C \setminus S}]\!]$ then $\nu \in [\![\Phi_C^\delta]\!]$. S is a safe representative of e if S is a safe representative of each pair (e, C) with $C \in \textsc{Enable}(e)$.*

If S is a safe representative of (e, C), then if $\gamma(e)$ holds together with Φ_S, e can be added to the unfolding. For example, $(0, A)$ is not a safe representative of $(0, A, U)$

[6] As equation (4) already imposes $\delta_{p'} = \delta_p$ for $p, p' \in C$ we can add $\delta = \delta_p$ for any p in C.

because $\Phi^{\delta}_{(0,A)} = \delta \geq 0$ and $\Phi^{\delta}_{(0,A,U)} = \delta \leq 3$. $(0,U)$ is a safe representative of $(0,A,U)$ as well as $(0,A,U)$ itself. $(0,B)$ is a safe representative of $(0,B,V)$. As each cut $C \in \text{ENABLE}(e)$ is a safe representative of itself, there is always one safe representatives for any C which is $\text{ENABLE}(e)$. We can state a theorem which is a variant of Theorem 2 using only safe representatives of an event (item *b)* of the theorem is altered):

Theorem 4. *If K is a configuration of $\text{TBP}(\mathcal{A}_1|\ldots|\mathcal{A}_n)_I$ and $\Theta \in [\![\Phi_{\text{CUT}(K)}]\!]$ then a) $\text{GS}(\text{CUT}(K),\Theta) = (\mathbf{l},v)$ for some $(\mathbf{l},v)$ reachable in $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_I$, and b) if $K \cup \{e\}$ is a configuration and S is a safe representative of $\text{CUT}(K)$ and $\Theta \in [\![\Phi_S \wedge \gamma(e) \bigwedge(\wedge_{p\in{}^\bullet e}\delta_p = \delta_e)]\!]$ then $(\mathbf{l},v) \xrightarrow{\lambda(e)} (\mathbf{l}',v')$ with $\text{GS}(K\cup\{e\},\Theta') = (\mathbf{l}',v')$ and $\Theta'_{|\text{CUT}(K)} = \Theta$ and $\Theta'(x) = \Theta(p)$ for some $p \in \text{CUT}(K)$ otherwise.*

This theorem is a direct consequence of Theorem 2 and Def. 6. It states that a safe representative for e contains enough information to decide whether event e can be fired or not. As a consequence, if whenever we add a new event e to a (finite) extended symbolic branching process of a NTA $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_I$, we use a safe representative $S = {}^\bullet e \cup {}^\circ e$ and add *read-arcs* to the places of ${}^\circ e$, then $\lceil e \rceil$ (including $\lceil S \rceil$) gives the accurate constraints on the date δ_e at which e can fire.

To build an extended complete finite prefix for a NTA we can proceed as follows: 1) build the symbolic net defined in section 3; this enables us to obtain the safe co-sets for each event; 2) build an extended net by adding an event to the unfolding using safe co-sets instead of simple co-sets. On the example of Fig. 1 this gives the unfolding of Fig. 2:

1. start with places $0, A, U$ and event $\perp$;
2. to add an event labelled t_0 use a safe co-set: we choose $(0,U)$ and add event e'_1 with a read arc to U;
3. add e_2 and e_3;
4. now a new safe co-set has appeared: $(0,B)$; we can add an event e_1 labelled by t_0 with a read arc from place B.

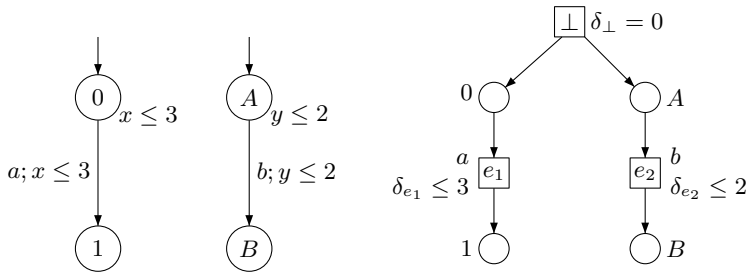
This construction can be formally defined (see [7]). The result is a finite extended symbolic complete prefix $\text{EPREF}((\mathcal{A}_1|\ldots|\mathcal{A}_n)_I)$ that satisfies property (P'). Formally, we define *symbolic configurations*. Assume $\text{EPREF}((\mathcal{A}_1|\ldots|\mathcal{A}_n)_I) = (E,P,\gamma)$.

Definition 7. *(K,Ψ) is a* symbolic configuration *of (E,P,γ) if: 1) K is a configuration of (E,P), and 2) $\Psi = \Psi_1(K) \wedge \Psi_2(K)$ where $\Phi_i(M), 1 \leq i \leq 2$ are defined by:*

$$\Psi_1(K) = \bigwedge_{e\in\lceil K\rceil} \gamma(e) \quad (5) \quad \textit{and} \quad \Psi_2(K) = \bigwedge_{e\in K} \left(\wedge_{p\in{}^\bullet e\cup{}^\circ e}\delta_p = \delta_e\right) \quad (6)$$

Notice that Ψ uses only information in the past of K and is uniquely determined thus we can write it Ψ_K. Let $\nu : K \to \mathbb{R}_{\geq 0}$. (K,ν) is a *timed configuration* if $\nu \in [\![\Psi_K]\!]$.

Theorem 5. *If (K',Ψ') is a symbolic configuration of $\text{EPREF}((\mathcal{A}_1|\ldots|\mathcal{A}_n)_I)$ and $\Theta' \in [\![\Psi']\!]$ then: there exists a symbolic configuration (K,Ψ) with $K \supseteq K'$ and $\Theta \in [\![\Psi]\!]$ s.t. 1) $\text{GS}(\text{CUT}(K),\Theta) = (\mathbf{l},v)$ for some $(\mathbf{l},v)$ reachable in $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_I$, and*



(a) Two Independent Automata (b) The Unfolding

Figure 3. A Network of two Independent Timed Automata

2) if $(K' \cup \{e\}, \Psi'')$ *is a symbolic configuration and* $[\![\Psi'']\!] \neq \emptyset$ *then* $(\mathbf{l}, v) \xrightarrow{\lambda(e)} (\mathbf{l}', v')$ *and* $\mathrm{GS}(\textsc{Cut}(K' \cup \{e\}), \Theta') = (\mathbf{l}', v')$ *for some* $\Theta' \in [\![\Psi'']\!]$.

On the example of Fig. 1(a), $\{(\perp, 0), (e_1, \delta_{e_1}), (e_2, \delta_{e_2})\}$ is a timed configuration iff $\delta_{e_1} - \delta_{e_2} \leq 2$ and $\delta_{e_2} \leq 3$.

Minimality for Safe Co-sets. The purpose of unfoldings is to keep explicit the concurrency of events. In the case of untimed network of automata, ${}^\bullet e$ is sufficient to ensure e can fire. For NTA, we have to use read arcs, but we should be concerned about the number of the new dependencies: for instance, if we use $\textsc{Enable}(e)$ as the set of safe representatives for each e, we require that the global state of the network is known each time we want to fire e. This means we do not keep explicit any concurrency in the unfolding. It is thus important to try and reduce the number of read arcs from each event. To this extent we define a notion of *minimality* for safe representatives.

We can define a partial order $\sqsubseteq$ on co-sets *i.e.* sets of places using the cardinality of the sets: $C_1 \sqsubseteq C_2$ iff $|C_1| \leq |C_2|$. For each $C \in \textsc{Enable}(e)$ we can take one minimal element in the set of safe representatives of C. Given $e \in E$, $\textsc{Safe}(e)$ denotes a set of minimal safe representatives, one for each $C \in \textsc{Enable}(e)$. In the example for $\mathcal{N}_1$ we can take the sets: $\textsc{Safe}(e_1) = \{(0, U), (0, B)\}$, $\textsc{Safe}(e_2) = \{(A, U)\}$, $\textsc{Safe}(e_3) = \{(0, B)\}$. For the independent automata of Fig. 3(a), we obtain that 0 is a safe representative of e_1 (in Fig. 3(b)): indeed 0 is a safe representative of $(0, A)$ and a safe representative of $(0, B)$ which belongs to $\textsc{Enable}(e_1)$. For the NTA given by Fig. 3(a) we obtain the unfolding of Fig. 3(b).
The minimality criterion we have defined does not give a unique set of safe representatives. A consequence is that there is no smallest complete finite prefix for a NTA but rather a set of set of minimal complete finite prefixes. Moreover as we take at least one safe representative for each pair (e, C) the branching process we build is still complete.

Checking Validity of Timed Configuration. To complete the construction and provide a solution to the problem of checking whether a timed configuration is valid, we can define the constraint $\Gamma(e)$ associated with an event e by: $\Gamma(e) = \Psi(\lceil e \rceil)_{|\lceil e \rceil \cap E}$. This

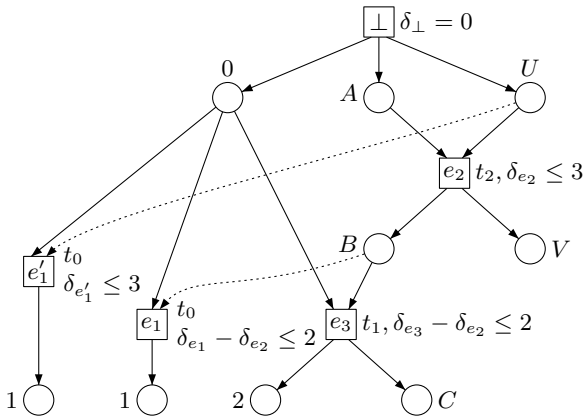


Figure 4. Reduced Extended symbolic unfolding for the example of Fig. 1(a)

constraint gathers the constraints of all the past events. The branching process obtained this way is a *reduced* branching process with only constraints on events. For the network of timed automata of Fig. 1(a), the reduced branching process is given on Fig. 4. It enables us to decide whether a closed set of events K is a prefix of an extended symbolic branching process.

5 Conclusion

In this paper we have defined a model, *extended symbolic branching process*, to define the concurrent semantics of timed systems. We have also proved that each NTA admits a *finite complete prefix* which is a symbolic extended branching process, and we have given an algorithm to compute such a prefix. Other interesting results are: 1) there is no unique complete finite prefix for a NTA but rather a set of complete finite prefixes; 2) building a *small* (optimal) complete finite prefix is very expensive as it requires the computation of information spread across the network; and 3) we have pointed out the difficulties arising in the construction of such a prefix, namely the need for *safe co-sets*. Our future work will consist in: a) define heuristics to determine when an event can be added to a prefix of an unfolding; this means having an efficient way of computing safe representatives, which are no more guaranteed to be minimal; b) when step 1 is developed, we can define algorithms to check properties of the NTA using the unfolding and assess the efficiency of these algorithms.

References

1. Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.
2. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In

Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01), volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

3. Ramzi Ben Salah, Marius Bozga, and Oded Maler. On interleaving in timed automata. In *Proceedings of the* 17th *International Conference on Concurrency Theory (CONCUR'06)*, LNCS, aug 2006. To appear.
4. Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. In Paul Pettersson and Wang Yi, editors, *Proceedings of the third International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 211–225, Uppsala, Sweden, September 2005. Springer.
5. Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.
6. Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.
7. Franck Cassez, Thomas Chatain, and Claude Jard. Symbolic Unfoldings for Networks of Timed Automata. Technical Report RI-2006-4, IRCCyN/CNRS, Nantes, May 2006.
8. Franck Cassez and Olivier H. Roux. Structural translation from time petri nets to timed automata. *Journal of Systems and Software*, 2006. forthcoming.
9. Thomas Chatain and Claude Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In *ICATPN*, volume 4024 of *LNCS*, pages 125–145, june 2006.
10. Javier Esparza and Stefan Römer. An unfolding algorithm for synchronous products of transition systems. In *CONCUR*, volume 1664 of *LNCS*, pages 2–20. Springer, 1999.
11. Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
12. Hans Fleischhack and Christian Stehno. Computing a finite prefix of a time Petri net. In *ICATPN*, pages 163–181, 2002.
13. J. Bengtsson, B. Jonsson, J. Lilius, W. Yi. Partial order reductions for timed systems. In *CONCUR 99*, volume 1466 of *LNCS*, pages 485–500, 1999.
14. Kim G. Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In *Proc. 18th IEEE Real-Time Systems Symposium (RTSS'97)*, pages 14–24. IEEE Computer Society Press, 1997.
15. Denis Lugiez, Peter Niebert, and Sarah Zennou. A partial order semantics approach to the clock explosion problem of timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 296–311. Springer, 2004.
16. Kenneth L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
17. P.M. Merlin and D.J. Farber. Recoverability of communication protocols – implications of a theorical study. *IEEE Transactions on Communications*, 24, 1976.
18. M. Minea. Partial order reduction for model checking of timed automata. In *CONCUR 99*, volume 1664 of *LNCS*, pages 431–446, 1999.
19. T. Aura and J. Lilius. A causal semantics for time petri nets. *Theoretical Computer Science*, 1–2(243):409–447, 2000.
20. T. Yoneda, B-H. Schlingloff. Efficient verification of parallel real-time systems. *Formal Methods in System Design*, 2(11):187–215, 1997.
21. A. Valmari. Stubborn sets for reduced state space generation. In *Applications and Theory of Petri Nets*, volume 483 of *LNCS*, pages 491–515, 1989.
22. W. Belluomini, C. J. Myers. Verification of timed systems using posets. In *CAV 98*, volume 1427 of *LNCS*, pages 403–415, 1998.