NTAs and their Symbolic Unfolding

Summer Internship Research Report

July 5, 2007

by

Aaditya Ramdas (aaditya.ramdas@gmail.com)
2nd Year Undergraduate Student
B.Tech. Computer Science & Engineering
Indian Institute of Technology, Bombay

under the guidance of

Prof. Emmanuel Fleury (fleury@labri.fr)
Prof. Marc Zeitoun (mz@labri.fr)

at

LaBRI, University of Bordeaux I

Contents

1 Introduction

There have been several models that have been used to represent real-time systems. Out of these, timed automata, with now standardized semantics, has become an accepted model, and several fairly advanced results have already been derived within this framework.

One of the common problems with timed automata is that of reachability. For the most general timed automata, an infinite number of reachable states exist. Hence, reachability analysis is extremely difficult in practical situations.

For example, if we want to see if a system ever reaches an error state, we can never be sure until the reachability algorithm has explored "all" the states. This kind of question is especially important in verifying systems and seeing whether their defined behaviour (which can be often very complicated) is indeed perfect and there is no way for an error to exist.

There might be cases where we are willing to compromise a little bit on exact correctness so as to achieve a terminating reachability algorithm. If we use "unfolding" as our reachability checking method (can be naively thought of as a breadth first search across all possible transitions), we would be interested to know if I can stop the unfolding at some stage, because every state that will be reached from that point has already been reached in the past.

It turns out that in general cases it is not possible, but by applying the common extrapolation method, we can indeed generate what is called a finite prefix of the unfolding. As we would like, this prefix is also complete - obviously not exactly complete, as that would have to be infinite, but complete at least upto extrapolation.

This idea has been extensively pursued recently in <fill in>. I have been working on correctly developing the initial ideas given in the paper and the thesis, as well as providing intuitions, examples and concise proofs to support the definitions, algorithms and theorems. The organization of this report/notes can be seen in the table of contents.

Happy reading! :)

2 Formal Syntax

This section is of 2 parts. One will introduce general definitions relating to networks of timed automata and its semantics. The next will introduce more specific definitions to the topic of symbolic unfoldings of these networks. Intuitions and examples will be given wherever possible.

2.1 Basics

Here, we shall look to define the basic definitions (standard) regarding Networks of Timed Automata, its semantics, as well as Read-Arc Petri Nets and some associated relations on them.

2.1.1 Definition 1 (Timed Automaton (TA))

A timed automaton A over Σ is a tuple $(L, l_0, X, \Sigma, E, Inv)$ where :

- L is a set of locations
- $l_0 \in L$ is the initial location
- X is a finite set of clocks
- Σ is a finite set of actions that cause state transitions
- $E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is a finite set of edges
- $Inv \subseteq C(X)^L$ is the set of invariants for each location □

Clock Constraints
A clock constraint is a conjunctive formula of atomic constraints of the form $x \sim n$ or $x - y \sim n$ for $x, y \in X, \sim \in \{\leq, <, =, >, \geq\}, n \in \mathcal{Z}$. These will be used as guards (look below).
$C(X)$ will denote the set of clock constraints, $C_{ub}(X)$ is the set of upper bounded clock constraints (ie $\sim \in <, \leq$), and $C_{df}(X)$ is the set of diagonal free clock constraints (ie only of the form $x \sim n$, also called rectangular constraints). Invariants are generally subsets of $C_{ub}(X)$ but this is not necessary.

Edges
As defined, $e \in E$ will be of the form (l, g, a, R, l') which we represent as $l \xrightarrow{g,a,R} l'$. This means that there is an edge from location l to location l' if the action a occurs at a time when the guard g is satisfied and this transition causes a reset of all the clocks in $R \subseteq X$.

2.1.2 Definition 2 (Network of TA (NTA))

A network of timed automata is a finite family $(A_i)_{1\leq i\leq n}$ of n TA (with pairwise disjoint locations) along with an n-ary function $f : (\Sigma \cup \{\perp\})^n \to \Sigma$. $\square$

Synchronization Function
f is called a synchronisation function as it gives all the synchronisations that exist in the network. Intuitively, each untimed transition that the network can make is given by an action, which is an n-tuple describing what action to perform in each of the n automata.

Other Notations
The clocks are assumed global and not disjoint. Hence the entire network reads the same set of clocks $X = \bigcup_{1\leq i\leq n} X_i$.
It is assumed that $A_i = (L_i, l_{i,0}, X_i, \Sigma_i, E_i, Inv_i)$ represents the i-th automaton and that we extend Inv over $L = \bigcup_{1\leq i\leq n} L_i$.

2.1.3 Definition 3 (Semantics of NTA))

First we define the following symbols which we will use frequently:

- $\bar{\Sigma}_\perp, \bar{E}$ are respectively $(\Sigma \cup \{\perp\})^n, \Pi_{1\leq i\leq n}(E_i \cup \{\perp\})\backslash\{\perp^n\}$. Their elements are denoted by $\bar{a}$ and $\bar{e}$ respectively.
- $Label : \bar{E} \longrightarrow \bar{\Sigma}_\perp$ is a labelling function that maps $\bar{e}$ to $\bar{a}$ as $a_i = b_i$ if $e_i = l_i \overset{g_i,b_i,R_i}{\longrightarrow} l_i'$ and $a_i = \perp$ if $e_i = \perp$.
- $Sync = Label^{-1}(f^{-1}(\Sigma)) \subseteq \bar{E}$ is the set of transitions of the NTA (the set of synchronized edges).
  In other words, $Sync = \{\bar{t} \in \bar{T} \mid \exists a \in \Sigma \text{ such that } f(a_1, \ldots, a_n) = a\}$
- $I(\bar{e}) = \{1 \leq i \leq n \mid e_i \neq \perp\}$. Then we have $g(\bar{e}) = \bigwedge_{i\in I(\bar{e})} g_i$ and $R(\bar{e}) = \bigcup_{i\in I(\bar{e})} R_i$.

Let $\mathcal{A} = ((\mathcal{A}_i)_{1\leq i\leq n}, f)$ be an NTA. Then, its semantics is the transition system $S_\mathcal{A} = (Q, q_0, \to)$ where

- $Q = \Pi_{1\leq i\leq n} L_i \times \mathbb{R}^X_{\geq 0}$
- $q_0 = (\bar{l_0}, 0)$
- $\to$ is defined by
  - $(\bar{l}, v) \xrightarrow{d} (\bar{l}, v + d)$ if $d \in \mathbb{R}_{\geq 0}$ and $\forall i, v + d \models Inv(l_i)$
  - $(\bar{l}, v) \xrightarrow{a} (\bar{l'}, v')$ if $\exists \bar{e} \in Label^{-1}(f^{-1}(a))$ s.t.
    - $v \models g(\bar{e})$

* $v' = v[R(\bar{e}) \leftarrow 0]$
* $l'_i = l_i$ if $e_i = \perp$ and it is given by e_i otherwise

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

Also, given an element $\bar{e} = (e_i)_{1 \leq i \leq n} \in Sync$, we define:

- $X_{inv} = \{x \in X \mid \exists i, l_i \in L_i, x \in Clocks(Inv_i(l_i))\}$
- $ModifyInv(\bar{e}) = \{x \in X \mid [\![\bigwedge_{i \in I(\bar{e})} Inv_i(l_i)]\!]_{|x} \neq [\![\bigwedge_{i \in I(\bar{e})} Inv_i(l'_i)]\!]_{|x}\}$
- $Test(\bar{e}) = X_{inv} \cup Clocks(g(\bar{e}))$
- $Modified(\bar{e}) = R(\bar{e}) \cup ModifyInv(\bar{e})$
- $Pre(\bar{e}) = \bigcup_{i \in I(\bar{e})} l_i \cup Modified(\bar{e})$
- $Post(\bar{e}) = \bigcup_{i \in I(\bar{e})} l'_i \cup Modified(\bar{e})$
- $Read(\bar{e}) = Test(\bar{e}) \setminus Modify(\bar{e})$

2.1.4 Definition 4 (Read Arc Petri Nets (RAPN))

A read-arc Petri net is a tuple $\mathcal{N} = (P, T, Pre, Post, Read, M_0)$ where

- P is a finite set of places
- T is a finite set of transitions
- $Pre, Post, Read$ are three mappings from T to 2^P
- $M_0 \in 2^P$ is the initial marking □

$Pre(t), Post(t), Read(t)$, (respectively called the backward, forward and read incidence mappings) are respectively represented by ${}^\bullet t, t^\bullet$, ${}^\circ t$. Also, ${}^\bullet p, p^\bullet$ will represent the sets $\{t \in T \mid p \in t^\bullet\}, \{t \in T \mid p \in {}^\bullet t\}$ respectively.

A special type of net called the Occurrence Net is used to model the unfolding of an NTA, and it is given in Definition 5.

2.1.5 Definition 5 (Relations on RAPNs)

1. Let $\prec$ **(the strong precedence relation)** be the minimal transitive relation over $P \cup T$ satisfying for every $t, t' \in T, p \in P$,
   - if $p \in {}^\bullet t$ then $p \prec t$
   - if $t \in {}^\bullet p$ then $t \prec p$

- if $p \in {}^\circ t$ and $p \in {}^\bullet t'$ then $t \prec t'$
- if $p \in {}^\circ t$ and $p \in t'^\bullet$ then $t' \prec t$

Let $\preceq$ be its reflexive closure, ie, let $x \preceq x$ for all $x \in P \cup T$. □

The ***weak precedence relation*, $<$,** does not include the third point above. Similarly, let $\leq$ be its reflexive closure.

It is worth noticing that $p \in {}^\circ t$ does not imply $p \prec t$ or $p < t$.

In short, it can be captured by the Figure 1.

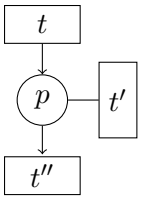

Figure 1: Precedence Relation

Here, the strong precedence asserts $t \prec t' \prec t''$ and $t \prec p \prec t''$ while the weak precedence only asserts $t < t'$ and $t < p < t''$. We shall deal only with the strong precedence operator and show why as well.

2. Let $\#$ **(the conflict relation)** be defined by $x \# y$ iff
   - $\exists p \in P, \exists t, t' \in p^\bullet$
   - $t \neq t'$
   - $t \leq x$
   - $t' \leq y$ □

Intuitively, we model executions of NTAs using single token Petri Nets (each place cannot have more than one token at any given point of time). This means that if there are 2 transitions t, t' which can be fired from the same place p, then obviously, in a single execution, only one of the 2 transitions can be fired. Hence there is a 'conflict' between the two transitions.

We extend this idea to say that all places and transitions that follow t (S_t) will be in conflict with those that follow t' ($S_{t'}$)(with 'follow' being represented by $t \leq x$).

In short, it can be captured by the Figure 2. Here, $t \# t'$ and it follows from this, that $\forall l, l' \in S_t, S_{t'}$ respectively we have $l \# l'$. Either the

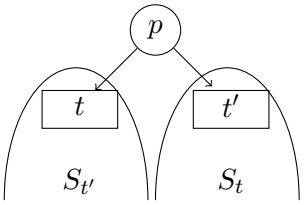

Figure 2: Conflict Relation

events inside S_t can occur or the events in $S_{t'}$ (in a single execution) but not both.

Let us make this notion a little clearer with 2 examples. This will also show us why we chose $<$ and not $\prec$ for the ordering for deciding conflict.

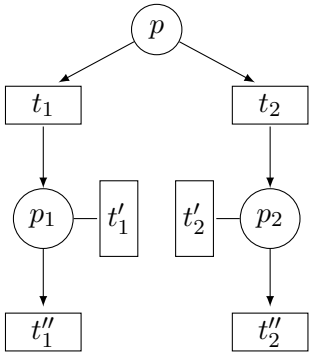

Figure 3: Example 1 : Conflict Relation

In Figure 3 (Example 1), by our intuition behind what *conflict* should mean, we want $t'_1 \# t_2$, $t'_1 \# t'_2$, $t'_1 \# t''_2$ and (symmetrically for t'_2) and so on for other nodes. This is what we want, and this is exactly what using $<$ will give us. Clearly, t_1 and t_2 are in conflict, and $t'_1 < t_1$, $t'_2 < t_2$, $t''_1 < t_1$, etc. which causes the mentioned conflicts. Now we shall see why not to use $\prec$.

In Figure 4, notice that $t'_1 \prec t''_1$ but $t'_1 \not< t''_1$. This is the main point to notice. If this represented an execution, we can definitely have t''_1 and t''_2 both occurring (say by firing t_1, t_2, t''_2, t''_1 in that order). We don't want the conflict relation to represent a conflict between t''_1 and t''_2, even though there is indeed a conflict between t'_1 and t_2. Hence, we discard $\prec$ for the conflict relation, and use $<$.

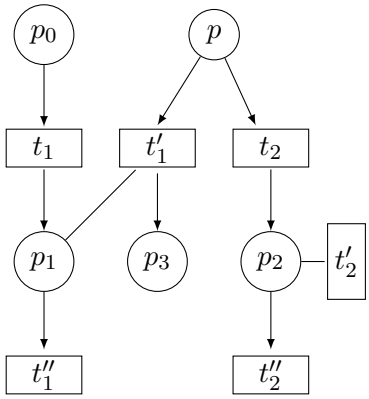

Figure 4: Example 2 : Conflict Relation

Finally, we move forward accepting no conflict between t''_1 and t''_2.

3. Let $||$ **(the concurrency relation)** be defined, using $\bowtie \in \{\leq, \preceq\}$, by $x \;||_{\bowtie}\; y$ or $x - co_{\bowtie} - y$ iff none of the following are satisfied:

   - $x \bowtie y$
   - $y \bowtie x$
   - $x \# y$

   This yields 2 different concurrency relations $||_{\leq}$ and $||_{\preceq}$. Unless otherwise stated, the co- relation will use $\bowtie = \preceq$. □

   Intuitively, this means that there is no relation between the occurrences of two concurrent events t and t'. They can occur in the same execution with t occuring before t' or vice versa, independent of each other.

   Otherwise stated, they are occurring in disconnected automata that aren't synchronized at that point (the 2 sets of automata involved are disjoint at this point).

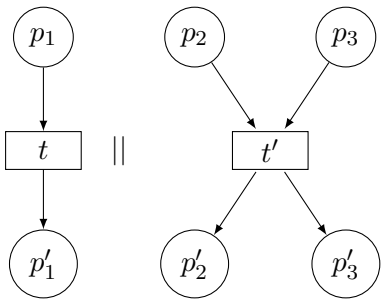

Figure 5: Concurrency Relation

This can be captured in Figure 3. Here, we see that in such a network, t and t' are concurrent events (even if p'_1, p'_2, p'_3 all get synchronized by some other event t'' at the next step).

2.2 Specifics

Here, we shall look to develop a few notions which will be specific to the reachability problem that we are trying to solve, and make it as useful as possible to correctly and conveniently model executions of NTAs.

For this purpose, a *branching process* shall be used to represent several possible executions of the NTA, whereas a single execution is to be given by a *non-branching process*. We shall define some common terms which shall be referred to throughout the paper, and also define what it means to extend a process. Based on this formalization, we shall go on to explore the related algorithms in the next section.

2.2.1 Definition 6 (Occurrence Net)

An occurrence net is an RAPN $\mathcal{N}$ such that :

- $\forall p \in P, |{}^\bullet p| \leq 1$
- $\mathcal{N}$ is acyclic ie $<$ is a well-founded partial order of $\mathcal{N}$.
- $<$ is finitely preceded, ie $\forall x \in P \cup T$, we have $\{y \in P \cup T \mid y < x\}$ is finite.
- $\forall x \in P \cup T$, the ordering $\prec$ on the $predecessors_<$ of x is a partial order.
- No element is in conflict with itself, ie $\forall x \in P \cup T, \neg(x \# x)$
- $M_0 = Min(P)$ where $Min(P)$ is the set $\{p \mid {}^\bullet p = \phi\}$ □

The first point means that each place/location can't have more than 1 event/transition feeding it. But clearly, an event can have several places feeding it (or it can synchronize several places to act together, as we will interpret it when dealing with executions of NTAs).

The second point ensures that something that is built makes 'causal sense'. We don't want a case where t should have necessarily occurred before t' and vice versa. Note that it is not necessary for it to be a total ordering, but a partial order will do (I don't really need an ordering between all pairs of elements)

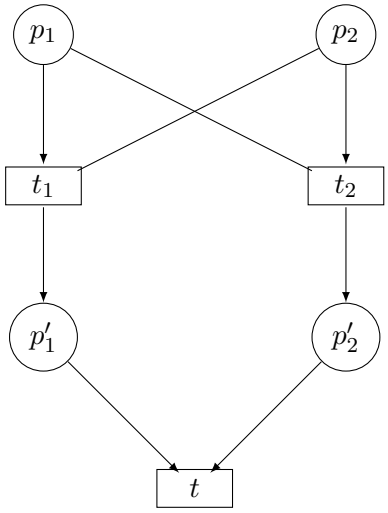

Figure 6: Incorrect ON, partial ordering of $\preceq$ on $predecessors_<$ disobeyed

explore precise reasons for 3rd point and CHECK on $<$ vs $\prec$

The fourth point ensures that there will not be any contextual conflict, as shown in Figure 6 (which shows an RAPN that cannot represent an execution). This is not captured in the second point, since $<$ does not capture any relation between $\{p_1, t_1\}$ and $\{p_2, t_2\}$. But applying $\preceq$ onto the $predecessors_<$ of t, disallows $\preceq$ from being a partial order.

The fifth point means, that in Figure 2, S_t and $S_{t'}$ don't *intersect*, ie they don't have common places or events. We require this because we want to use occurrence nets for modelling executions, and in such an execution, once you choose to go along either S_t or $S_{t'}$, we cannot get a place which is also a part of another conflicting execution.

The last point helps the base case to be the starting point of the unfolding of an NTA, as we will see soon with our modelling of executions of NTAs.

Lemma 2.1. *The above conditions for an ON automatically ensure the following important property : $\forall t \in T$, ${}^\bullet t$ is a co-set.*

Proof. Assume that this is not the case. Then, there are is a pair of places $p_1, p_2 \in {}^\bullet t$ such that the co-relation does not hold for them. This means that one of the following is disobeyed : $p_1 < p_2$ or $p_2 < p_1$ or $\neg(p_1 \# p_2)$. Let us take it case by case.

Since $p_1 < p_2$, consider the first transition t_{12} that is along the path from p_1 that will (eventually) lead to p_2. There is definitely such a transition, as

we know that $<$ also defines minimal causal past (let us abuse the definition slightly so that it includes conditions and transitions), we know that p_1 is definitely needed for p_2 to appear along any execution, and hence at least t_{12} must be fired to obtain p_2 (immediately or later). Hence, we immediately have $t_{12}\#t$ (since both are in $p^\bullet$) and $t_{12} < p_2$ (by choice of t_{12}), and $p_2 < t$ (since $p_2 \in {}^\bullet t$). Hence, we get $t_{12} < t$ (by transitivity of $<$). Plugging the 2 events t_{12}, t as t, t' and $x = y = t$ into the conflict conditions, we obtain $t\#t$, which is a contradiction to our assumption of an ON.

It is a symmetrical argument for the second case. If the third condition is violated, then we have $p_1\#p_2$. This means that there were 2 conditions $t_1, t_2 \in p^\bullet$ (for some p) such that $t_1 < p_1$, $t_2 < p_2$. Also, we have $p_2 < t, p_1 < t$ since $p_1, p_2 \in {}^\bullet t$. Hence we have $t_1 < t, t_2 < t$ (by transitivity of $<$). Plugging this trivially into the conflict conditions, we obtain $t\#t$, which leads to the same contradiction.

Hence our assumption was wrong, and $\forall t \in T$, ${}^\bullet t$ is indeed a co-set. □

2.2.2 Definition 7 (Branching Process (BP))

Let $\mathcal{A}$ be an NTA given as a family $((\mathcal{A}_\rangle)_{1\leq i\leq n}, f)$. A branching process, β of $\mathcal{A}$ is defined as a pair of an occurrence net $\mathcal{N} = (P, T, Pre, Post, Read, M_0)$ and a labeling function λ ranging over $P \cup T$ such that:

- $\lambda(P) \subseteq L \cup X$
- $\lambda(T) \subseteq Sync$
- λ is a bijection between M_0 and $\bigcup_{1\leq i\leq n} l_0 \ \cup \ X$.
- $\forall t \in T$, λ is a bijection between ${}^\bullet t, {}^\circ t, t^\bullet$ and $Pre(\lambda(t)), Read(\lambda(t)), Post(\lambda(t))$ respectively.
- $\forall t, t' \in T$, if $\lambda(t) = \lambda(t'), {}^\bullet t = {}^\bullet t', {}^\circ t = {}^\circ t'$, then we have $t = t'$.
- **CHECK - one more pt?** □

The first 2 points ensure that conditions correspond to locations or clocks of $\mathcal{A}$, while transitions/events correstpond to possible transitions $\mathcal{A}$.

The third point ensures that the initial marking consists of the initial locations of all the TAs in the NTA as well as all the clocks in the NTA.

The 4th point is trivial, while the last point ensures that there is no redundancy, so that the same event (which means that they have the same marking as well as the same input and read conditions) is not built twice.

2.2.3 Definition 8 (Non-branching Process (NBP))

Let $\beta = (N, \lambda)$ be a BP with events T and conditions P. We consider an occurrence net N' on $(P', T') \subseteq (P, T)$ with λ' as the restriction of λ on N'. Then, $\beta' = (N', \lambda')$ is called an NBP if it satisfies :

- $\forall t \in T, \forall p \in {}^\bullet t \cup {}^\circ t \cup t^\bullet, t \in T' \Rightarrow p \in P'$
- $\forall p \in P, \forall t \in {}^\bullet p, p \in P' \Rightarrow t \in T'$
- $\preceq$ restrained onto $P' \cup T'$ is a partial order.
- $\forall x, y \in P' \cup T', \neg(x \# y)$
- $Min(P') = Min(P)$ □

The first 2 points in the definition are present to ensure that the events and conditions are consistent in the NBP. It should not be the case that there are events without some of their preconditions or postconditions present, or that there is a place without a transition creating it (unless it is in $Min(P)$).

The third condition is because there should be no inconsistencies in causality (given by the causal relation $\preceq$). When we map an NBP to an execution, we should not arrive at a result that is contradictory, where 2 events a, b seem to have caused each other!

Also, as the 4th point mentions, we don't want any conflicts between places. We will soon model an execution of an NTA as an NBP, and in a single execution, we cannot have 2 conflicting processes occurring. The last point is trivial, and says that our base which we start from, must be the same for the BP and NBP.

2.2.4 Definition 9 (Terms on NBPs)

We fix a BP called β (with places P and transitions T) and an NBP of β called β' (with places P' and transitions T'). Let t be some event in β'. With respect to this we define :

1. **A Configuration C of β and $Conf(\beta')$**
   $C \subseteq T$ is a configuration iff it satisfies:
   - $\forall t' \in C,\ \forall t \in T,\ t \leq t' \Rightarrow t \in C$ (C is closed w.r.t. $\leq$).
   - $\preceq$ should be a partial order on C.
   - $\forall t, t' \in C,\ \neg(t \# t')$.

Also, $Conf(\beta')$ is used to represent the set of all events of β'. □

We note, importantly, that the events of NBP β' form a configuration and this is denoted by $[\beta']$. This holds the other way around as well, ie, given a configuration C, there exists a unique NBP β_C such that $C = [\beta_C]$. Hence we note that $Conf(\beta')$ is indeed a configuration.

2. **Cut of β' and $Cut(\beta')$**
   A cut of β' is a maximal-sized $co_{\preceq}$-set of β'.

   We now define $Cut(\beta')$ as the maximal-sized $maximal_{\preceq}$ $co_{\preceq}$-set of β'. Given a configuration C, it can be defined in terms of C as $Cut(C) = (Min(P) \cup C^\bullet) \backslash {}^\bullet C$. We shall call the $Cut(\beta')$ as $Cut([\beta'])$.

   Also, if c is a cut of β' then $Loc(c)$ is defined as the unique vector element of $\Pi_{1 \leq i \leq n} L_i$ obtained as an image of λ on $c \cap \lambda^{-1}(L)$ □

   Intuitively, the $Cut(C)$ is the maximal set of all maximal clocks and all those maximal places in an NTA which are concurrent, ie, the NBP can be extended from any one of them. This can be captured in the Lemma 2.2.

   It is important to note the difference between a cut of β' and the $Cut(\beta')$.

3. **Causal Pasts of t in β', β or $[t]_{\beta'}, [t]$**
   The strong causal past of t w.r.t β' is $[t]_{\beta'} = \{t' \in Events(\beta') \mid t' \preceq t\}$ and it's weak causal past w.r.t. β' is $[t] = \{t' \in Events(\beta) \mid t' \leq t\}$. We can verify that $[t] = \bigcap_{\beta'} [t]_{\beta'}$ and we call the weak causal past of t as the minimal causal past of t. □

   $[t]$ can also be inductively defined by $[t] = \{t\} \cup \bigcup_{t' {}^\bullet \in ({}^\bullet t \cup {}^\circ t)} [t']$. This says that what is minimally needed to fire t is the union of t itself (obviously) and what is minimally need to fire t', where t' activates whatever t *inputs* or *reads* from.

4. **Extension of β' by β^+ or $\beta' \sqsubseteq \beta^+$**
   Given 2 configurations C', C^+ of β, we say that C^+ extends C', given by $C' \sqsubseteq C^+$ iff :

   - $C' \subseteq C^+$
   - $\forall t' \in C', \forall t^+ \in C^+ \backslash C'$ we don't have $t^+ \prec t'$.

We say that $\beta' \sqsubseteq \beta^+$ if $C' \sqsubseteq C^+$ and if $Y = C^+ \backslash C'$ then we say $C^+ = C' \oplus Y$. □

We don't enforce that $t' \prec t^+$ because they may be concurrent events. So, in short, t^+ is an extension if $t' \prec t^+$ or $t^+ || _{\preceq} t'$. We use $\preceq$ because of the presence of read-arcs. Otherwise, in a case like Figure 1, we'll have t' extends the NBP $\beta_{t,t''}$, while it actually cannot do so in an execution.

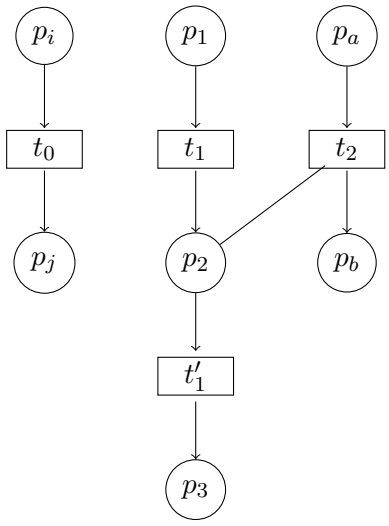

Figure 7: NBP example for Definition

Note that $[t]_{\beta'}, [t]$ may differ due to the presence of read arcs. In the Figure 5, because of the definitions of $<$ as compared to $\preceq$ given earlier, $[t]_{\beta'}$, the causal past of t'_1 in the given NBP (called β') includes $\{t_1, t'_1, t_2\}$ but one can note that the minimal causal past $[t]$ need not include t_2. You could have an NBP without it as well, and this is exactly what *minimal* causal past captures.

In the Figure 5, $C = \{t_0, t_1, t'_1, t_2\}$, $Cut(C) = \{p_j, p_3, p_b\}$, $[t'_1] = \{t_1, t'_1\}$, $[t_2] = \{t_1, t_2\}$ and the NBP can be extended by any element of the cut. Also, the given NBP extends several others, whose configurations can be given by $\{t_0, t_1, t_2\}$ or $\{t_1, t_2, t'_1\}$ or $\{t_1, t_2\}$ or $\{t_0\}$, etc, but it does not extend the NBP with configuration $\{t_0, t_1, t'_1\}$ as we have $t_2 < t'_1$.

Lemma 2.2. *Let $\mathcal{A}$ be an NTA, $\beta = (\mathcal{N}, \lambda)$ be a BP on $\mathcal{A}$, then if c is a cut of β then c satisifies the following properties:*

- *$Cut(\beta) \cap \lambda^{-1}(L)$ consists of one location per TA of the NTA.*

- $\lambda^{-1}(X)$ *is in bijection (by* λ*) with the set* X *of clocks*

Proof. M_0, our starting point in a BP, is a cut obeying these rules. We must just show that everytime we add an event, this property is not destroyed. This is because of the definitions of Pre and $Post$ functions that are used. Accordingly, whenever a clock is consumed, it is reproduced, and everytime a place whose label is a location of a TA is consumed, another place with label as another location from the same TA is produced. Hence, if we start off with the above 2 rules obeyed, they will be obeyed after every step (by easy induction). □

Lemma 2.3. *Let* $\mathcal{A}$ *be an NTA,* $\beta = (\mathcal{N}, \lambda)$ *be a BP on* $\mathcal{A}$*. Consider an event* t *of* β*. Then, given 2 cuts* c *and* c' *of* β *verifying* ${}^\bullet t \cup t^\circ \subseteq c$ *and* ${}^\bullet t \cup t^\circ \subseteq c'$*, then we have* $[\![Inv(Loc(c))]\!] = [\![Inv(Loc(c'))]\!]$*.*

Proof. By definition of $Read(\bar{t})$ and $Pre(\bar{e})$, every clock $x \in X_{inv}$ is represented in ${}^\bullet t \cup t^\circ$. Also, by examining the rules behind these definitions, we can easily see that whenever we fire a transition, if the invariant constraint for a particular clock is modified or the value of the clock is modified, then the condition with respect to this clock is consumed and a new condition is produced. Hence, 2 cuts having the same subset ${}^\bullet t \cup t^\circ$ necessarily contain the same invariant constraints and the same values of clocks for the all the invariant clocks (X_{inv}), which demonstrates the result. □

2.2.5 Definition 10 (Possible Extensions (PE))

Let $\beta = ((N), \lambda)$ be a branching process of an NTA (A). The possible extensions of β are triplets $t = (\bar{e}, Y_{in}, Y_r)$ where

- $\bar{e}$ is an element of Sync
- $Y_{in} \cup Y_r$ is a $co_{\preceq}$-set of conditions of β
- λ is one to one mapping from Y_{in}, Y_r to $Pre(\bar{e}), Read(\bar{e})$
- $(\bar{e}, Y_{in}, Y_r)$ does not already belong to β.

We then define the extension of β by t, obtained by $Extend(\beta, t)$, as the branching process obtained from β by adding the event $\bar{e}$, connected to conditions in Y_{in} with pre-arcs and conditions in Y_r with read-arcs, and with new conditions according to $Post(\bar{e})$. □

It is important to note here, that PEs are extended from a $co_{\preceq}$-set of conditions and not necessarily a $maximal_{\preceq}\ co_{\preceq}$-set! This may mean that the PE builds from the middle of the branching process too! This is necessary for the correct operation of PE in the unfolding algorithms.

In Figure 5, assume that till now, only t_0 and t_1 have been built. PE will give t_1', t_2 as its options. Now, because of the definition of PE being based on co-set and not cut, it does not matter which order you choose from t_1', t_2, as even if t_1' is built first, p_2, p_a will still be related by the co- relation, and hence t_2 will come up as an option in PE the next time as well.

3 Algorithms

This section will deal with the algorithms (and related definitions, intuitions, ideas, theorems and proofs) that we will require to obtain a finite and complete prefix of the unfolding of an NTA.

First, we shall give the most basic, naive semi-algorithm that can be used in the untimed framework. Based on this, we shall use the concept of *zones* to develop another semi-algorithm for timed frameworks. We shall then explore the reasons of non-termination, what problems we face when trying to enforce termination, and how we cleverly compromise and overcome these problems, to end up with the final algorithm for the required prefix.

3.1 Getting Started

3.1.1 Algorithm (Untimed Unfolding Semi-Algorithm)

By untimed, we mean that we forget about all guards, resets and invariants, but we do keep all the read arcs. Also, we need to choose the event from *pe* (Line 4) in an efficient way (say, using a queue) because it is possible that given 2 independent/concurrent automata, the algorithm will keep choosing events from (and extending) only one of them, which is not what we would like.

This simple semi-algorithm will build the eventually infinite untimed unfolding of the NTA $\mathcal{A}$, which is also its maximal BP. There is no guarantee that the algorithm will terminate! Nevertheless, this is the basic exploration algorithm we shall base ourselves on, making improvements along the way.

In the algorithm, each condition is encoded as (p, t) where p is the label condition/clock and t is the unique input event for this location (note that a BP was defined through occurrence nets, which had the condition $|{}^\bullet p| \leq 1$, which makes sure that the event which created it is unique!).

Algorithm 1 Building the untimed unfolding (eventually infinite)

Require: An NTA $\mathcal{A}$
Ensure: The unfolding Unf of $\mathcal{A}$
1: $Unf := \{(l_{1,0}, \emptyset), \ldots, (l_{n,0}, \emptyset)\} \cup \{(x, \emptyset) \mid x \in X\}$;
2: $pe := PE(Unf)$;
3: **while** $pe \neq \emptyset$ **do**
4: Choose an event $t = (\bar{e}, Y_{in}, Y_r)$ in pe
5: $Extend(Unf, t)$;
6: $pe := PE(Unf)$;
7: **end while**

3.2 Including Time

Consider an NTA $\mathcal{A} = ((\mathcal{A}_i)_{1 \leq i \leq n}, f)$ (as defined earlier in the paper). Any configuration of the NTA is given by $(\bar{l}, v)$ where $\bar{l} \in \Pi_{1 \leq i \leq n} L_i$ and $v \in \mathbb{R}^X$. Then, a timed execution can be represented as :

$$(\bar{l}_0, v_0) \xrightarrow{\bar{t}_1, d_1} (\bar{l}_1, v_1) \xrightarrow{\bar{t}_2, d_2} \ldots \xrightarrow{\bar{t}_p, d_p} (\bar{l}_p, v_p)$$

where for all i we have $\bar{t}_i \in Sync$ and $d_i \in \mathbb{R}$ verify $d_i \leq d_{i+1}$. Intuitively, d_i represents the date at which t_i was taken. We associate, with this execution, a timed sequence $\sigma = (\bar{t}_i, d_i)_{1 \leq i \leq p}$, a finite word on the alphabet $Sync \times \mathbb{R}$.

We would like to represent an execution of the NTA in terms of Petri Nets using the above defined structures. For this purpose, we now introduce the concept of a Timed NBP.

3.2.1 Definition (Timed Non-Branching Process (TNBP))

Let β' be an NBP of NTA $\mathcal{A}$ having events T and conditions/places P. We define the variables of the TNBP as $D(\beta') = Var(P \cup T)$ where

- $\forall t \in T, Var(t) = \{d_t(t)\}$
- $\forall p \in P \cap \lambda^{-1}(L)$, $Var(p) = \{d_c(p)\}$
- $\forall p \in P \cap \lambda^{-1}(X), Var(p) = \{d_b(p), d_e(p), d_r(p)\}$

A timed NBP is then defined as a pair (β', d) where d is a valuation of $D(\beta')$, ie, an element of $\mathbb{T}^{D(\beta')}$. □

The following dates are absolute dates:

- d_t is date of transition

- d_c is date of creation (of a place/location)
- d_b, d_e, d_r is date of birth, end and last reset of clock

If the clock has not been consumed yet, then its d_e has the value $+\infty$. Also, the date of consumption of a place has not been introduced as a variable, as it is always equal to the date of the transition (when the TNBP is "feasible", as introduced next), making it a redundant variable.

This TNBP now almost corresponds (via λ) to a single real timed execution of the NTA $\mathcal{A}$ from which it was created. We just have to satisfy the conditions which appear in the next definition, ie, presently, the TNBP may be feasible or infeasible. Next, we introduce the concept of feasibility of the timed valuation, and inductively define a TNBP based on that.

3.2.2 Definition (TNBP of σ)

Let $\mathcal{A}$ be an NTA and σ a timed sequence of $\mathcal{A}$. The TNBP $Pr - T(\sigma) = (\beta', d),\ d = (d, d_b, d_e, d_r, d_c)$ associated with σ is then inductively defined as follows:

- If σ is the empty sequence ϵ then β' is $Min(P)$ where
  - $\forall p_x \in Min(P) \cap \lambda^{-1}(X),\ d_b(p_x) = 0, d_e(p_x) = \infty, d_r(p_x) = 0$
  - $\forall p_l \in Min(P) \cap \lambda^{-1}(L),\ d_c(p_l) = 0$
- If $\sigma = \sigma'(\bar{e}, d)$ (d is the date of occurrence of $\bar{e}$) and $(\beta^-, v^-), v^- = (d^-, d_b^-, d_e^-, d_r^-, d_c^-)$ is the TNBP of σ' then there is a unique possible extension β' of β^- from $Cut(\beta^-)$, c^-, by an event t labeled $(\bar{e}, Y_{in}, Y_r)$ (having $Y_{in} \cup Y_r \subseteq c^-$, such that
  - The timed valuation v^- is preserved except for places $p_x \in {}^\bullet t$, for which we set $d_e(p_x) = d$.
  - We set $d(t) = d$, and for every place $p_l \in t^\bullet \cap \lambda^{-1}(L)$, we set $d_c(p_l) = d$.
  - Also, $\forall p_x \in t^\bullet \cap \lambda^{-1}(X)$, we set $d_b(p_x) = d$ and $d_e(p_x) = \infty$ and we set $d_r(p_x) = d$ if it was reset, and $d_r(p_x) = d_r(p_x^-)$ otherwise (where p_x^- is the unique place in C^- whose label is x).

□

3.2.3 Definition ($Conf(\beta', d)$)

Let $\mathcal{A}$ be an NTA. Given an NBP (β', d) we define the configuration $Conf(\beta', d)$ of the NTA by the pair $(\bar{l}, v)$ where:

- $l = l(Cut(\beta'))$

- $\forall x \in X, v(x) = v(d_b(p_x)) - v(d_r(p_x))$

where $p_x = p_x(Cut(\beta'))$ is a unique condition in $Cut(\beta')$ labelled by x. □

Lemma 3.1. *Let $\mathcal{A}$ be an NTA and σ be the timed sequence associated with the execution*

$$(\bar{l}_0, v_0) \xrightarrow{\bar{t}_1, d_1} (\bar{l}_1, v_1) \xrightarrow{\bar{t}_2, d_2} \dots \xrightarrow{\bar{t}_p, d_p} (\bar{l}_p, v_p)$$

Then we have the following equality :

$$(\bar{l}_p, v_p) = Config(Pr - T(\sigma))$$

□

3.2.4 Definition (Feasibility of Timed Valuation d)

Here, we will use the following notations:

- $c^+(t), c^-(t)$ are the cuts corresponding to the configurations $[t], [t]\backslash\{t\}$ respectively.
- $L^+(t), L^-(t)$ are the locations $c^+(t) \cap \lambda^{-1}(L), c^-(t) \cap \lambda^{-1}(L)$ respectively.
- $D(t) = Var(\{t\} \cup c^- \cup c^+)$ is the set of variables which Z_t depends on, and $D_0 = Var(Min(P))$. We then define $D(\beta) = D_0 \cup \bigcup_{t \in T} D(t)$ for an NBP β.
- p_x^+, p_x^- are the unique places corresponding to clock x in $c^+(t), c^-(t)$ respectively.
- Finally, we have $v_x^-(t) = d_t(t) - d_r(p_x^-)$ and $v_x^+(t) = d_t(t) - d_r(p_x^+)$.

First, we define the zone Z_t, defined on the variables $D(t)$ as the zone associated with the conjunction of the following constraints:

Causal (in)equations:

- $\forall p_l \in t^\bullet \cap \lambda^{-1}(L),\ d_c(p_l) = d_t(t)$
- $\forall p_x \in t^\bullet \cap \lambda^{-1}(X),\ d_b(p_x) = d_t(t), d_e(p_x) = \infty$
- $\forall p_x \in\ ^\circ t,\ d_b(p_x) \leq d_t(t) \leq d_e(p_x)$
- $\forall p_l \in\ ^\bullet t \cap \lambda^{-1}(L),\ d_c(p_l) \leq d_t(t)$
- $\forall p_x \in\ ^\bullet t \cap \lambda^{-1}(X),\ d_b(p_x) \leq d_t(t) = d_e(p_x)$

Timed (in)equations:

- $g(\lambda(t))[\{x \leftarrow v_x^-(t)\}_{x\in X}]$
- $\bigwedge_{l\in L^-(t)} Inv(l)[\{x \leftarrow v_x^-(t)\}_{x\in X}]$
- $\bigwedge_{x\in R(\lambda(t))} v_x^+(t) = 0$
- $\bigwedge_{x\in Redefined(\lambda(t))} v_x^+(t) = v_x^-(t)$

Next, we define the zone Z_0 on the variables D_0 as the zone associated with the conjunction of constraints :

- $\forall p_l \in Min(P') \cap \lambda^{-1}(L),\ d_c(p_l) = 0$
- $\forall p_x \in Min(P') \cap \lambda^{-1}(X),\ d_b(p_x) = 0 = d_r(p_x)$

Then, given an NBP β, we define its zone $W_\beta = Z_0 \cap (\bigcap_{t\in T} Z_t)$ on the variables $D(\beta)$. As mentioned earlier, the bijection between configurations and NBPs allows us to define the zone W_C associated with a configuration as the zone W_β with $[\beta] = C$.

Finally, we say that a timed valuation $d = (d_t, d_b, d_e, d_r, d_c)$ of NBP β' is $feasible$ (or (β', d) is $feasible$) iff $d \in W_{\beta'}$. □

Remark Given a zone Z_t on the variables $D(t)$, and a valuation d of all these variables, we have :

$$d \in Z_t \Rightarrow \forall \delta \in \mathbb{R}, d + \delta \in Z_t$$

□

Proposition 3.2. *(Feasibility is equivalent to execution) Let $\mathcal{A}$ be an NTA and (β, d) be a TNBP of $\mathcal{A}$. Then we have :*

(β, d) is feasible $\iff$ there exists a timed sequence σ of $\mathcal{A}$ s.t. $Pr - T(\sigma) = (\beta, d)$.

In other words, we have the following equality :

$W_\beta = \{v \in \mathbb{T}^{D(\beta)} \mid \exists \sigma$, timed sequence of $\mathcal{A}$ such that $Pr - T(\sigma) = (\beta, d)\}$ □

Lemma 3.3. *Given an NTA $\mathcal{A}$, an event t of its unfolding, we define $Max_{<t} = \{t' \in [t]\ \{t\} \mid \forall t'' \in [t]\ \{t\}, t' \not< t''\}$ Then we have*

$$W_{[t]} = Z_t \cap \bigcap_{t' \in Max_{<t}} W_{[t']}$$

□

3.2.5 Algorithm (Timed Unfolding Semi-Algorithm with W_β)

Algorithm 2 Building the Timed Unfolding (Eventually infinite)

Require: An NTA $\mathcal{A}$
Ensure: The timed unfolding $T-Unf(\mathcal{A})$ of $\mathcal{A}$
1: $T-Unf := \{(l_{1,0},\phi),\ldots,(l_{n,0},\phi)\} \cup \{(x,\phi)|x \in X\}$;
2: $pe := PE(T-Unf)$;
3: **while** $pe = \phi$ **do**
4: Choose an event $t = (\bar{e}, Y_{in}, Y_r)$ in pe.
5: Compute the zone $W_{[t]}$ using Lemma <>.
6: **if** $W_{[t]} \neq \phi$ **then**
7: $Extend(T-Unf, t)$;
8: $pe := PE(T-Unf)$;
9: **else**
10: Mark t as useless event.
11: **end if**
12: **end while**
13: **return** $T-Unf$

Theorem 3.4. *Let $\mathcal{A}$ be an NTA. Algorithm 2 calculates its timed unfolding $T-Unf(\mathcal{A})$. This algorithm, which need not terminate, has the following properties:*

- *An event occurs in the unfolding iff there is at least one timed execution of $\mathcal{A}$, associated with a timed sequence σ, whose TNBP $Pr-T(\sigma) = (\beta, d)$ satisfies $[\beta] = [t]$.*
- *The zone $W_{[t]}$ associated with an event t of $T-Unf(\mathcal{A})$ projected onto the variables $d_t(t')$ for $t' \leq t$, represents the possible dates for the timed sequence σ whose NBP is $[t]$.* □

This algorithm builds the (possibly infinite) timed unfolding of an NTA and its basic idea is introduced in Algorithm 1.

Unlike Algorithm 1, we associate with each event t of the unfolding, the zone $W_{[t]}$. We add the event t only if $W_{[t]}$ admits a solution (line 6-9). Otherwise we mark it as useless (line 10).

Using this algorithm as our basic idea, we introduce some concepts and approximations to ensure termination without harming reachability, which will finally result in Algorithm 5.

3.2.6 Definition (Local Zones W_β^{loc})

Given an NTA $\mathcal{A}$ and an NBP β of the unfolding of $\mathcal{A}$, we consider the clocks $Loc - D(\beta) = D(Cut(\beta))$, which are all variables of $Cut(\beta)$. The local zone associated with β, W_β^{loc}, is then defined as the projection of the zone W_β on the clocks $Loc - D(\beta)$. □

The number of clocks defining the local zone W_β^{loc} is equals to $n + 3|X|$, where n is the number of automata and $|X|$ is the number of clocks of $\mathcal{A}$ (since the number of variables per location is 1, and the number of variables per clock is 3, and there are n locations and $|X|$ clocks).

Note that comparision of local zones directly is not sufficient as they don't hold sufficient information and will hence lead to incorrect conclusions. This is shown in the following example.

Example Consider the NTA $\mathcal{A}$ given in Figure **fill** whose unfolding is given in Figure **fill**. We let β_1 and β_2 represent the NBP of configurations $\{t_1, t_2, t_3, t_4\}$ and $\{t_1, t_2, t_3, t_5\}$ respectively. A simple analysis of $\mathcal{A}$ leads to the following conclusions :

- The action b can have any date, zero or positive.
- The action c_1 is permissable iff b has a date 1.
- The action c_2 is permissable iff b has a date 0.
- The two actions c_1, c_2 cannot appear in the same timed sequence.

This analysis shows that there cannot exist a feasible NBP whose configuration has both t_4 and t_5. In the unfolding of $\mathcal{A}$, the local zones $W_{\beta_1}^{loc}$ and $W_{\beta_2}^{loc}$, associated with the NBPs β_1 and β_2 respectively, use disjoint sets of variables (ie, $Loc - D(\beta_1) \cap Loc - D(\beta_2) = \emptyset$). This is because the $Cut(\beta_1) \cap Cut(\beta_2) = \emptyset$. Hence, we obtain the intersection $W_{\beta_1}^{loc} \cap W_{\beta_2}^{loc} \neq \emptyset$ (we cannot reach a contradiction since the 2 sets of variables used is distinct). Hence, it does not give the correct conclusion about the feasability of a configuration $\{t_1, t_2, t_3, t_4, t_5\}$. Hence, it is not sufficient to just take the intersection of the local zones. □

Lemma 3.5. *Let $\mathcal{A}$ be an NTA and β be an NBP of the unfolding of $\mathcal{A}$. Let $T = \{t_1, \ldots, t_k\}$ be a set of events of $Unf(\mathcal{A})$ which satisfy the following properties :*

- $\forall 1 \leq i \leq k,\ {}^\bullet t_i \cup\ {}^\circ t_i \subseteq Cut(\beta)$
- $\forall 1 \leq i \neq j \leq k, t_i - co_< - t_j$

We say that T is a slice extending β. Also, denoting $\beta_0 = \beta$, we have for all $i \in \{1, \ldots, k\}$, t_i extends the NBP β_{i-1} and we denote $\beta_i = Extend(\beta_{i-1}, t_i)$. Then, we claim that the zone $W^{loc}_{\beta_k}$ is a projection on the clocks $Loc - D(\beta_k)$ of the zone $W^{loc}_{\beta} \cap (\bigcap_{1 \leq i \leq k} Z_{t_i})$.
We denote, from here onwards, $Extend(\beta, T)$ of NBP β by the slice T, as the extension β_k by the events of T. □

Lemma 3.6. *Let $\mathcal{A}$ be an NTA and β and β' be 2 NBPs of the unfolding of $\mathcal{A}$ such that $\beta \sqsubseteq \beta'$. We denote $T = [\beta'] \backslash [\beta]$ as the set of events present in $[beta']$ but absent from $[\beta]$. Then, there exists a partition of T into k pairwise disjoint subsets $\{T_i, \ldots, T_k\}$, such that for all indices $i \in \{1, \ldots, k\}$, T_i is a maximal slice extending the process β_{i-1} to the process β (assuming $\beta_0 = \beta$) and satisfies $\beta_k = \beta'$.* □

3.2.7 Algorithm (Updating NBP : *fire*)

Algorithm 3 Update NBP by firing concurrent events : *fire*

Require: A quadruplet (β, C, W, T) where β is an NBP with cut C, local zone $W = W^{loc}_{\beta}$ and T is slice extending β.
Ensure: The triplet (β', C', W') with β' being the extension of β (on firing T), with cut C' and local zone $W' = W^{loc}_{\beta'}$.
1: $C := (C \backslash {}^\bullet T) \cup T^\bullet$;
2: Let $\beta' = \beta$; Let ϕ be the constraint (on $Loc - Var(\beta)$) representing W.
3: **for** $t \in T$ **do**
4: $\beta' := Extend(\beta', t)$
5: Define the constraint ϕ_t, defined by the zone Z_t
6: **end for**
7: $\phi' := \phi \wedge (\bigwedge_{t \in T} \phi_t)$;
8: $W_{\beta'} := [\phi']$
9: Project $W_{\beta'}$ on $Var(Cut(\beta'))$ to get $W' = W^{loc}_{\beta'}$.
10: **return** (β', C', W')

We can calculate the new cut straightaway (line 1). For every transition in the concurrent set T, extend the NBP β' to include the event and calculate the constraints ϕ_t associated with it (lines 3-6). Conjunct these to the overall

set of constraints and calculate the corresponding zone $W_{\beta'}$ reached (lines 6-8). Project this on the variables of the cut to achieve the local zone $W^{loc}_{\beta'}$, which is what we maintain.

3.2.8 Algorithm (Computation of local Zone)

Algorithm 4 Calculation of the new local zone associated with a PE

Require: A possible extension $t = (\bar{e}, Y_{in}, Y_r)$
Ensure: The zone $W^{loc}_{[t]}$ associated with the firing of t

1: Compute the maximal co-set of events $t_1, \ldots, t_m$ w.r.t $<$ of $[t]\backslash\{t\}$
2: **for** $i \in 1, \ldots, m$ **do**
3: $t' := t_i$;
4: Let β' be the non-branching process of $[t']$
5: **while** $\exists p \in \beta'\backslash Cut(\beta'), \exists t'' \in [t]\backslash[t'],\ s.t.\ p \in {}^\circ t''$ **do**
6: $t' := {}^\bullet p$;
7: **end while**
8: $t'_i := t'$;
9: **end for**
10: Select $k \in \{1, ..., m\}$ such that the size of $[t'_k]$ is maximal
11: Let $C = Cut([t'_k])$ and $W' = W^{loc}_{[t'_k]}$
12: Let β' be the non-branching process of $[t'_k]$
13: Compute a topological sort $[T_1, \ldots, T_m]$ w.r.t. $<$ of $[t]\backslash(t \cup [t'_k])$
14: **for** $j \in \{1, ..., m'\}$ **do**
15: $(\beta', C, W') := fire(\beta', C, W', T_j)$;
16: **if** $[Z] = \phi$ **then**
17: **return** ϕ;
18: **end if**
19: **end for**
20: $(\beta', C, W^{loc}_{[t]}) := fire(\beta', C, W', t)$;
21: **return** $W^{loc}_{[t]}$;

Aim
The aim of this algorithm is to compute the local zone $W^{loc}_{[t]}$ associated with the firing of t. This is how our thought process proceeds :

Isn't it trivial?
Since we don't keep the entire equation system, but just a projection of it, computing $W^{loc}_{[t]}$ is not trivial. We compute additional zones associated with intermediate non-branching processes to help us out.

What can we do easily?
A first remark is that given the zone $W^{loc}_{\beta'}$ corresponding to some non-

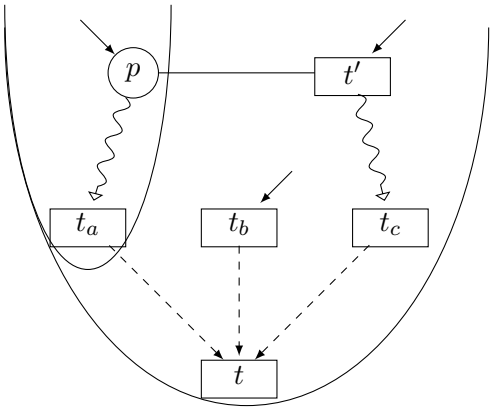

Figure 8: Building new Zone - intuition!

branching process β', and an extension β^+ created by firing a set of concurrent events, it is easy to compute the local zone $W^{loc}_{\beta^+}$, simply by applying Algorithm 3.

From what point can we fire concurrent slices?
Suppose we are trying to find out what we call t'_a, the best (correct and zero-redundancy) location in the causal past of t_a from which I could fire slices to obtain my required zone, as shown in the figure. From t'_a, we will fire all events which are not in it's past, and hence not already accounted for.

What's the problem in taking $t'_a = t_a$?
Notice that if there is a location p in the past of t_a which is being read by an event t' which is not in the past of t_a, but is in the past of t (hence t' would have to be fired after t_a and before t), then $t'_a = t_a$ will not do because I will not be able to fire t' (since it reads from a place that has already been consumed). Infact, t'_a can maximum be ${}^\bullet p$ or something earlier. This is true because t_a or any other such event after p would have already consumed p and hence t' cannot be fired after that!

Okay! How is this achienved?
Lines 1-9 of the algorithm do exactly this. Given the maximal set T (line 1) of concurrent events, we want to calculate the possible correct events from which to start firing slices of concurrent events, before we fire t. With the above mentioned idea, from each $t_i \in T$ (done by lines 2-9), we calculate such events t'_i (done by lines 5-7) in its causal past. Then, out of all these possibilites, line 10 will choose t'_k, the one with the largest causal past (so that we have to fire the least number of events, to make it less redundant).

Complicated! Can you give me a concrete example?

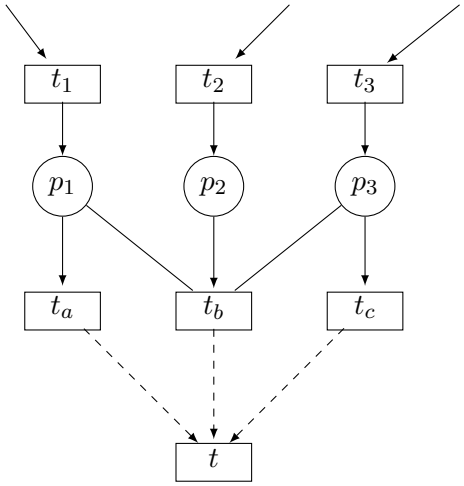

Figure 9: Concrete example for zone building

Look at the Figure 7. Suppose we want the local zone associated with the firing of event t. We must start of from the zone of some already fired event. But, not any event will do! If we consider the zone of NBP $[t_a]$ as our starting point, we clearly cannot fire t_b as the place which it reads from has already been consumed! The same logic applies to t_c. The reason behind this, as mentioned above, is that there is a place p_1 in the past of t_a which is being read by t_b (which must be fired, since it is not already in $[t_a]$).

Hence, for t_a the best event to start off firing concurrent slices from, $t'_a = {}^\bullet p_1 = t_1$ (from which it can fire $\{t_2, t_3, t_b, t_a, t_c, t\}$). It is a symmetric case for t_c Similarly for t_b, we get $t'_b = t_b$ itself (from which it can fire $\{t_a, t_c, t\}$). Now, having calculated the best options for each of the 3 automata, we choose the best option, which is t_b, since starting from here would involve firing the least number of events, and hence is a correct and zero-redundant to start from!

Understood! Please proceed!
We now start with the NBP associated with t'_k, ie $[t'_k]$, its cut C and it's associated local zone $W^{loc}_{[t'_k]}$. Line 13 then calculates the concurrent slices that require to be fired using a topological sort over $<$. The for loop (lines 14-19) then fire these slices one by one, using the Algorithm 3. If at any point, we get a zone which is equivalent to the null-set ϕ, we abandon the process and return the zone obtained as ϕ. Otherwise, we go on and fire the last event which is t, and return our obtained zone.

And we're done!

3.2.9 Algorithm (Timed Unfolding Semi-Algorithm with W^{loc}_{β})

Algorithm 5 Building the Timed Unfolding (Eventually infinite)

Require: An NTA $\mathcal{A}$
Ensure: The timed unfolding $TL-Unf(\mathcal{A})$ of $\mathcal{A}$
1: $TL-Unf := \{(l_{1,0},\phi),\ldots,(l_{n,0},\phi)\} \cup \{(x,\phi)|x \in X\}$;
2: $pe := PE(TL-Unf)$;
3: **while** $pe = \phi$ **do**
4: Choose an event $t = (\bar{e}, Y_{in}, Y_r)$ in pe.
5: Compute the zone $W^{loc}_{[t]}$ using Algorithm <>.
6: **if** $W^{loc}_{[t]} \neq \phi$ **then**
7: $Extend(TL-Unf, t)$;
8: $pe := PE(TL-Unf)$;
9: **else**
10: Mark t as useless event.
11: **end if**
12: **end while**
13: **return** $TL-Unf$

Theorem 3.7. *Let $\mathcal{A}$ be an NTA. Algorithm <> calculates its timed unfolding $TL-Unf(\mathcal{A})$. This algorithm, which need not terminate, has the following properties:*

- *An event occurs in $TL-Unf(\mathcal{A})$ iff there is at least one timed execution of $\mathcal{A}$, associated with a timed sequence σ, whose TNBP $Pr-T(\sigma) = (\beta, d)$ satisfies $[\beta] = [t]$.*
- *The zone $W^{loc}_{[t]}$ associated with an event t of $TL-Unf(\mathcal{A})$ represents the possible values for the dates/clocks associated with the $Cut([t])$ obtained by timed sequences σ whose TNBP satisfies $Pr-T(\sigma) = ([t], d)$.* □

3.3 Finite & Complete Prefix

3.3.1 Finite Prefix

Consider an NTA $\mathcal{A}$ and let $\mathcal{N}$ be a subset of $Unfolding(\mathcal{A})$. Then, we say that $\mathcal{N}$ is a prefix of $Unfolding(\mathcal{A})$ iff it verifies the following conditions :

- $Min(T-Unf(\mathcal{A})) \subseteq \mathcal{N}$
- ${}^{\bullet}\mathcal{N} \cup {}^{\circ}\mathcal{N} \subseteq \mathcal{N}$

Also, if $\mathcal{N}$ is finite, we say that $\mathcal{N}$ is a finite prefix of $T-Unf(\mathcal{A})$. □

3.3.2 Complete Prefix

Consider an NTA $\mathcal{A}$ and let $\mathcal{N}$ be a prefix of $Unfoldin(\mathcal{A})$. Then, we say that $\mathcal{N}$ is a complete prefix of $Unfolding(\mathcal{A})$ iff it satisfies the condition:

$$\forall(\bar{l}, v) \in Acc(\mathcal{A}), \exists(\beta, d),\ feasible,\ s.t.[\beta] \in \mathcal{N},\ andConf(\beta, d) = (\bar{l}, v).$$

where $Acc(\mathcal{A})$ represents the set of accessible configurations from the initial configuration in the NTA $\mathcal{A}$. □

Corollary 3.8. *The unfoldings $T - Unf$ and $TL - Unf$ obtained by the algorithms 2 and 5 respectively are complete, but not necessarily finite.* □

3.3.3 Ideas for achieving finiteness! How do we proceed?

What was the idea in the untimed case?
In the untimed framework, when the marking associated with an event t is equal to the marking associated with an event t' and [t] $\ll$ [t'] for some well-founded order defined over configurations (for instance, the size of the history), one does not explore processes extending [t'] since "equivalent" processes can be built that extend [t]. Note that $\ll$ must extend the inclusion order, and be preserved by finite extensions.

So, why can't we use this idea in the timed case?
In the timed framework, we must in addition take into account the zones associated with the two events for checking whether "equivalent" processes can always be built. However, in the context of TA, it is well-known that there are infinitely many incomparable zones.

Infinite? Can we do something?
Yes, for this very purpose, an extrapolation operator has been designed, which bounds the number of zones which can be computed, thus enforcing termination of the forward reachability computation. This extrapolation is an over-approximation, but is correct for checking reachability properties.

Great! Can we use it directly?
No! To compare the configurations reached after the firing of two events t and t', we cannot use directly the zones W_t^{loc} and $W_{t'}^{loc}$ 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.

Oh! We're stuck!
Not really! We do have enough information at hand! We can compute the new zone $Relativise(t)$ 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]. This zone is a true reflection of what future states can be reached, and hence can be compared!

And this is where extrapolation comes in?
Yes! To enforce termination, we then apply the classical extrapolation operator on this last zone $Relativise(t)$ and get the so-called clock zone $Test_t$. We shall perform our comparisions with these zones!

Any other problems we overlooked?
Yes, unfortunately! From two events t and t' whose clock zones and cuts are similar, we can have 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' does not have to be greater 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.

And I guess you have a solution to this too?
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 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.

Fantastic idea! Let's get to work...

3.3.4 Definition ($Prim(t)$)

Let C be a configuration (with respect to the unfolding $Unf(\mathcal{A})$ of $NTA\mathcal{A}$). C is called primitive if it admits a unique maximal event t according to the strong causal relation $\preceq$ of $Unf(\mathcal{A})$.
The primitive configurations of a maximal event t is denoted by $Prim(t)$.
Also, we call an NBP primitive if it's configuration is primitive. □

Remark For a primitive configuration C of a maximal event t, all feasible TNBPs (β, d) such that $[\beta] = C$ satisfy the property:

$$\forall d \in D(\beta), d \neq d_e(p_{Cut(\beta)}) \Rightarrow d_t(t) \geq d$$

□

Lemma 3.9. *If C is a configuration and $t \in C$ is an event. Then there exists a configuration $C' \in Prim(t)$ satisfying $C' \sqsubseteq C$.*

Proof. It is sufficient to consider $C' = \{t' \in C \mid t' \preceq t\}$ □

3.3.5 Definition (Relativised Zone of C ($Relativise(C)$))

Remark

3.3.6 Definition (Unavoidable Edges (UE))

Let $\mathcal{A} = (\mathcal{A}_i)_{1 \leq i \leq n}$ be an NTA and E be a subset of global edges of $\mathcal{A}$ (i.e. a subset of $Sync$). E is unavoidable iff

- For all i, every circuit of A_i intersects E (ie to say, it cannot be completed without edges from E).
- For all i, there is some $e_{i,c}$ belonging to each circuit c in A_i such that if $e_{i,c}$ occurs in $e \in Sync$ then $e \in E$. □

This is trying to specify a set of edges which is unavoidable on the long run. If I have a long enough path, I would have had to take a loop/circuit, and hence by definition, would not be able to avoid these edges. Any NTA has at least one unavoidable subset of edges, the set Sync itself.

We shall now transform the NTA (into an SNTA) in such a way that when one fires an edge of E, one synchronizes the whole NTA.

Lemma 3.10. *The two definitions of UE are equivalent.*

Proof. It is clear that 2 implies 1. Now, assume that 1 does not imply 2, ie every circuit of A_i intersects E, but at least for $i = i_1$, there is a circuit c_a in A_{i_1} for which the following holds : $\forall j_{0 \leq j \leq len(c_a)}$ for every edge $e_{i_1,c_a,j}$ in c_a, there is an edge $e_j \in Sync$ involving $e_{i_1,c_a,j}$ but $e_j \notin E$. Then, it would hav been able to form the circuit consisting of all edges e_j, such that the circuit c_a of A_{i_1} does not intersect E, which contradicts our assumption of 1. □

3.3.7 Definition (Synchronized NTA (SNTA))

Let $\mathcal{A} = (A_i)_{1 \leq i \leq n}$ be an NTA and UE be an unavoidable set of edges of $\mathcal{A}$, then define :

- $\forall \bar{e} \in UE$, $EdgeSync(\bar{e}) = \{\bar{f} \mid \forall i \in I(\bar{e}), f_i = e_i, and$
  $\forall i \notin I(\bar{e})\ \exists l_{i,j} \in L_i, f_i = l_{i,j} \overset{true,\epsilon,\phi}{\longrightarrow} l_{i,j}\}$
- SNTA $\mathcal{A}_{Sync-UE}$ is the NTA where UE has been replaced by $UE' = \bigcup_{\bar{e} \in UE} EdgeSync(\bar{e})$. □

The point of an SNTA is that firing any transition of UE' now synchronizes the whole automaton without changing any properties. Edge loops get added to a lot of locations, but these can be taken only if such an edge exists in UE'. This clearly increases the number of possible edges by a large amount. Note that $\forall \bar{e} \in UE, I(\bar{e}) = \{1, \ldots, n\} \Rightarrow \mathcal{A}_{Sync-UE} = \mathcal{A}$

For example, <fill in>

$\mathcal{A}_{Sync-UE}$ is not defined via a synchronization function but directly with its set of edges. However all previous results equally apply on such NTA. $\mathcal{A}$ and $\mathcal{A}_{Sync-UE}$ 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 $\mathcal{A}_{Sync-UE}$. This in particular the case for reachability, and event occurrence which are the usual properties checked by the unfolding method.

Lemma 3.11. *If t is a synchronized event and C' is a configuration satisfying $t \in C'$. The following properties are verified :*

- $Prim(t) = [t]$
- $[t] \sqsubseteq C'$

The first property expresses the fact that there exists a unique relativised zone associated with a synchronized event t, the zone $Relativise([t])$. In this case, we denote this zone as $Relativise(t)$. The second property comes from Lemma <>.

Proof. □

3.3.8 Definition (Adequate Order $\lhd$)

A partial order $\lhd$ on the finite configurations of the unfolding of an NTA is an adequate order iff it verifies the following properties:

- $\lhd$ is well founded,
- $C_1 \sqsubseteq C_2$ implies $C_1 \lhd C_2$
- $\lhd$ is preserved by finite extensions : if $C_1 \lhd C_2$ and $\lambda(Cut(C_1)) = \lambda(Cut(C_2))$, then for all finite extensions $C_1 \oplus E$ of C_1, there exists an isomorphic extension $C_2 \oplus E'$ of C_2, and vice versa.

Example The following are 2 adequate orders (can be verified) :

- $C_1 \lhd C_2 \iff C_1 \sqsubseteq C_2$
- $C_1 \lhd C_2 \iff |C_1| < |C_2|$

3.3.9 Definition (Redundant Synchronized Event (RSE))

Let $\mathcal{A}$ be an NTA and S be a set of unavoidable edges. Let K denote the maximal constant that appears in the NTA $\mathcal{A}$. Consider the unfolding $Unf - T(A_{Sync-S})$ of the NTA $\mathcal{A}_{\mathcal{S}\dagger\backslash\rfloor-\mathcal{S}}$, and a partial order $\lhd$. A synchronized event t is called redundant if there exists another synchronized event t' in $Unf - T(A_{Sync-S})$ verifying the following conditions:

- $t' \lhd t$
- $\lambda(Cut(t')) = \lambda(Cut(t))$
- $Approx_K(Relativise(t)) \subseteq Approx_K(Relativise(t'))$

We say that t is redundant vis-a-vis t'.

3.3.10 Definition (K-equivalence ($\approx_K$))

Given 2 valuations v, v' defined on the set of clocks, X, we define the relation $\approx_K$ by :

$v \approx_K v' \iff \forall x \in X,\ v(x) = v'(x)$ if $|v(x)| \leq K$ and $|v'(x)| > K$ otherwise.

□

Lemma 3.12. *Let (β, d) be a feasible TNBP such that $[\beta]$ contains a synchronised event t which is redundant wrt t'. The second point of Lemma <> implies that $[t] \sqsubseteq [\beta]$ We denote this extension by $[\beta] = [t] \oplus Y$. Then, there exists feasible TNBP (β', d') verifying the following properties :*

- $[\beta'] = [t'] \oplus Y'$
- *Y', Y are isomorphic via λ*
- $Conf(\beta', v') \approx_K Conf(\beta, v)$

Proof. □

3.3.11 Example - Incorrect Usage of Extrapolation

□

Algorithm 6 Building a Finite, Complete Prefix of the Timed Unfolding

Require: An NTA $\mathcal{A}$

Ensure: A finite and complete prefix Fin of $T - Unf(A)$.

```
1: FCP := {(l_{1,0}, φ), ..., (l_{n,0}, φ)} ∪ {(x, φ)|x ∈ X};
2: pe := PE(T − Unf);
3: while pe = φ do
4:     Choose an event t = (ē, Y_in, Y_r) in pe.
5:     Compute the zone W^loc_[t] by Algorithm 2.
6:     if W^loc_[t] = φ then
7:         Mark t as useless event.
8:     else
9:         if t is not a synchronized event then
10:            Extend(FCP, t);
11:            pe := PE(T − Unf);
12:        else
13:            if t is redundant then
14:                Mark t as useless event.
15:            else
16:                Extend(FCP, t);
17:                pe := PE(T − Unf);
18:            end if
19:        end if
20:    end if
21: end while
22: return FCP
```

3.3.12 Algorithm (Timed Unfolding - Finite Prefix)

Theorem 3.13. *The Algorithm <> terminates and the finite prefix calculated $FCP(\mathcal{A})$ is complete upto K-approximatin, ie, if a configuration $(\bar{l}, v)$ is accessible in $\mathcal{A}$, then there exists a configuration $(\bar{l}, v')$ represented in this prefix which satisfies $v \approx_K v'$.*

Proof. **Termination**
Completeness

□