Timed Unfoldings for Networks of Timed Automata

Patricia Bouyer[1], Serge Haddad[2], Pierre-Alain Reynier[1]

[1] LSV, CNRS & ENS Cachan, France
[2] LAMSADE, CNRS & Université Paris-Dauphine, France
{bouyer,reynier}@lsv.ens-cachan.fr, haddad@lamsade.dauphine.fr

Abstract. Whereas partial order methods have proved their efficiency for the analysis of discrete-event systems, their application to timed systems remains a challenging research topic. Here, we design a verification algorithm for networks of timed automata with invariants. Based on the unfolding technique, our method produces a branching process as an acyclic Petri net extended with *read arcs*. These arcs verify conditions on tokens without consuming them, thus expressing concurrency between conditions checks. They are useful for avoiding the explosion of the size of the unfolding due to clocks which are compared with constants but not reset. Furthermore, we attach *zones* to events, in addition to markings. We then compute a complete finite prefix of the unfolding. The presence of invariants goes against the concurrency since it entails a global synchronization on time. The use of read arcs and the analysis of the clock constraints appearing in invariants helps increasing the concurrency relation between events. Finally, the finite prefix can be used to decide reachability properties, and transition enabling.

1 Introduction

Partial-order methods for discrete-event systems. In the last decades, major advances in the analysis of distributed systems were based on two paradigms: the *independence* and the *locality* of actions. Whereas *partial-order* methods mainly take advantage of the independence (see e.g. [20]), the *unfolding* methods rely on both concepts [13, 17]. Furthermore from a semantical point of view, system unfoldings are a theoretical well-defined alternative to the usual interleaving semantics. It must be emphasized that this semantics is more discriminant than the classical one and may be applied for other purposes than verification like observation and diagnosis (see e.g. [9]).

Timed systems. Several timed models have been proposed for representing real-time systems, e.g. various extensions of Petri nets, but the most studied and well-established model is the one of timed automata (TA for short). It has been defined in [1] and since then much investigated, with the development of several tools based on this model.

Partial-order methods for timed systems. If this approach led to efficient tools and algorithms in the untimed case, no counterpart has so far been achieved for *timed systems*. The main reason is that time synchronization of actions in the standard timed models is essentially *global* and thus yields numerous conceptual and technical difficulties for adapting or extending the previous methods. We discuss in Section 5 existing works.

Our contribution. In this paper, we design an efficient verification algorithm for *networks of timed automata with invariants* (NTA). Our algorithm is based on the unfolding technique, and produces an acyclic Petri net with *read arcs*. Conditions (*i.e.* places of the net) are labeled either by locations or by clocks, and events (*i.e.* transitions of the net) represent the transitions of the NTA. Read arcs are convenient for modeling clock testing with no clock reset (see for instance [7]), and, though they add some complexity to the building of the unfoldings [21, 22], they increase the independence relation between events.

More precisely, we define a timed unfolding of an NTA close to the untimed case, by attaching *zones* (a classical symbolic representation in the framework of timed systems) to events, in addition to markings. Roughly the zone attached to an event t will capture all relevant timing informations of possible configurations reached after having fired all events belonging to the minimal causal past of t. It must be emphasized that the dimension of the zones that we attach to events is small (and constant while the NTA is unfolded): it is equal to three times the number of clocks plus twice the number of TA.

The main problem encountered by previous works is that urgency requirements (for instance due to invariants) entail global synchronization between *a priori* independent transitions. When a clock appears in an invariant, we use read arcs to express dependencies of the transitions w.r.t. this invariant. This increases the concurrency relation between events, even in the presence of invariants and enables a local decision of the firability of an event (*i.e.* only by looking at its cut).

Finally, we prove that we can build a complete finite prefix which can be used, as in the untimed case, for deciding in linear time (w.r.t. the size of the finite prefix) reachability (as well as transition firing) properties in NTA.

Due to lack of space, proofs are omitted, but can be found in [8]

2 Networks of Timed Automata

Let X be a finite set of variables, called *clocks*. We write $\mathcal{C}(X)$ for the set of *constraints* over X, which consist of conjunctions of atomic formulae of the form $x \bowtie c$ and $x - y \bowtie c$ for $x, y \in X$, $c \in \mathbb{Z}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. We write *Clocks*$(\gamma)$ for the set of clocks involved in γ. We define the proper subset $\mathcal{C}_{df}(X)$ of *diagonal-free* constraints over X where constraints $x - y \bowtie h$ (called *diagonal constraints*) are not allowed. Similarly, we define the proper subset $\mathcal{C}_{ub}(X)$ of *upper-bounded* constraints over X where only constraints $x \prec h$ with $\prec \in \{<, \leq\}$ are allowed.

Let s be a mapping from X to elementary expressions over some set X' (*i.e.* x, $x-y$ or $x - c$). Then the substitution of s in a diagonal-free constraint γ, denoted $\gamma[\{x \leftarrow s(x)\}_{x\in X}]$ is defined as the expression obtained by replacing in γ every occurrence of x by the term $s(x)$, for any clock x. Note that the resulting expression belongs to $\mathcal{C}(X')$.

We will use as timed domain the set $\mathbb{R}_{\geq 0}$ of nonnegative real numbers. A *valuation* over the set X of clocks is an element of $\mathbb{R}_{\geq 0}^X$. For $R \subseteq X$, the valuation $v[R \leftarrow 0]$ is the valuation v' such that $v'(x) = 0$ when $x \in R$ and $v'(x) = v(x)$ otherwise. For $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is defined by $(v + d)(x) = v(x) + d$ for every $x \in X$. Constraints of $\mathcal{C}(X)$ are interpreted in a natural way over valuations: we write $v \models \gamma$ when the constraint γ is satisfied by v.

We use the classical notion of zones to represent symbolically infinite sets of valuations [12]. A *zone* over a set of variables Y is defined as a constraint of $\mathcal{C}(Y)$. We assume the reader to be familiar with the following operations on zones (see [6]): conjunction, extension of the set of variables, elimination of a set of variables (we write $\exists V.Z$), and emptiness checking. The *extrapolation* of zone Z w.r.t. constant M is the smallest zone containing Z defined with constants in $\{-M, \ldots, 0, \ldots, M\}$.

Definition 1 (Timed Automaton (TA) [1]). *A* timed automaton $\mathcal{A}$ *over* Σ *is a tuple* $(L, \ell_0, X, \Sigma, E, \mathit{Inv})$ *where* L *is a finite set of* locations, $\ell_0 \in L$ *is the* initial location*,* X *is a finite set of* clocks*,* Σ *is a finite alphabet of actions,* $E \subseteq L \times \mathcal{C}_{df}(X) \times \Sigma \times 2^X \times L$ *is a finite set of* edges *and* $\mathit{Inv} \subseteq \mathcal{C}_{ub}(X)^L$ *associates to each location an* invariant *given as an upper bound constraint. An edge* $(\ell, g, a, R, \ell') \in E$ *(or* $\ell \xrightarrow{g,a,R} \ell'$*) represents a transition from location* ℓ *to location* ℓ' *labeled by* a*, with the guard* g *defined by a constraint and reset* $R \in 2^X$.

Definition 2 (Network of TA (NTA)). *A partial function* $f : (\Sigma \cup \{\bot\})^n \to \Sigma$ *is called an* n*-ary* synchronization function*. A* network of timed automata *is a finite family* $(\mathcal{A}_i)_{1 \le i \le n}$ *of* n *TA, whose sets of locations are pairwise disjoint, together with an* n*-ary synchronization function* f.

Note that we do not assume that clocks are local to each TA of an NTA. Before giving the semantics of an NTA, we first give some notation and definitions which will be useful in the rest of the paper. We fix an NTA $\mathcal{A}$, and we assume that $\mathcal{A}$ is given by $(\mathcal{A}_i)_{1 \le i \le n}$, and f a synchronization function. We write $\mathcal{A}_i = (L_i, \ell_{i,0}, X_i, \Sigma, E_i, \mathit{Inv}_i)$ for every $1 \le i \le n$. We then denote by X (resp. L) the set $\bigcup_{1 \le i \le n} X_i$ (resp. $\bigcup_{1 \le i \le n} L_i$). We extend naturally the function *Inv* over the set L.

Finally, we consider a synchronization function $f : (\Sigma \cup \{\bot\})^n \to \Sigma$. In the sequel, we denote $\overline{\Sigma_\bot}$ (resp. $\overline{E}$) the set $(\Sigma \cup \{\bot\})^n$ (resp. the set $\prod_i (E_i \cup \{\bot\})$). We use a similar notation for their elements: we denote $\overline{a}$ (resp. $\overline{e}$) an n-uple $(a_1, \ldots, a_n) \in \overline{\Sigma_\bot}$ (resp. $(e_1, \ldots, e_n) \in \overline{E}$). We define the function *Lab* from $\overline{E}$ to $\overline{\Sigma_\bot}$ which maps an element $\overline{e}$ to the element $\overline{a}$ defined for every $1 \le i \le n$ by $a_i = b$ if $e_i = \ell_i \xrightarrow{g,b,R} \ell'_i$, and by $a_i = \bot$ otherwise. We define the subset $\mathit{Sync} = \mathit{Lab}^{-1}(f^{-1}(\Sigma))$ of $\overline{E}$, which is the set of possible synchronizations of edges, *i.e.* the set of transitions of the NTA. Given $\overline{e} \in \mathit{Sync}$, assuming $e_i = \ell_i \xrightarrow{g_i,a_i,R_i} \ell'_i$, for all i such that $e_i \neq \bot$, we define $I(\overline{e})$ the set $\{1 \le i \le n \mid e_i \neq \bot\}$, $g(\overline{e})$ the constraint $\bigwedge_{i \in I(\overline{e})} g_i$ and $R(\overline{e})$ the set $\bigcup_{i \in I(\overline{e})} R_i$. Finally, given an n-tuple $\overline{\ell}$, we note $\mathit{Inv}(\overline{\ell}) = \bigwedge_{1 \le i \le n} \mathit{Inv}(\ell_i)$.

Definition 3 (Semantics of an NTA). *Let* $\mathcal{A} = ((\mathcal{A}_i)_{1 \le i \le n}, f)$ *be an NTA. The semantics of* $\mathcal{A}$ *is the transition system* $S_{\mathcal{A}} = (Q, q_0, \to)$ *where* $Q = (\Pi_{1 \le i \le n} L_i) \times (\mathbb{R}_{\ge 0})^X$*,*[3] $q_0 = (\overline{\ell}_0, \mathbf{0})$ *and* $\to$ *is defined by:*

$$\begin{cases} (\overline{\ell}, v) \xrightarrow{d} (\overline{\ell}, v + d) & \text{if } d \in \mathbb{R}_{\ge 0} \text{ and } v + d \models \mathit{Inv}(\overline{\ell}) \text{ (delay moves);} \\ (\overline{\ell}, v) \xrightarrow{a} (\overline{\ell}', v') & \text{if } \exists \overline{e} \in \mathit{Lab}^{-1}(f^{-1}(\{a\})) \text{ s.t. } v \models g(\overline{e}), v' = v[R(\overline{e}) \leftarrow 0] \text{ and} \\ & \ell'_i \text{ is given by } e_i \text{ if } i \in I(\overline{e}) \text{ and by } \ell_i \text{ otherwise (discrete moves).} \end{cases}$$

Finally, an element $\sigma = (\bar{e}_i, d_i)_{i \ge 0} \in (\mathit{Sync} \times \mathbb{R}_{\ge 0})^*$ *is a* timed sequence *of* $\mathcal{A}$ *if the sequence of moves* $q_0 \xrightarrow{d_0} \ldots \xrightarrow{d_i - d_{i-1}} \xrightarrow{f(\mathit{Lab}(\bar{e}_i))} \ldots \xrightarrow{f(\mathit{Lab}(\bar{e}_n))}$ *is in* $\mathcal{S}_{\mathcal{A}}$.

[3] We denote $\overline{\ell}$ an n-tuple of $\Pi_{1 \le i \le n} L_i$, and $\overline{\ell}_0 = (\ell_{i,0})_{1 \le i \le n}$.

W.l.o.g. we assume that the constraints and resets associated with edges syntactically ensure that the invariants associated with the output locations of every edge are satisfied when a discrete move following that edge is performed.

Important and unusual definitions. We define several other notions, which will be fundamental for defining our unfolding. Let $\mathcal{A}$ be an NTA. Let X be its set of clocks, then X_{inv} is the subset of clocks occurring in the invariant of some location of L. Given an edge $\bar{e} = \bar{\ell} \xrightarrow{g,\bar{a},R} \bar{\ell}'$, and a clock $x \in X$, we say that x is *redefined* by $\bar{e}$ if x is not reset by $\bar{e}$, and if the constraints $\mathit{Inv}(\overline{\ell})$ and $\mathit{Inv}(\overline{\ell'})$ are not equivalent w.r.t. x. We denote by $\mathit{Redefined}(\bar{e})$ the set of clocks redefined by $\bar{e}$. Given a clock $x \in X$, we say that x is *modified* by $\bar{e}$ if $x \in R(\bar{e}) \cup \mathit{Redefined}(\bar{e})$. This means that x has either been reset by one of the edges, or an invariant constraint over x has been redefined. Moreover, we say that x is *tested* by $\overline{e}$ if $x \in \mathit{Clocks}(g(\bar{e})) \cup X_{inv}$. This means that the clock x is either tested in one of the constraints, or used in some invariant of the NTA. It is worth noticing that we include here the whole set X_{inv}. This latter point will be discussed later. Finally, we note:

$$\begin{cases} \mathsf{Pre}(\overline{e}) = \{\ell_i \mid i \in I(\overline{e})\} \cup \{x \in X \mid x \text{ is modified by } \overline{e}\} \\ \mathsf{Read}(\overline{e}) = \{x \in X \mid x \text{ is tested but not modified by } \overline{e}\} \\ \mathsf{Post}(\overline{e}) = \{\ell'_i \mid i \in I(\overline{e})\} \cup \{x \in X \mid x \text{ is modified by } \overline{e}\} \end{cases}$$

3 Unfoldings of NTA

3.1 Untimed Nets

We first define the untimed structures we use. These are classical structures defined e.g. in [17, 13], extended with read arcs [21, 22]. Even if read arcs do not add expressiveness to (untimed) Petri nets (w.r.t. reachability), they improve quite a lot unfolding techniques, since they increase the concurrency relation between events. However, their unfolding is more involved.

Definition 4 (Read Arc Petri Net). *A* read arc Petri net *is a tuple* $\mathcal{N} = (P, T, \mathit{Pre}, \mathit{Post}, \mathit{Read}, M_0)$ *where* P *is a (finite) set of* places*,* T *is a (finite) set of* transitions *with* $P \cap T = \emptyset$*, Pre, Post and Read are three mappings from* T *to* 2^P *called resp.* backward*,* forward *and* read *incidence mapping. Finally,* $M_0 \in 2^P$ *is the* initial *marking.*

The untimed structure associated with the unfolding of a NTA is a particular kind of read arc Petri net. Before giving the structure, we first define precedence, strong precedence and conflict relations between nodes of a net. We first give some notation. Let t be a transition and p be a place of a net $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathsf{Read}, M_0)$:

- $^\bullet t$ denotes the set $\mathsf{Pre}(t)$, $t^\bullet$ denotes the set $\mathsf{Post}(t)$, $^\circ t$ denotes the set $\mathsf{Read}(t)$,
- $^\bullet p$ denotes the set $\{t' \in T \mid p \in t'^\bullet\}$, $p^\bullet$ denotes the set $\{t' \in T \mid p \in {}^\bullet t'\}$.

We extend the notation to set of nodes as usual. We now define relations between nodes:

- Let $<$ (the *precedence relation*) be the minimal transitive relation over $P \cup T$ satisfying for every $t, t' \in T$, for every $p \in P$,
  if $p \in {}^\bullet t$ then $p < t$, if $t \in {}^\bullet p$ then $t < p$, if $p \in {}^\circ t$ and $p \in t'^\bullet$ then $t' < t$.
  We denote $\leq$ the reflexive closure of $<$.

- Let $\prec$ (the *stong precedence relation*) be the minimal transitive relation over $P \cup T$ satisfying for every $t, t' \in T$, for every $p \in P$, and for every nodes x and y, if $x < y$ then $x \prec y$, if $p \in {}^\circ t$ and $p \in {}^\bullet t'$ then $t \prec t'$.
  We denote $\preceq$ the reflexive closure of $\prec$.
- Let $\#$ (the *conflict relation*) be defined by $x \ \# \ y$ iff $\exists p \in P,\ \exists t, t' \in p^\bullet$ s.t. $t \neq t' \wedge t \leq x \wedge t' \leq y$.

These definitions are those given in [22] which are a slight variant of those in [21].

Definition 5 (Occurrence Net). *An* occurrence net *is a net $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathsf{Read}, M_0)$ fulfilling the following conditions. $|{}^\bullet p| \leq 1$ for every $p \in P$. The precedence relation $<$ of $\mathcal{N}$ is a finitary partial order (*i.e. *every item of $P \cup T$ has a finite number of predecessors). For every item $x \in P \cup T$, the strong precedence relation restricted to the set of predecessors of x w.r.t. $<$ is a partial order. No element is in conflict with itself. $M_0 = \mathit{Min}(P)$, where $\mathit{Min}(P)$ denotes the set $\{p \mid {}^\bullet p = \emptyset\}$.*

In an occurrence net, elements of P are called *conditions* and elements of T *events*. We define the branching process associated with an NTA as a labeled occurrence net:

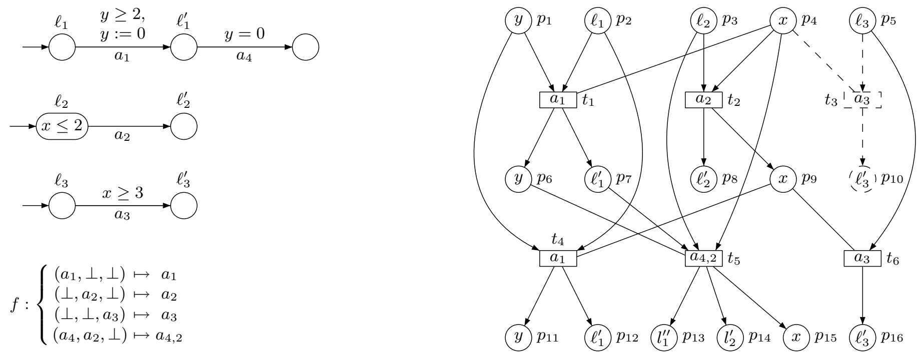
Definition 6 (Branching Process of an NTA). *Let $\mathcal{A}$ be the NTA given as a family $(\mathcal{A}_i)_{1 \leq i \leq n}$ of n TA and an n-ary function f. A* branching process of $\mathcal{A}$ *is defined as a pair of an occurrence net $\mathcal{N} = (P, T, \mathsf{Pre}, \mathsf{Post}, \mathsf{Read}, M_0)$ and a labeling function λ ranging over $P \cup T$ such that:*

- *$\lambda(P) \subseteq \bigcup_{1 \leq i \leq n}(L_i \cup X_i)$ (conditions correspond to locations or clocks of $\mathcal{A}$),*
- *$\lambda(T) \subseteq \mathit{Sync}$ (events correspond to possible transitions of $\mathcal{A}$),*
- *λ is a one-to-one mapping from M_0 to $\bigcup_{1 \leq i \leq n} \ell_{i,0} \cup X$ (initially, the marking consists in initial locations plus the clocks),*
- *for every element $t \in T$ with $\lambda(t) = \bar{e} \in \mathit{Sync}$, λ is a one-to-one mapping from ${}^\bullet t$ (resp. ${}^\circ t$, $t^\bullet$) to $\mathsf{Pre}(\bar{e})$ (resp. to $\mathsf{Read}(\bar{e})$, $\mathsf{Post}(\bar{e})$).*
- *$\forall t, t' \in T$, $\lambda(t) = \lambda(t') \wedge {}^\bullet t = {}^\bullet t' \wedge {}^\circ t = {}^\circ t' \Rightarrow t = t'$ (no redundancy)*

We use read-arcs in our unfoldings for increasing the concurrency relation between events: indeed, when firing a transition, there is no need to create a new place for a clock which is not modified, that's thus relevant to test its value using a read-arc, and not a pre-arc.

In [21, 22], a prefix relation is defined between branching processes of an NTA and it is shown that these processes form a complete lattice w.r.t. this relation which implies that there is a maximal branching process. The branching processes differ on "how much they unfold". The *untimed unfolding of an NTA* is defined as its maximal branching process.

Example 1. An example of branching process is depicted on Figure 1. Conditions are represented by circles, and events by boxes, as usual for Petri nets. Labels are written close to the nodes. A read arc is represented by an arc with no arrow (for instance there is a read arc from the top-most condition labeled x to the top-most event labeled a_1: for being fired, event a_1 will check that there is a token in condition x, since x is involved in an invariant). The dashed part of the branching process represents an event that will be considered by our algorithm but whose timing constraints are unconsistent, and thus which will not be built (see Subsection 3.2).



(a) A NTA $\mathcal{A}$ (b) A branching process of $\mathcal{A}$

Fig. 1. An example of branching process of an NTA.

We introduce more or less classical notions concerning branching processes. Note that these definitions take into account read arcs.

Definition 7 (Non-branching Process, Configuration, Cut, Causal Past). *Let $\beta = (\mathcal{N}, \lambda)$ be a branching process of an NTA $\mathcal{A}$. We write T (resp. P) for the set of events (resp. of conditions) of $\mathcal{N}$. We consider the occurrence net $(P', T') \subseteq (P, T)$ obtained as a restriction $\mathcal{N}'$ of $\mathcal{N}$, and the labeling function λ' defined as the restriction of λ to $\mathcal{N}'$. Then $\beta' = (\mathcal{N}', \lambda')$ is called a* non-branching process *of β if it satisfies the five following conditions:*

- $\forall t \in T,\ \forall p \in {}^\bullet t \cup {}^\circ t \cup t^\bullet,\ t \in T' \Rightarrow p \in P'$ *(events are consistent with β),*
- $\forall p \in P,\ \forall t \in {}^\bullet p,\ p \in P' \Rightarrow t \in T'$ *(conditions are consistent with β),*
- *Relation $\prec$ restricted to $P' \cup T'$ is a partial order,*
- $\forall x, y \in P' \cup T',\ \neg(x \# y)$ *($\mathcal{N}'$ is conflict-free),*
- *$Min(P') = Min(P)$.*

We fix a non-branching process β'. The configuration *C of β' is the set of events of β'. A set of conditions is a* co-set *if it is an antichain w.r.t. $\prec$ in β' (*i.e. *where items are pairwise incomparable). A* cut *is a maximal co-set. If C is the configuration of β', we associate with C the cut $Cut(C)$ defined by $Cut(C) = (Min(P) \cup C^\bullet) \setminus {}^\bullet C$. We also define the cut of a non-branching process as the cut of its configuration.*

Given a non-branching process β' of β, and an event t belonging to β', we denote $[t]_{\beta'}$ the causal past *of t relative to β' defined as the set of events $\{t' \in T' \mid t' \preceq t\}$. The minimal causal past [4] of t, denoted $[t]$, is $\bigcap_{\beta'} [t]_{\beta'}$ where β' ranges over the set of non-branching processes of β containing t. $[t]$ is a configuration and we denote by β_t its associated non branching process.*

Finally, we say that a non-branching process β^+ extends a non-branching process β, denoted by $\beta \sqsubseteq \beta^+$ if the events of β are events of β^+ and if given any event t of β and any event t^+ of $\beta^+ \setminus \beta$, we do not have $t^+ \prec t$ in β^+.

[4] Note that $[t]$ may be inductively defined by $[t] = \{t\} \cup \bigcup_{t' \in {}^\bullet({}^\bullet t \cup {}^\circ t)} [t']$. Due to the lattice structure of branching processes of a read arc Petri net, $[t]$ does not depend on β.

Example 1 continued. Let β be the branching process of Figure 1. Then the subgraph underlied by nodes $\{p_i\}_{i=1..9} \cup \{t_1, t_2\}$ is a non branching process (say β'); its associated configuration is $\{t_1, t_2\} = [t_2]_{\beta'} \neq [t_2] = \{t_2\}$. Let β_1 (resp. β_2) be the non branching process corresponding to $\{t_1, t_2, t_3\}$ (resp. $\{t_1, t_2, t_6\}$). Then $\beta' \subseteq \beta_i$ for $i = 1, 2$ and $\beta' \sqsubseteq \beta_2$ but $\beta' \not\sqsubseteq \beta_1$ due to the arc between p_4 and t_3 (implying $t_3 \prec t_2$).

Important remark. It is worth noticing that if C is a configuration of an NTA, the set $\mathit{Cut}(C) \cap \lambda^{-1}(X)$ is in bijection (by λ) with the set X of clocks of the NTA and that λ maps the set $\mathit{Cut}(C) \cap \lambda^{-1}(L)$ to a set consisting of one location per TA of the NTA. Indeed, each time a clock place is consumed, it is produced back and each time a place whose label is a location of a TA is consumed another place whose label is a location of the same TA is produced.

We use the notation of [21] to present the (semi-)algorithm (Algorithm 1) for the construction of the untimed unfolding of an NTA. In the algorithm, a condition of the unfolding is encoded as a pair (p, t) where p is the label of this condition, and t is the unique input event of this condition (t equals to $\emptyset$ if the condition has an empty preset). An event is represented with three fields $(\bar{e}, Y_{in}, Y_r)$ where $\bar{e}$ is the label of this event (a synchronized edge), Y_{in} and Y_r are two lists of pointers to conditions (respectively the input and read conditions).

Definition 8 (Possible Extensions (PE)). *Let $\beta = (\mathcal{N}, \lambda)$ be a branching process of an NTA $\mathcal{A}$. The* possible extensions *of β are the triples $t = (\bar{e}, Y_{in}, Y_r)$ where $\bar{e}$ is an element of Sync such that there exists a non branching process β' with $Y_{in} \cup Y_r$ being a co-set of β', such that λ is a one-to-one mapping from Y_{in} (resp. Y_r) to Pre($\bar{e}$) (resp. Read($\bar{e}$)), and such that $(\bar{e}, Y_{in}, Y_r)$ does not already belong to β.*

In this case, we define the extension of β by t, obtained by the operation Extend(β, t) as the branching process β' obtained from β by adding an event labeled by $\bar{e}$, connected to conditions in Y_{in} with pre-arcs and to conditions in Y_r with read arcs, and with new conditions, according to Post($\bar{e}$).

Algorithm 1 Building the (eventually infinite) untimed unfolding (semi-algorithm)

Require: An NTA $\mathcal{A}$.
Ensure: The unfolding *Unf* of $\mathcal{A}$.

Unf := $\{(\ell_{1,0}, \emptyset), \ldots, (\ell_{n,0}, \emptyset)\} \cup \{(x, \emptyset) \mid x \in X\}$; (Initialization)
$pe := PE(\mathit{Unf})$; (Possible Extensions)
while $pe \neq \emptyset$ **do**
  Choose an event $t = (\bar{e}, Y_{in}, Y_r)$ in pe. ($\bar{e}$ is the label of t)
  Extend(*Unf*, t);
  $pe := PE(\mathit{Unf})$;
end while

3.2 Adding Timing Constraints to the Untimed Unfolding

Our objective is to add timing information in the untimed structure described before for getting a new symbolic representation of the set of timed sequences of an NTA.

This will also reduce the size of the untimed structure, by removing extensions with unfeasible timed part (see the dashed part of Example 1).

Timed executions. In order to define and compute the timed unfolding of an NTA, we first add time to a non-branching process. We associate an absolute date, written $\mathbf{d}$, with every event corresponding to its occurrence and two or three dates with every condition. The first one corresponds to its production (or birth), written $\mathbf{d_b}$. The second date corresponds to the consumption (or end) of the condition (it may be $+\infty$), written $\mathbf{d_e}$. A third date is associated with a condition corresponding to a clock, and represents the date at which the clock has been reset the last time (written $\mathbf{d_r}$).

Definition 9 (Timed Valuation of a Non-branching Process). *Let β be a branching process, and β' a non-branching process of β. A* timed valuation *of β' is a mapping $\mathbf{d}$ from T' to $\mathbb{R}_{\geq 0}$, a mapping $\mathbf{d_b}$ from P' to $\mathbb{R}_{\geq 0}$, a mapping $\mathbf{d_e}$ from P' to $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ and a mapping $\mathbf{d_r}$ from $P' \cap \lambda^{-1}(X)$ to $\mathbb{R}_{\geq 0}$.*

We want to characterize the timed valuations of a non-branching process corresponding to a real timed execution of the NTA. In order to obtain such a characterization, we introduce some additional notation. Let t be an event, $\mathcal{C}^+(t)$ (resp. $\mathcal{C}^-(t)$) is the cut corresponding to configuration $[t]$ (resp. $[t]\setminus\{t\}$). We denote by $L(t) = \mathcal{C}^-(t) \cap \lambda^{-1}(L)$. Given a clock x, there is a unique place p_x^+ (resp. p_x^-) in cut $\mathcal{C}^+(t)$ (resp. $\mathcal{C}^-(t)$) whose label is x. Given a timed valuation of a non-branching process including t, we note $v(t)_x = \mathbf{d}(t) - \mathbf{d_r}(p_x^-)$ and $v'(t)_x = \mathbf{d}(t) - \mathbf{d_r}(p_x^+)$.

Definition 10 (Feasibility of a Timed Valuation). *Let β be a branching process of an NTA, and β' a non-branching process of β. A* timed valuation $(\mathbf{d}, \mathbf{d_b}, \mathbf{d_e}, \mathbf{d_r})$ *of β' is* feasible *iff it satisfies the following (in)equations: for every $t \in T'$,*

Causal (in)equations:	*Timed (in)equations:*
- $\forall p \in t^\bullet$, $\mathbf{d_b}(p) = \mathbf{d}(t)$	- $g(\lambda(t))[\{x \leftarrow v(t)_x\}_{x \in X}]$
- $\forall p \in {}^\bullet t$, $\mathbf{d_e}(p) = \mathbf{d}(t)$	- $\bigwedge_{\ell \in L(t)} \mathit{Inv}(\ell)[\{x \leftarrow v(t)_x\}_{x \in X}]$
- $\forall p \in {}^\circ t$, $\mathbf{d_b}(p) \leq \mathbf{d}(t) \leq \mathbf{d_e}(p)$	- $\bigwedge_{x \in R(\lambda(t))} v'(t)_x = 0$
- $\forall p \in P'$, $\mathbf{d_b}(p) \leq \mathbf{d_e}(p)$	- $\bigwedge_{x \in \mathit{Redefined}(\lambda(t))} v'(t)_x = v(t)_x$
- $\forall p \in \mathit{Min}(P')$, $\mathbf{d_b}(p) = \mathbf{d_r}(p) = 0$	

Definition 11. *Let $\mathcal{A}$ be an NTA and σ a timed sequence of $\mathcal{A}$. Its* timed non-branching process $(\beta, \mathbf{d}, \mathbf{d_b}, \mathbf{d_e}, \mathbf{d_r})$ *is inductively defined as follows:*

- *If σ is the empty sequence then β is $\mathit{Min}(P)$, $\forall p \in \mathit{Min}(P)$, $\mathbf{d_b}(p) = 0$, $\mathbf{d_e}(p) = \infty$, and for every $p \in \mathit{Min}(P) \cap \lambda^{-1}(X)$, $\mathbf{d_r}(p) = 0$.*
- *If $\sigma = \sigma'(\bar{e}, d)$ (d represents the date of the occurrence of $\bar{e}$) and $(\beta', \mathbf{d}', \mathbf{d_b}', \mathbf{d_e}', \mathbf{d_r}')$ is the timed non-branching process of σ' then, denoting $\mathcal{C}$ the cut associated with β', there is a unique possible extension of β' from $\mathcal{C}$ by an event t labeled by $\bar{e}$. β is this extension.*
  - *The timed valuation on places and transitions of β' is preserved except for the places $p \in {}^\bullet t$, for which we set $\mathbf{d_e}(p) = d$.*
  - *We set $\mathbf{d}(t) = d$, and for every place $p \in t^\bullet$, we set $\mathbf{d_b}(p) = d$ and $\mathbf{d_e}(p) = \infty$.*
  - *If $p \in t^\bullet$ is s.t. $\lambda(p) = x \in X$, if x is reset by e, we set $\mathbf{d_r}(p) = d$; otherwise let p' be the unique place of $\mathcal{C}$ whose label is x, then $\mathbf{d_r}(p) = \mathbf{d_r}(p')$.*

The next proposition shows the close relation between timed sequences and feasible timed non-branching processes, *i.e.* admitting a feasible timed valuation.

Proposition 1 (Feasibility is Equivalent to Execution). *Let $\mathcal{A}$ be an NTA. Then:*

1. *If σ is a timed sequence of $\mathcal{A}$ then its timed non-branching process is feasible.*
2. *If β is a non-branching process of $\mathcal{A}$ and $(\mathbf{d}, \mathbf{d_b}, \mathbf{d_e}, \mathbf{d_r})$ a feasible time valuation of β, then there is a timed sequence σ of $\mathcal{A}$ whose timed non-branching process is $(\beta, \mathbf{d}, \mathbf{d_b}, \mathbf{d_e}, \mathbf{d_r})$.*

We obtain as a corollary that the set of configurations obtained after firing a shuffle of concurrent transitions is a zone, a result also proved in [3] by other means.

The proof of this proposition (see [8]) heavily relies on the way invariants are handled: since transitions are connected by read arcs or pre arcs to a single condition per clock involved in some invariant, two concurrent transitions *must share these conditions and be connected to them by a read arc*. Thus, given an event t of the non-branching process β of σ, the satisfaction of the invariant constraint by t in σ is equivalent to the satisfaction of the invariant equation in $[t]$. If an event t is not firable in $[t]$ (its non-branching process β is not feasible) then it is firable in no extension of β. We illustrate this point in Example 1. Every event is connected to one place labeled by x by a read arc. Since the firing of a_2 redefines the invariant on clock x, there are two places labeled by x. This leads to two different occurrences of a_1 and a_3, depending on their ordering with a_2, which are necessary since they yield different behaviors. Firing a_3 before a_2 is unfeasible (see the dashed event), whereas a_3 is firable after a_2 with the constraint $x = y \wedge x \geq 3$. For a_1, we get similarly different timing constraints over clocks x and y.

Remark. It is worth noticing that we could increase slightly the locality of events by restricting connections to invariants clocks. Indeed, given a global edge $\bar{e}$, we could perform an offline untimed analysis of the system to restrict the possible set of undetermined locations, thus restricting the set of invariants to consider. That way to proceed would be similar to the method of *active* clocks [11].

Symbolic representation of timed executions. If we interpret the dates of a non-branching process β as variables and the (in)equations of Definition 10 as a system of linear inequations, we obtain a zone, denoted *Eq*(β). As stated by Proposition 1, this zone characterizes the set of timed sequences of β and β admits a timed sequence iff *Eq*(β) is satisfiable. The set of variables of *Eq*(β) is $\{\mathbf{d}(t) \mid t \in T\} \cup \{\mathbf{d_b}(p), \mathbf{d_e}(p) \mid p \in P\} \cup \{\mathbf{d_r}(p) \mid p \in P \cap \lambda^{-1}(X)\}$, whose size is larger than that of β. Since the complexity of operations on zones heavily depends on the number of variables, we will reduce the number of variables as much as possible. We thus keep only variables which are necessary to decide whether one can extend the non-branching process. To this aim, we state the following proposition, which is a key ingredient to compute incrementally timed feasibility of non-branching processes, and whose proof follows by examining the inequations of Definition 10.

Proposition 2. *Let β, β^+ be non-branching processes of some NTA such that $\beta \sqsubseteq \beta^+$, let $\mathcal{C}$ be the cut associated with β. We partition the variables of Eq(β^+) into three sets: $V_{\mathcal{C}}$ the variables associated with places of $\mathcal{C}$, V^- the variables of Eq(β) different* *from $V_{\mathcal{C}}$ and V^+ the remaining variables. Then Eq(β^+) can be decomposed as the conjunction Eq$(\beta) \wedge$ Eq$'(\beta^+ \setminus \beta)$, where the set of variables of Eq(β^+) (resp. Eq(β) and Eq$'(\beta^+ \setminus \beta)$) is the disjoint union $V^- \cup V_{\mathcal{C}} \cup V^+$ (resp. $V^- \cup V_{\mathcal{C}}$ and $V_{\mathcal{C}} \cup V^+$).*

Given a non-branching process β, we now define the zone Z_β as the zone $\exists V^-.Eq(\beta)$, with the notation of Proposition 2. If t is an event, Z_t denotes Z_{β_t}. By previous proposition, the set of variables of Z_t is equal to $V_{\mathcal{C}}$. We have $V_{\mathcal{C}} = \{\mathbf{d_b}(p), \mathbf{d_e}(p) \mid p \in \mathcal{C}\} \cup \{\mathbf{d_r}(p) \mid p \in \mathcal{C} \cap \lambda^{-1}(X)\}$, where $\mathcal{C}$ denotes the cut *Cut*$([t])$ (note that variable $\mathbf{d}(t)$ has been eliminated). It is worth noticing that the size[5] of $V_{\mathcal{C}}$ is equal to $2n + 3|X|$.

Timed unfolding. We can now propose a (semi-)algorithm, namely Algorithm 2, which builds the (possibly infinite) timed unfolding of an NTA such that an event occurs in the unfolding iff there is at least one timed sequence whose branching process includes this event. This algorithm is an extension of Algorithm 1, in which we associate with each event t of the unfolding the zone Z_t defined above. By previous study, we thus add the event t if and only if Z_t admits a solution (line 6). If Z is a zone, we write $\langle Z \rangle$ for the set of valuations satisfying Z. We also need to record the possible extensions already considered but leading to empty zones (line 7). The remaining point is the computation of the zone Z_t (line 5).

Algorithm 2 Building the (eventually infinite) timed unfolding (semi-algorithm)

Require: An NTA $\mathcal{A}$.
Ensure: The timed unfolding *T-Unf*$(\mathcal{A})$ of $\mathcal{A}$.
1: *T-Unf* $:= \{(\ell_{1,0}, \emptyset), \ldots, (\ell_{n,0}, \emptyset)\} \cup \{(x, \emptyset) \mid x \in X\}$;
2: $pe := PE($*T-Unf*$)$;
3: **while** $pe \neq \emptyset$ **do**
4: Choose an event $t = (\bar{e}, X, Y)$ in pe.
5: Compute the zone Z_t associated with the firing of t
6: **if** $\langle Z_t \rangle \neq \emptyset$ **then** $Extend($*T-Unf*$, t)$; $pe := PE($*T-Unf*$)$;
7: **else** Mark t as useless event. **end if** (In order to not consider t again)
8: **end while**

Since we do not keep the entire equation system of the non-branching process yielding an event t but only a projection of it, the computation of a new zone Z_t is a difficult task. To solve this problem, we compute additional zones associated with intermediate non branching processes. A first remark is that given the zone Z_β corresponding to some non-branching process β, and an extension β^+ of β consisting of a set of concurrent events, it is easy to compute the zone Z_{β^+}, simply by applying Definition 10 (see [8]).

Let T be the set of maximal events of configuration $C = [t] \setminus \{t\}$ and β_T be the non branching process associated with C. Using previous remark, it is easy, given the zone Z_T corresponding to β_T, to compute the zone Z_t. Our goal is thus to compute Z_T. Let $t_0 \in C$. A topological sort of $C \setminus [t_0]$ w.r.t. $\prec$ gives sets of concurrent events, which we call "slices". If we can apply the previous remark from β_{t_0} to these successive

[5] We obtain the bound claimed in the introduction.

slices, then we can compute iteratively, for each of these slices, the zone resulting from the firing of a slice, and thus get the desired zone. To apply the remark, the different intermediate non-branching processes have to extend each other. Because of read arcs, given a non-branching process β and an event $t' \in \beta$, this may be the case that β does not extend $\beta_{t'}$. This happens exactly when $[t']_\beta \setminus [t'] \neq \emptyset$. In this case, a transition t'' of this difference set reads a place belonging to $\beta' \setminus \mathit{Cut}(\beta')$. Using this characterization, we can compute correctly the initial event t_0. The previous discussion is formalized in [8], providing an algorithm for the computation of the zone Z_t.

As a direct consequence of the previous developments, we obtain the following theorem, which states properties of our (infinite) timed unfolding.

Theorem 1. *Algorithm 2 is correct: if $\mathcal{A}$ is an* NTA*, an event t occurs in the timed unfolding T-Unf$(\mathcal{A})$ iff there is at least one timed sequence whose non-branching process is β_t, and Z_t is the set of possible values for the variables associated with Cut$([t])$ obtained by timed sequences whose non-branching process is β_t.*

4 Algorithm for the Construction of a Finite Prefix

The construction of a complete finite prefix for read arcs Petri nets is much more involved that in classical Petri nets. It has been first studied in [21] where the problem is solved for a subclass of read arcs Petri nets, and a solution for the general class has then been proposed in [22]. All the algorithms rely on the detection of *cut-off events*: the cut obtained from every non-branching process including a cut-off event can be obtained by a non-branching process built from another already computed event.

In the timed framework, we must take into account the zones associated with the cut-off event and the previously computed event for checking whether the current cut-off event is redundant also w.r.t. timing constraints. In the context of TA, it is well-known that there are infinitely many incomparable zones. Thus, an *extrapolation* operator has been designed, which bounds the number of zones which can be computed. This extrapolation is an over-approximation, but is correct for checking reachability properties [6].

However, to compare the configurations reached by two non-branching processes $[t]$ and $[t']$, we cannot use directly the zones Z_t and $Z_{t'}$ computed in the previous section: indeed, the (unbounded) dates of occurrence of t and t' are irrelevant w.r.t. to the corresponding configurations reached in the NTA. Thus, we compute from zone Z_t a new zone corresponding to the possible valuations of the clocks reached in the NTA after firing all possible timed sequences corresponding to the non-branching process of $[t]$. To enforce termination, we then apply the classical extrapolation operator on this last zone and get the so-called *clock zone Test*$_t$. Unfortunately, two events whose clock zones and cuts are identical can lead to different processes: indeed, it must be noticed that a configuration $[t]$ may be extended by an event t' whose timed occurrence precedes the one of t! This may occur if the new event added t' is concurrent with t. Then, the date of t' may be smaller than that of t, which implies that classical extrapolation may induce mistakes, and thus that we can no more “forget the past” by comparing only clock zones and cuts. We will thus use a subclass of *synchronized events*, which have the desired property of “forgettable past”. Indeed, when an event t synchronizes all the

TA of an NTA $\mathcal{A}$, the timing occurrences of all events extending configuration $[t]$ will follow the one of t. This is the key ingredient which enables us to obtain a finite prefix, see Lemma 2. Note that this observation is quite similar to the one of [16] (operator \$). Note also that our algorithm avoids using the sophisticated algorithm of [22]. We now define an *unavoidable* subset of edges of an NTA.

Definition 12. *Let $\mathcal{A} = ((\mathcal{A}_i)_{1\leq i\leq n}, f)$ be an NTA and E' be a subset of global edges of $\mathcal{A}$ (i.e. a subset of Sync), then E' is* unavoidable *iff for every i, every circuit of the underlying graph of $\mathcal{A}_i$ intersects E': there is some e_i belonging to the circuit such that if e_i occurs in $\bar{e} \in$ Sync then $\bar{e} \in E'$.*

Obviously, any NTA has at least one *unavoidable* subset of edges. However the efficiency of the method will depend on two characteristics of the selected subset: its size and the *synchronization factor* of its edges (*i.e.* $|I(\bar{e})|$). Now we transform the NTA in such a way that when one fires an edge of E', one synchronizes the whole NTA.

Definition 13. *Let $\mathcal{A} = ((\mathcal{A}_i)_{1\leq i\leq n}, f)$ and E' be an unavoidable set of edges, then*

- *if $\bar{e} \in E'$, its synchronized version is $Sync(\bar{e}) = \{\bar{e'} \mid \forall i \in I(\bar{e}),\ e'_i = e_i$ and $\forall i \notin I(\bar{e}),\ \exists \ell_i \in L_i$ s.t. $e'_i = idle(\ell_i)\}$ with $idle(\ell_i) = \ell_i \xrightarrow{true,\varepsilon,\emptyset} \ell_i$.*
- *$\mathcal{A}(E')$ is the NTA where E' has been replaced by $\bigcup_{\bar{e}\in E'} Sync(\bar{e})$.*

Note that $\mathcal{A}(E')$ is not defined *via* a synchronization function but directly with its set of edges. However all previous results equally apply on such NTA. Note also that $\mathcal{A}$ and $\mathcal{A}(E')$ have the same set of (finite or infinite) timed sequences with the same intermediate configurations and so any property expressible in terms of these extended timed sequences is equivalent for $\mathcal{A}$ and $\mathcal{A}(E')$. This is in particular the case for reachability, and event occurrence which are the usual properties checked by the unfolding method. Note that if for all $\bar{e} \in E'$, $I(\bar{e}) = \{1, \ldots, n\}$ then $\mathcal{A}(E') = \mathcal{A}$.

Let us now explain how we build the finite prefix of the timed unfolding of $\mathcal{A}(E')$ (Algorithm 3, page 13). When we fire a synchronized event t, we build the clock zone $Test_t$ as follows. We project the last zone (corresponding to Z_t of the previous section before elimination of variable $\mathbf{d}(t)$) over the variables $\mathbf{d}(t)$ and $\{\mathbf{d_r}(p) \mid p \in Cut([t]) \cap \lambda^{-1}(X)\}$. Then we relativise the result w.r.t. variable $\mathbf{d}(t)$, *i.e.* we replace variables $\mathbf{d_r}(p)$ by $\mathbf{d}(t) - \mathbf{d_r}(p)$, and we eliminate variable $\mathbf{d}(t)$. We note W_t this new zone.

Lemma 1. *The zone W_t corresponds to the set of valuations v such that there exists a timed sequence whose non-branching process β_t, and such that in $\mathcal{A}(E')$, the clock valuation after having fired the above timed sequence is v.*

We close zone W_t by time elapsing and intersect it with the invariant specified by $Cut([t])$, *i.e.* the conjunction of invariants of locations appearing in $Cut([t])$. At last we extrapolate the result, yielding the zone $Test_t$. We then check whether there exists a synchronized event $t' \lhd t$ [6] with $\lambda(Cut([t'])) = \lambda(Cut([t]))$ and $\langle Test_t\rangle \subseteq \langle Test_{t'}\rangle$. If this is the case, we mark t as useless and we do not produce its output places.

[6] $\lhd$ denotes an *adequate order*, as required by [13, 17] for proving completeness of the finite prefix construction. A possible such order is $Card([t']) < Card([t])$.

It is worth noticing that diagonal constraints appearing in zones Z_t do not induce wrong extrapolation results as in timed automata using diagonal constraints [6]. Indeed, the zones Test_t are related to the NTA $\mathcal{A}(E')$, which does not have diagonal constraints, the extrapolation operator can thus safely be used.

Algorithm 3 Building a finite and complete prefix of the timed unfolding

Require: An NTA $\mathcal{A}$.
Ensure: A finite and complete prefix *Fin* of *T-Unf*($\mathcal{A}$).

$\mathit{Fin} := \{(\ell_{1,0}, \emptyset), \ldots, (\ell_{n,0}, \emptyset)\} \cup \{(x, \emptyset) \mid x \in X\}$; $pe := PE(\mathit{Fin})$;
while $pe \neq \emptyset$ **do**
  Choose an event $t = (\bar{e}, Y_{in}, Y_r)$ in pe.
  if t is not a synchronized event **then**
    Compute the zone Z_t associated with the firing of t
    if $\langle Z_t \rangle \neq \emptyset$ **then** $\mathit{Extend}(\mathit{Fin}, t)$; $pe := PE(\mathit{Fin})$;
      else Mark t as useless event; **end if**   (In order to not consider t again)
  else   (t is a synchronized event)
    Compute the extrapolated zone Test_t of clock values.
    if $\exists$ a synchronized event $t' \lhd t \mid \lambda(\mathit{Cut}([t'])) = \lambda(\mathit{Cut}([t])) \wedge \langle \mathit{Test}_t \rangle \subseteq \langle \mathit{Test}_{t'} \rangle$ **then**
      Mark t as useless event.   (In order to not consider t again)
    else if $\langle Z_t \rangle \neq \emptyset$ **then** $\mathit{Extend}(\mathit{Fin}, t)$; $pe := PE(\mathit{Fin})$;
      else Mark t as useless event; **end if**   (In order to not consider t again)
  end if
end while

Synchronized events enjoy the following nice property, proved in [8].

Lemma 2 (Forgettable Past of Synchronized Events). *Let t be a synchronized event of a branching process of an NTA. It is equivalent to extend β_t and to build a non-branching process from Cut$([t])$ with constraints on variables $\{\mathbf{d_r}(p) \mid p \in \mathit{Cut}([t]) \cap \lambda^{-1}(X)\}$ given by Test_t.*

Finally the following theorem states the termination and soundness of Algorithm 3.

Theorem 2. *Algorithm 3 terminates and the computed finite prefix Fin is such that: (1) a transition t can become firable in $\mathcal{A}(E')$ iff an event labeled by t occurs in Fin; (2) a configuration is reachable in $\mathcal{A}(E')$ iff an equivalent configuration (w.r.t. strong time bisimulation) is reachable by a timed sequence whose non-branching process is included in Fin.*

We have thus constructed for any NTA $\mathcal{A}$ a finite prefix which is complete for checking reachability properties, and transition enabling.

5 Related Work

Partial order method for TA with ample sets. During the state exploration, partial-order methods select a subset of transitions rather than developping all the state successors. This subset, called an *ample* set, fulfills some properties relying on an independence relation between transitions (see [19] for more details). Thus the efficiency of these methods is closely related to the size of the independence relation. So introducing time (and its implicit synchronizations) will necessarily restrict the corresponding relation for the associated untimed model. In [4, 18], the authors define an alternative semantics for NTA based on local time elapsing. Despite the fact that this semantics allows more behaviours than the standard semantics, the reachability relation associated with the usual semantics can be checked on the system corresponding to the new one. Moreover, the independence relation is enlarged when considering local time elapsing. Clearly, the efficiency of this method depends on two opposite factors: local time semantics generate more states but the independence relation restricts the exploration.

Partial order method for TA with Mazurkiewicz trace. In [16], the independence between transitions of a TA are exploited in a different way: the occurrences of two independent transitions do no need to be ordered (and consequently nor the occurrences of the clock resets). Thus a symbolic state in this framework is defined by a location and constraints between variables related to both the clock resets and the transition occurrences. When two sequences ab and ba are developped from a state with a and b independent, they will lead to the same symbolic state whereas with the ordinary construction they would generally yield two different states. However this method does not exploit the independence relation for limiting the exploration.

Partial order method for time Petri nets with ample (or stubborn) sets. In Petri nets, ample sets are denoted as stubborn sets [20]. Stubborn sets are similar to ample sets but their definition takes advantage of the "locality" of the firing rule. In [23], the authors generalise this concept to time Petri nets (TPN) calling it *a ready set* and applying it to the class graph construction of [5] where a class is similar to a symbolic state of a TA. Given a symbolic state, a ready set is a stubborn set with an additional constraint relative to the timing occurrences of enabled transitions. Thus the efficiency of the method depends on the weakness of the timing coupling between transitions.

Partial order method for TPNs with unfoldings. Depending on the Petri net to be analysed, the unfolding and stubborn set methods behave very differently. For instance, the former one outperforms the latter one when the net presents "confusion", (*i.e.* when the firing of a transition may influence the conflict set of another unrelated transition of the net). The generalisation of the unfoldings for TPNs has been developed by different searchers. First, in [2] the authors have studied the realisability of a non-branching process in a TPN showing that the temporal mechanism of these nets requires a global analysis of the process in order to check the firing of a transition in such a process. Starting from this analysis, [10] has recently designed a finite complete prefix for TPNs. In another direction, [15] proposes a method controlling the class graph construction with an unfolding of the untimed net. However this unfolding may be infinite whereas the TPN is bounded. In [14] the authors propose a discrete-time semantics for TPNs equivalent to the dense-time one w.r.t. reachability. The net include a special transition of the net modelling time elapsing but the occurrence of this transition in the unfolding requires a complete cut drastically decreasing the locality of the unfolding. Furthermore, this method suffers the combinatorial explosion related to the discrete time approach.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.
2. T. Aura and J. Lilius. A causal semantics for time Petri nets. *Theor. Comp. Sci.*, 243(1–2):409–447, 2000.
3. R. Ben Salah, M. Bozga, and O. Maler. On interleaving in timed automata. In *17th Int. Conf. Concur. Theory (CONCUR'06)*, LNCS 4137. Springer, 2006. To appear.
4. J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial order reductions for timed systems. In *9th Int. Conf. Concur. Theory (CONCUR'98)*, LNCS 1466, 485–500. Springer, 1998.
5. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Softw. Engineering*, 17(3):259–273, 1991.
6. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in Syst. Design*, 24(3):281–320, 2004.
7. P. Bouyer, S. Haddad, and P.-A. Reynier. Timed Petri nets and timed automata: On the discriminating power of Zeno sequences. In *33rd Int. Coll. Automata, Languages and Programming (ICALP'06)*, LNCS 4052, 420–431. Springer, 2006.
8. P. Bouyer, S. Haddad, and P.-A. Reynier. Timed unfoldings for networks of timed automata. Research Rep. LSV-06-09, Lab. Spécification et Vérification, ENS de Cachan, France, 2006.
9. Th. Chatain and C. Jard. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In *3rd Int. Conf. Formal Modeling and Analysis of Timed Syst. (FORMATS'05)*, LNCS 3829, 196–210. Springer, 2005.
10. Th. Chatain and C. Jard. Complete finite prefixes of symbolic unfoldings of time Petri nets. In *27th Int. Conf. Appl. and Theory of Petri Nets (ICATPN'06)*, LNCS 4024, 125–145. Springer, 2006.
11. C. Daws and S. Tripakis. Model-checking of real-time reachability properties using abstractions. In *4th Int. Conf. Tools and Algo. for the Construction and Analysis of Syst. (TACAS'98)*, LNCS 1384, 313–329. Springer, 1998.
12. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *of the Work. Automatic Verification Methods for Finite State Systems (1989)*, LNCS 407, 197–212. Springer, 1990.
13. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in Syst. Design*, 20(3):285–310, 2002.
14. H. Fleischhack and C. Stehno. Computing a finite prefix of a time Petri net. In *23rd Int. Conf. Appl. and Theory of Petri Nets (ICATPN'02)*, LNCS 2369, 163–181. Springer, 2002.
15. J. Lilius. Efficient state space search for time Petri nets. ENTCS 18, 1998.
16. D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. In *10th Int. Conf. Tools and Algo. for the Construction and Analysis of Syst. (TACAS'04)*, LNCS 2988, 296–311. Springer, 2004.
17. K. McMillan. A technique of state space search based on unfolding. *Formal Methods in Syst. Design*, 6(1):45–65, 1995.
18. M. Minea. Partial order reduction for model checking of timed automata. In *10th Int. Conf. Concur. Theory (CONCUR'99)*, LNCS 1664, 431–446. Springer, 1999.
19. D. Peled. All from one, one for all: on model checking using representatives. In *5th Int. Conf. Computer Aided Verif. (CAV'93)*, LNCS 697, 409–423. Springer, 1993.
20. A. Valmari. Stubborn sets for reduced state space generation. In *10th Int. Conf. Appl. and Theory of Petri Nets (ICATPN'89)*, LNCS 483, 491–515. Springer, 1989.
21. W. Vogler, A. L. Semenov, and A. Yakovlev. Unfolding and finite prefix for nets with read arcs. In *9th Int. Conf. Concur. Theory (CONCUR'98)*, LNCS 1466, 501–516. Springer, 1998.
22. J. Winkowski. Reachability in contextual nets. *Fundam. Inform.*, 51(1-2):235–250, 2002.
23. T. Yoneda and B.-H. Schlingloff. Efficient verification of parallel real-time systems. *Formal Methods in Syst. Design*, 11(2):187–215, 1997.