Forward Reachability Analysis of Timed Petri Nets

Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén

Uppsala University, Sweden
{parosh,johannd,pritha,aletta}@it.uu.se

Abstract. We consider verification of safety properties for concurrent real-timed systems modelled as timed Petri nets, by performing symbolic forward reachability analysis. We introduce a formalism, called *region generators* for representing sets of markings of timed Petri nets. Region generators characterize downward closed sets of regions, and provide exact abstractions of sets of reachable states with respect to safety properties. We show that the standard operations needed for performing symbolic reachability analysis are computable for region generators. Since forward reachability analysis is necessarily incomplete, we introduce an acceleration technique to make the procedure terminate more often on practical examples. We have implemented a prototype for analyzing timed Petri nets and used it to verify a parameterized version of Fischer's protocol and a producer-consumer protocol. We also used the tool to extract finite-state abstractions of these protocols.

1 Introduction

Timed Petri nets (TPNs) are extensions of Petri nets in the sense that each token has an "age" which is represented by a real valued clock (see [Bow96] for a survey). TPNs are computationally more powerful than timed automata [AD90], since they operate on a potentially unbounded number of clocks. This implies that TPNs can, among other things, model parameterized timed systems (systems consisting of an unbounded number of timed processes) [AN01].

A fundamental problem for TPNs (and also for standard Petri nets) is that of *coverability*: check whether an upward closed set of *final markings* is reachable from a set of initial markings. Using standard techniques [VW86], several classes of safety properties for TPNs can be reduced to the coverability problem where final markings represent violations of the safety property. To solve coverability, one may either compute the set of *forward reachable markings*, i.e., all the markings reachable from the initial markings; or compute *backward reachable markings*, i.e., all the markings from which a final marking is reachable.

While backward and forward analysis seem to be symmetric, they exhibit surprisingly different behaviours in many applications. For TPNs, even though the set of backward reachable states is computable [AN01], the set of forward reachable states is in general not computable. Therefore any procedure for performing forward reachability analysis on TPNs is necessarily incomplete. However, forward analysis is practically very attractive. The set of forward reachable states contains much more information about system behaviour than backward reachable states. This is due to the fact that forward closure characterizes the set of states which arises during the execution of the system, in contrast to backward closure which only describes the states from which the system may fail. This implies for instance that forward analysis can often be used for constructing a symbolic graph which is a finite-state abstraction of the system, and which is a simulation or a bisimulation of the original system (see e.g. [BLO98,LBBO01]).

Contribution: We consider performing forward reachability analysis for TPNs. We provide an abstraction of the set of reachable markings by taking its *downward closure*. The abstraction is exact with respect to coverability (consequently with respect to safety properties), i.e, a given TPN satisfies any safety property exactly when the downward closure of the set of reachable states satisfies the same property. Moreover, the downward closure has usually a simpler structure than the exact set of reachable states.

The set of reachable markings (and its downward closure) is in general infinite. So, we introduce a symbolic representation for downward closed sets, which we call *region generators*. Each region generator denotes the union of an infinite number of *regions* [AD90]. Regions are designed for timed automata (which operate on a finite number of clocks), and are therefore not sufficiently powerful to capture the behaviour of TPNs. We define region generators hierarchically as languages where each word in the language is a sequence of multisets over an alphabet. The idea is that elements belonging to the same multiset correspond to clocks with equal fractional parts while the ordering among multisets in a word corresponds to increasing fractional parts of the clock values.

We show that region generators allow the basic operations in forward analysis, i.e, checking membership, entailment, and computing the post-images with respect to a single transition. Since forward analysis is incomplete, we also give an acceleration scheme to make the analysis terminate more often. The scheme computes, in one step, the effect of an arbitrary number of firings of a single discrete transition interleaved with timed transitions.

We have implemented the forward reachability procedure and used the tool to compute the reachability set for a parameterized version of Fischer's protocol and for a simple producer/consumer protocol. Also, we used the tool for generating finite state abstractions of these protocols.

Related Work: [ABJ98] considers *simple regular expressions (SRE)* as representations for downward closed languages over a *finite* alphabet. SREs are used for performing forward reachability analysis of lossy channel systems. SREs are not sufficiently powerful in the context of TPNs, since they are defined on a finite alphabet, while in the case of region generators the underlying alphabet is infinite (the set of multisets over a finite alphabet).

Both [DR00] and [FRSB02] consider (untimed) Petri nets and give symbolic representations for upward closed sets and downward closed sets of markings, respectively. The works in [FIS00,BG96,BH97] give symbolic representation for FIFO automata. These representations are designed for weaker models (Petri nets and FIFO automata) and cannot model the behaviour of TPNs.

[AN01] considers timed Petri nets. The symbolic representation in this paper characterizes upward closed sets of markings, and can be used for backward analysis, but not for forward analysis.

Outline: In the next section, we introduce timed Petri nets and define the coverability problem for TPNS. In Section 3, we introduce region generators. Section 4 gives the forward reachability algorithm. Section 5 and Section 6 give algorithms for computing post-images and acceleration respectively. In Section 7 we report on some experiments with our implementation. Finally, we give conclusions and directions for future research in Section 8.

2 Definitions

We consider *Timed Petri Nets* (*TPN*s) where each token is equipped with a real-valued clock representing the "age" of the token. The firing conditions of a transition include the usual ones for Petri nets. Additionally, each arc between a place and a transition is labelled with an interval of natural numbers. When firing a transition, tokens which are removed (added) from (to) places should have ages in the intervals of corresponding arcs.

We use $\mathbb{N}, \mathbb{R}^{\geq 0}$ to denote the sets of natural numbers, nonnegative reals respectively. We use a set *Intrv* of intervals. An open interval is written as (w,z) where $w \in \mathbb{N}$ and $z \in \mathbb{N} \cup \{\square\}$. Intervals can also be closed in one or both directions, e.g. $[w,z)$ is closed to the left and open to the right.

For a set A, we use A^* and $A^{\circledast}$ to denote the set of finite words and finite multisets over A respectively. We view a multiset over A as a mapping from A to $\mathbb{N}$. Sometimes, we write multisets as lists, so $[2.4\ ,\ 5.1\ ,\ 5.1\ ,\ 2.4\ ,\ 2.4]$ represents a multiset B over $\mathbb{R}^{\geq 0}$ where $B(2.4) = 3$, $B(5.1) = 2$ and $B(x) = 0$ for $x \neq 2.4, 5.1$. We may also write B as $\left[2.4^3\ ,\ 5.1^2\right]$. For multisets B_1 and B_2 over $\mathbb{N}$, we say that $B_1 \leq^m B_2$ if $B_1(a) \leq B_2(a)$ for each $a \in A$. We define addition $B_1 + B_2$ of multisets B_1, B_2 to be the multiset B where $B(a) = B_1(a) + B_2(a)$, and (assuming $B_1 \leq^m B_2$) we define the subtraction $B_2 - B_1$ to be the multiset B where $B(a) = B_2(a) - B_1(a)$, for each $a \in A$. We use $\square$ to denote both the empty multiset and the empty word.

Timed Petri Nets A *Timed Petri Net (TPN)* is a tuple $N = (P, T, In, Out)$ where P is a finite set of places, T is a finite set of transitions and In, Out are partial functions from $T \times P$ to *Intrv*.

If $In(t,p)$ ($Out(t,p)$) is defined, we say that p is an *input (output) place* of t. A *marking* M of N is a multiset over $P \times \mathbb{R}^{\geq 0}$. We abuse notations and write[1] $p(x)$ instead of (p,x). The marking M defines the numbers and ages of tokens in each place in the net. That is, $M(p(x))$ defines the number of tokens with age x in place p. Notice that untimed Petri nets are a special case in our model where all intervals are of the form $[0, \square)$.

For a marking M of the form $[p_1(x_1)\ ,\ \ldots\ ,\ p_n(x_n)]$ and $x \in \mathbb{R}^{\geq 0}$, we use M^{+x} to denote the marking $[p_1(x_1 + x)\ ,\ \ldots\ ,\ p_n(x_n + x)]$.

Transitions: There are two types of transitions : *timed* and *discrete* transitions. A *timed transition* increases the age of each token by the same real number. Formally, for $x \in \mathbb{R}^{\geq 0}$, $M_1 \longrightarrow_x M_2$ if $M_2 = M_1^{+x}$. We use $M_1 \longrightarrow_{Time} M_2$ to denote that $M_1 \longrightarrow_x M_2$ for some $x \in \mathbb{R}^{\geq 0}$.

We define the set of *discrete transitions* $\longrightarrow_{Disc}$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of firing the transition t. More precisely, $M_1 \longrightarrow_t M_2$ if the set of input arcs $\{p(I) |\ In(t,p) = I\}$ is of the form $\{p_1(I_1), \ldots, p_k(I_k)\}$, the set of output arcs $\{p(I) |\ Out(t,p) = I\}$ is of the form $\{q_1(J_1), \ldots, q_\ell(J_\ell)\}$, and there are multisets $B_1 = [p_1(x_1)\ , \ldots\ , p_k(x_k)]$ and $B_2 = [q_1(y_1)\ , \ldots\ , q_\ell(y_\ell)]$ such that the following holds:
- $B_1 \leq^m M_1$
- $x_i \in I_i$, for $i : 1 \leq i \leq k$.
- $y_i \in J_i$, for $i : 1 \leq i \leq \ell$.
- $M_2 = (M_1 - B_1) + B_2$.

Intuitively, a transition t may be fired only if for each incoming arc to the transition, there

[1] Later, we shall use a similar notation. For instance, we write $p(n)$ instead of (p,n) where $n \in \mathbb{N}$, and write $p(I)$ instead of (p,I) where $I \in Intrv$.

is a token with the "right" age in the corresponding input place. These tokens will be removed when the transition is fired. The newly produced tokens have ages belonging to the relevant intervals.

We define $\longrightarrow = \longrightarrow_{Time} \cup \longrightarrow_{Disc}$ and use $\xrightarrow{*}$ to denote the reflexive transitive closure of $\longrightarrow$. We say that M_2 is *reachable* from M_1 if $M_1 \xrightarrow{*} M_2$. We define $Reach(M)$ to be the set $\left\{M' \mid M \xrightarrow{*} M'\right\}$.

Remark: Notice that we assume a lazy (non-urgent) behaviour of TPNS. This means that we may choose to "let time pass" instead of firing enabled transitions, even if that disables a transition by making some of the needed tokens "too old".

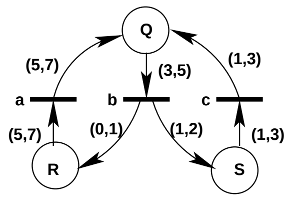

Fig. 1. A small timed Petri net.

Example 1. Figure 1 shows an example of a TPN where $P = \{Q,R,S\}$ and $T = \{a,b,c\}$. For instance, $In(b,Q) = (3,5)$ and $Out(b,R) = (0,1)$ and $Out(b,S) = (1,2)$.

Regions We define an ordering on markings which extends the equivalence relation on markings induced by the classical region graph construction of [AD90]. Let *max* be the maximum natural number which appears (in the intervals) on the arcs of the TPN. A *region* defines the integral parts of clock values up to *max* (the exact age of a token is irrelevant if it is greater than *max*), and also the ordering of the fractional parts among clock values. For TPNs, we need to use a variant which also defines the place in which each token (clock) resides. Following Godskesen [God94] we represent a region by a triple (B_0, w, B_{max}) where

– $B_0 \in (P \times \{0,\ldots,max\})^{\circledast}$. B_0 is a multiset of pairs. A pair of the form $p(n)$ represents a token with age exactly n in place p.
– $w \in \left((P \times \{0,\ldots,max-1\})^{\circledast}\right)^*$. w is a word over the set $(P \times \{0,\ldots,max-1\})^{\circledast}$, i.e., w is a word where each element in the word is a multiset over $P \times \{0,\ldots,max-1\}$. The pair $p(n)$ represents a token in place p with age x such that $x \in (n, n+1)$. Pairs in the same multiset represent tokens whose ages have equal fractional parts. The order of the multisets in w corresponds to the order of the fractional parts.
– $B_{max} \in P^{\circledast}$. B_{max} is a multiset over P representing tokens with ages strictly greater than *max*. Since the actual ages of these tokens are irrelevant, the information about their ages is omitted in the representation.

Formally, each region characterizes an infinite set of markings as follows. Assume a marking $M = [p_1(x_1), \ldots, p_n(x_n)]$ and a region $r = (B_0, B_1 B_2 \cdots B_m, B_{m+1})$. Let B_j be of the form $\left[q_{j1}(y_{j1}), \ldots, q_{j\ell_j}(y_{j\ell_j})\right]$ for $j : 0 \le j \le m$ and B_{m+1} is of the form $\left[q_{m+1\,1}, \ldots, q_{m+1\,l_{m+1}}\right]$. We say that M *satisfies* r, written $M \models r$, if there is a bijection

h from the set $\{1,\ldots,n\}$ to the set of pairs $\{(j,k) \mid (0 \le j \le m+1) \wedge (1 \le k \le \ell_j)\}$ such that the following conditions are satisfied:

- $p_i = q_{h(i)}$. Each token should have the same place as that required by the corresponding element in r.
- If $h(i) = (j,k)$ then $j = m+1$ iff $x_i > max$. Tokens older than max should correspond to elements in multiset B_{m+1}. The actual ages of these tokens are not relevant.
- If $x_i \le max$ and $h(i) = (j,k)$ then $\lfloor x_i \rfloor = y_{j\,k}$. The integral part of the age of tokens should agree with the natural number specified by the corresponding elements in w.
- If $x_i \le max$ and $h(i) = (j,k)$ then $fract(x_i) = 0$ iff $j = 0$. Tokens with zero fractional parts correspond to elements in multiset B_0.
- If $x_{i_1}, x_{i_2} < max$, $h(i_1) = (j_1,k_1)$ and $h(i_2) = (j_2,k_2)$ then $fract(x_{i_1}) \le fract(x_{i_2})$ iff $j_1 \le j_2$. Tokens with equal fractional parts correspond to elements in the same multiset (unless they belong to B_{m+1}). The ordering among multisets inside r reflects the ordering among fractional parts in clock values.

We let $[\![r]\!] = \{M \mid M \models r\}$.

The region construction defines an equivalence relation $\equiv$ on the set of markings such that $M_1 \equiv M_2$ if, for each region r, it is the case that $M_1 \in [\![r]\!]$ iff $M_2 \in [\![r]\!]$. It is well-known [AD90] that $\equiv$ is a congruence on the set of markings, i.e, if $M_1 \longrightarrow M_2$ and $M_1 \equiv M_3$ then there is an M_4 such that $M_2 \equiv M_4$ and $M_3 \longrightarrow M_4$.

Ordering Now we define an ordering $\preceq$ on the set of markings such that $M_1 \preceq M_2$ if there is an M_2' with $M_1 \equiv M_2'$ and $M_2' \le^m M_2$. In other words, $M_1 \preceq M_2$ if we can delete a number of tokens from M_2 and as a result obtain a new marking which is equivalent to M_1. We let $M_1 \prec M_2$ denote that $M_1 \preceq M_2$ and $M_1 \not\equiv M_2$.

A set M of markings is said to be *upward closed* if $M_1 \in \mathsf{M}$ and $M_1 \preceq M_2$ implies $M_2 \in \mathsf{M}$. We define the *upward closure* $\mathsf{M}\uparrow$ to be the set $\{M \mid \exists M' \in \mathsf{M} :\ M' \preceq M\}$. Downward closed sets and downward closure $M\downarrow$ of a set M are defined in a similar manner.

Next, we consider the following lemma which states that $\longrightarrow$ is *monotonic* with respect to the ordering $\preceq$.

Lemma 1. *If $M_1 \longrightarrow M_2$ and $M_1 \preceq M_3$ then there is an M_4 such that $M_2 \preceq M_4$ and $M_3 \longrightarrow M_4$.*

Example 2. Consider the TPN N in Figure 1 where $max = 7$. Here is an example of a region $r = ([R(2)],\ [S(5)]\ \ [R(1), S(5), S(2)],\ [Q])$. Markings $M_1 = [R(2.0)\ ,S(5.5), R(1.7), S(5.7), S(2.7), Q(8.9)]$ and $M_2 = [R(2.0)\ ,S(5.7), R(1.8), S(5.8), S(2.8), Q(9.9)]$ of N satisfy the above region. Notice that $M_1 \equiv M_2$. Let $M_3 = M_2 + [R(1.2),\ Q(14.2)]$. Since $M_2 \le^m M_3$ and $M_1 \equiv M_2$, we have $M_1 \preceq M_3$.

Coverability Problem for TPNs:

Instance: A set of initial markings M_{init} and a finite set M_{fin} of final markings.
Question: $Reach(\mathsf{M}_{init}) \cap (\mathsf{M}_{fin}\uparrow) = \emptyset$?

The coverability problem is interesting from the verification point of view, since checking safety properties (e.g, mutual exclusion properties) can almost always be reduced to coverability[VW86]. We use the set $\mathsf{M}_{fin}\uparrow$ to represent a set of "bad markings" which we do not want to occur during the execution of the system. Safety is then equivalent to non-reachability of $\mathsf{M}_{fin}\uparrow$.

In fact, from Lemma 1 it follows immediately that analyzing coverability will not be affected by taking the downward closure of the set of reachable markings.

Lemma 2. *For a set of markings* M_{init} *and an upward closed set* M *of markings, we have* $Reach(\mathsf{M}_{init}) \cap \mathsf{M} = \emptyset$ *iff* $(Reach(\mathsf{M}_{init}))\downarrow \cap \mathsf{M} = \emptyset$.

Since $\mathsf{M}_{fin}\uparrow$ (in the definition of the coverability problem) is upward closed by definition, it follows from Lemma 2 that taking downward closure of $Reach(M_{init})$ gives an exact abstraction with respect to coverability.

Infeasibility of the Karp-Miller Algorithm The Karp-Miller algorithm [KM69] is the classical method used for checking coverability in untimed Petri nets. However, it is not obvious how to extend the algorithm to TPNs. [KM69] constructs a reachability tree starting from an initial marking. It detects paths in the reachability tree which leads from a marking M_1 to a larger marking M_2. In such a case, it makes an over-approximation of the set of reachable markings by putting □ (interpreted as "unboundedly many tokens") in each place p with $M_1(p) < M_2(p)$. This over-approximation preserves safety properties.

In the case of TPNs, if $M_1 \prec M_2$ (in fact even if $M_1 < M_2$) the only conclusion we can draw is that we will generate unboundedly many tokens with ages greater than *max*. Even if all such tokens are abstracted by □, an unbounded number of tokens with ages less than *max* may still appear in the analysis. Termination is therefore not guaranteed.

3 Region Generators

In this section we introduce *region generators* which we define in a hierarchical manner. First, we introduce *multiset* and *word language generators* and then describe how a region generator characterizes a potentially infinite set (language) of regions.

Mlgs We define *multiset language generator (mlgs)*, each of which characterizes a language which consists of multisets over a finite alphabet.

Let A be a finite alphabet. A *multiset language* (over A) is a subset of $A^{\circledast}$. We will consider multiset languages which are downward closed with respect to the relation $\leq^m$ on multisets. If L is downward closed then $B_1 \in L$ and $B_2 \leq^m B_1$ implies $B_2 \in L$. Recall that $\leq^m$ is the ordering defined on multisets (Section 2).

We define *(downward-closed) multiset language generators* (or *mlgs* for short) over the finite alphabet A. Each mlg □ over A defines a multiset language over A, denoted L (□), which is downward closed. The set of mlgs over A and their languages are defined as follows

– An *expression* over A is of one of the following two forms:
  - an *atomic expression* a where $a \in A$. $L(a) = \{[a]\ ,\ □\}$.
  - a *star expression* of the form $S^{\circledast}$ where $S \subseteq A$. $L(S^{\circledast}) = \{[a_1\ ,\ \ldots\ ,\ a_m] \mid m \geq 0 \wedge a_1, \ldots, a_m \in S\}$.

– An *mlg* □ is a (possibly empty) sequence $e_1 + \cdots + e_\ell$ of expressions. L (□) $= \{B_1 + \cdots + B_\ell \mid B_1 \in L(e_1), \cdots, B_\ell \in L(e_\ell)\}$. We denote an empty mlg by □ and assume that L (□) $=$ {□}.

We also consider sets of mlgs which we interpret as unions. If $\square = \{\square_1, \cdots, \square_m\}$ is a set of mlgs, then $L(\square) = L(\square_1) \cup \cdots \cup L(\square_m)$. We assume $L(\emptyset) = \emptyset$.

Sometimes we identify mlgs with the languages they represent, so we write $\square_1 \subseteq \square_2$ (rather than $L(\square_1) \subseteq L(\square_2)$), and $B \in \square$ (rather than $B \in L(\square)$), etc.

An mlg $\square$ is said to be in *normal form* if it is of the form $e + e_1 + \cdots + e_k$ where e is a star expression and $e_1, \ldots, e_k$ are atomic expressions and for each $i : 1 \leq i \leq k$, $e_i \not\subseteq e$. A set of mlgs $\{\square_1, \cdots, \square_m\}$ is said to be *normal* if $\square_i$ is in *normal form* for each $i : 1 \leq i \leq m$, and $\square_i \not\subseteq \square_j$ for each $i, j : 1 \leq i \neq j \leq m$.

Wlgs We consider languages where each word is a sequence of multisets over a finite alphabet A, i.e., each word is a member of $(A^{\circledast})^*$. The language is then a subset of $(A^{\circledast})^*$. Notice that the underlying alphabet, namely $A^{\circledast}$ is infinite.

For a word $w \in L$, we use $|w|$ to denote the length of w, and $w(i)$ to denote the i^{th} element of w where $1 \leq i \leq |w|$. We observe that $w(i)$ is a multiset over A. We use $w_1 \bullet w_2$ to denote the concatenation of the words w_1 and w_2.

We define the ordering $\leq^w$ on set of words such that $w_1 \leq^w w_2$ if there is a strictly monotonic injection $h : \{1, \ldots, |w_1|\} \rightarrow \{1, \ldots, |w_2|\}$ where $w_1(i) \leq^m w_2(h(i))$ for $i : 1 \leq i \leq |w_1|$.

We shall consider languages which are downward closed with respect to $\leq^w$. In a similar manner to mlgs, we define downward closed *word language generators (wlgs)* and (word) languages as follows.

- A *word expression* over A is of one of the following two forms:
  - a *word atomic expression* is an mlg $\square$ over A. $L(\square) = \{B \mid B \in \square\} \cup \{\square\}$.
  - a *word star expression* of the form $\{\square_1, \cdots, \square_k\}^*$, where $\square_1, \ldots, \square_k$ are mlgs over A.
    $L(\{\square_1, \cdots, \square_k\}^*) = \{B_1 \bullet \cdots \bullet B_m \mid (m \geq 0) \text{ and } B_1, \ldots, B_m \in L(\square_1) \cup \cdots L(\square_k)\}$.
- A *word language generator (wlg)* $\square$ over A is a (possibly empty) concatenation $e_1 \bullet \cdots \bullet e_\ell$ of word expressions $e_1, \ldots, e_\ell$. $L(\square) = \{w_1 \bullet \cdots \bullet w_\ell \mid w_1 \in L(e_1) \wedge \cdots \wedge w_\ell \in L(e_\ell)\}$.

Notice that the concatenation operator is associative, but not commutative (as is the operator $+$ for multisets). Again, we denote the empty wlg by $\square$. For a set $\square = \{\square_1, \cdots, \square_m\}$ of wlgs, we define $L(\square) = L(\square_1) \cup \cdots \cup L(\square_m)$.

A word atomic expression e of the form $\square$ is said to be in normal form if $\square$ is a normal mlg. A word star expression $\{\square_1, \ldots, \square_k\}^*$ is said be in normal form if the set of mlgs $\{\square_1, \ldots, \square_k\}$ is in normal form.

A wlg $\square = e_1 \bullet \cdots \bullet e_\ell$ is said to be *normal* if (a) $e_1, \ldots, e_\ell$ are normal, (b) $e_i \bullet e_{i+1} \not\subseteq e_i$ and (c) $e_i \bullet e_{i+1} \not\subseteq e_{i+1}$, for each $i : 1 \leq i < \ell$. A set of wlgs $\{\square_1, \cdots, \square_m\}$ is said to be *normal* if $\square_1, \ldots, \square_m$ are normal and $\square_i \not\subseteq \square_j$ for each $i, j : 1 \leq i \neq j \leq m$.

Lemma 3. *For each set of wlgs (mlgs) $\square$, there is a unique normal set of wlgs (mlgs) $\square'$ such that $L(\square) = L(\square')$.*

From now on, we assume always that (sets of) mlgs and wlgs are in normal form.

Entailment Given two (sets of) mlgs $\square_1, \square_2$, we say that $\square_1$ entails $\square_2$ to denote that $\square_1 \subseteq \square_2$. We define entailment for (sets of) wlgs in a similar manner. The following theorem describes complexity of entailment on mlgs and wlgs.

Theorem 1. *Entailment of sets of mlgs and wlgs can be computed in quadratic time and cubic time, respectively.*

The next theorem relates mlgs and wlgs to downward closed languages.

Theorem 2. *For each downward closed word (multiset) language L, there is a finite set $\square$ of wlgs (mlgs) such that $L = L(\square)$.*

Region Generators A *region generator* $\square$ is a triple $(\square_0, \square, \square_{\max})$ where $\square_0$ is an mlg over $P \times \{0, \ldots, max\}$, $\square$ is a wlg over $P \times \{0, \ldots, max - 1\}$, and $\square_{\max}$ is an mlg over P. The language $L(\square)$ contains exactly each region of the form (B_0, w, B_{max}) where $B_0 \in \square_0$, $w \in \square$, and $B_{max} \in \square_{\max}$. We observe that if $\square_1 = \left(\square_0^1, \square^1, \square_{\max}^1\right)$ and $\square_2 = \left(\square_0^2, \square^2, \square_{\max}^2\right)$ then $\square_1 \subseteq \square_2$ iff $\square_0^1 \subseteq \square_0^2$, $\square^1 \subseteq \square^2$, and $\square_{\max}^1 \subseteq \square_{\max}^2$. In other words entailment between region generators can be computed by checking entailment between the individual elements.

For a region generator $\square$, we define $[\![\square]\!]$ to be $\cup_{r \in L(\square)} [\![r]\!]$. In other words, a region generator $\square$: (a) defines a language $L(\square)$ of regions; and (b) denotes a set of markings, namely all markings which belong to the denotation $[\![r]\!]$ for some region $r \in L(\square)$. A finite set $\square = \{\square_1, \ldots, \square_m\}$ of region generators denotes the union of its elements, i.e, $[\![\square]\!] = \bigcup_{1 \leq i \leq m} [\![\square_i]\!]$.

Given a marking M and a region generator $\square$, it is straightforward to compute whether $M \in [\![\square]\!]$ from the definition of $[\![r]\!]$ and $[\![\square]\!]$.

We define an ordering $\leq^r$ on regions such that if $r_1 = \left(B_0^1, w^1, B_{\max}^1\right)$ and $r_2 = \left(B_0^2, w^2, B_{\max}^2\right)$ then $r_1 \leq^r r_2$ iff $B_0^1 \leq^m B_0^2$, $w^1 \leq^w w^2$, and $B_{\max}^1 \leq^m B_{\max}^2$.

By Theorem 2 it follows that for each set R of regions which is downward closed with respect to $\leq^r$, there is a finite set of region generators $\square$ such that $L(\square) = R$. Also, by definition, it follows that if $r_1 \leq^r r_2$ then $M_1 \preceq M_2$ for each $M_1 \in [\![r_1]\!]$ and $M_2 \in [\![r_2]\!]$.

From this we get the following

Theorem 3. *For each set* M *of markings which is downward closed with respect to* $\preceq$ *there is a finite set of region generators* $\square$ *such that* $\mathsf{M} = [\![\square]\!]$.

Example 3. Consider again the TPN in Figure 1 with $max = 7$. Examples of mlgs over $\{Q, R, S\} \times \{0, \ldots, 7\}$ are $R(2)$, $\{S(5), R(1), S(2)\}^{\circledast}$, $Q(3)$, etc. $\left\{\{S(5), R(1), S(2)\}^{\circledast}\right\}^* \bullet Q(3)$ is an example of a wlg over $\{Q, R, S\} \times \{0, \ldots, 7\}$ and $Q + R$ is an mlg over $\{Q, R, S\}$. Finally, an example of region generator is given by $\square = (R(2), \left\{\{S(5), R(1), S(2)\}^{\circledast}\right\}^* \bullet Q(3)\ ,\ Q + R)$. Notice that the region r, given in Example 2, is in $L(\square)$.

4 Forward Analysis

We present a version of the standard symbolic forward reachability algorithm which uses region generators as a symbolic representation. The algorithm inputs a set of region generators $\square_{init}$ characterizing the set M_{init} of initial markings, and a set M_{fin} of final markings and tries to answer whether $[\![\square_{init}]\!] \cap \mathsf{M}_{fin} \uparrow = \emptyset$. The algorithm computes the sequence $\square_0, \square_1, \ldots$ of sets of region generators s.t $\square_{i+1} = \square_i \cup succ(\square_i)$ with $\square_0 = \square_{init}$. If $[\![\square_i]\!] \cap \mathsf{M}_{fin} \uparrow \neq \emptyset$ (amounts to checking membership of elements of M_{fin} in $[\![\square]\!]$), or if $\square_{i+1} = \square_i$, then the procedure is terminated. We define $succ(\square)$ to be $Post_{Time}(\square) \cup \bigcup_{t \in T} (Post_t(\square) \cup Step_t(\square))$. $Post_{Time}$ and $Post_t$, defined in Section 5, compute the effect of timed and discrete transitions respectively. $Step_t$, defined in Section 6, implements acceleration. Also, whenever there are two region generators $\square_1, \square_2$ in a set of region generators such that $\square_1 \subseteq \square_2$, we remove $\square_1$ from the set.

Even if we know by Theorem 3 that there is finite set $\square$ of region generators such that $Reach([\![\square_{init}]\!]) = [\![\square]\!]$, it is straightforward to apply the techniques presented in [May00] to show the following.

Theorem 4. *Given a region generator $\square_{init}$ we cannot in general compute a set $\square$ of region generators such that $Reach([\![\square_{init}]\!]) = [\![\square]\!]$.*

The aim of acceleration is to make the forward analysis procedure terminate more often.

5 Computing *Post*

In this section, we consider the post-image of a region generator $\square$ with respect to timed and discrete transitions respectively.

Post$_{Time}$

For an input region generator $\square$, we shall characterize the set of all markings which can be reached from a marking in $[\![\square]\!]$ through the passage of the time. We shall compute $Post_{Time}(\square)$ as a finite set of region generators such that $[\![Post_{Time}(\square)]\!] = \{M' \mid \exists M \in [\![\square]\!].\, M \longrightarrow_{Time} M'\}$.

First, we analyze informally the manner in which a marking changes through passage of time. Then, we translate this analysis into a number of computation rules on region generators. Consider a marking M and a region $r = (B_0, w, B_{\max})$ such that $M \models r$. Three cases are possible:

1. If $B_0 = \square$, i.e., there are no tokens in M with ages whose fractional parts are equal to zero. Let w be of the form $w_1 \bullet B_1$. The behaviour of the TPN from M due to passage of time is decided by a certain subinterval of $\mathbb{R}^{\geq 0}$ which we denote by $stable(M)$. This interval is defined by $[0 : 1-x)$ where x is the highest fractional part among the tokens whose ages are less than *max*. Those tokens correspond to B_1 in the definition of r. We call $stable(M)$ the *stable period* of M.

Suppose that time passes by an amount $\square \in stable(M)$. If $M \longrightarrow_{\square} M_1$ then $M_1 \models r$, i.e., $M_1 \equiv M$. In other words, if the elapsed time is in the stable period of M then all markings reached through performing timed transitions are equivalent to M. The reason is that, although the fractional parts have increased (by the same amount), the relative ordering of the fractional parts, and the integral parts of the ages are not affected.

As soon as we leave the stable period, the tokens which originally had the highest fractional parts (those corresponding to B_1) will now change: their integral parts will increase by one while fractional parts will become equal to zero. Therefore, we reach a marking M_2, where $M_2 \models r_2$ and r_2 is of the form $\left(B_1^{+1}, w_1, B_{\max}\right)$. Here, B_1^{+1} is the result of replacing each pair $p(n)$ in B_1 by $p(n+1)$.

2. If $B_0 \neq \square$, i.e., there are some tokens whose ages do not exceed *max* and whose fractional parts are equal to zero. We divide the tokens in B_0 into two multisets: *young tokens* whose ages are strictly less than *max*, and *old tokens* whose ages are equal to *max*. The stable period $stable(M)$ here is the point interval $[0:0]$. Suppose that we let time pass by an amount $\square : 0 < \square < 1-x$, where x is the highest fractional part of the tokens whose ages are less than *max*. Then the fractional parts for the tokens in B_0 will become positive.

The young tokens will still have values not exceeding *max*, while the old tokens will now have values strictly greater than *max*. This means that if $M \longrightarrow_{\square} M_1$ then $M_1 \models r_1$ where r_1 is of the form $(\square, young \bullet w, B_{max} + old)$. Here, *young* and *old* are sub-multisets of B_0 such that $young(p(n)) = B_0(p(n))$ if $n < max$, and $old(p) = B_0(p(max))$. Since the fractional parts of the tokens in *young* are smaller than all other tokens, we put *young* first in the second component of the region. Also, the ages of the tokens in *old* are now strictly greater than *max*, so they are added to the third component of the region.

3. If $B_0 = \square, w = \square$, all tokens have age greater than *max*. Now, if we let time pass by any amount $\square \geq 0$ and $M \longrightarrow_{\square} M_1$, then $M_1 \models r$. When all tokens reach age of *max*, aging of tokens becomes irrelevant.

Notice that in cases 1 and 2, the stable period is the largest interval during which the marking does not change the region it belongs to. Markings in case 3 never change their regions and are therefore considered to be "stable forever" with respect to timed transitions. Also, we observe that each of first two cases above corresponds to "rotating" the multisets in B_0 and w, sometimes also moving them to $B_{\max}$.

Now, we formalize the analysis above as a number of computation rules on region generators. First, we introduce some notations. Let $\square$ be an mlg of the form $\{a_1, \ldots, a_k\}^{\circledast} + a_{k+1} + \cdots + a_{k+\ell}$. Notice that, by the normal form defined in Section 3, we can always write $\square$ in this form. We define $\#\square$ to be the pair (B, B') where $B = [a_1, \ldots, a_k]$ and $B' = [a_{k+1}, \ldots, a_{k+\ell}]$.

Let $\square$ be an mlg over $P \times \{0, \ldots, max\}$ with $\#\square = (B, B')$. We define $young(\square)$ and $old(\square)$ to be mlgs over $P \times \{0, \ldots, max-1\}$ and P respectively such that the following holds: let $\#young(\square) = (B_1, B_1')$ and $\#old(\square) = (B_2, B_2')$ such that

– $B(p(n)) = B_1(p(n))$ and $B'(p(n)) = B_1'(p(n))$ if $n < max$.
– $B(p(max)) = B_2(p)$ and $B'(p(max)) = B_2'(p)$.

In other words, from $\square$, we obtain an mlg given by $young(\square)$ which characterizes tokens younger than *max* and an mlg $old(\square)$ which characterizes tokens older than *max*.

Let $\square$ be an mlg over $P \times \{0, \ldots, max-1\}$ of the form $\{p_1(n_1), \ldots p_k(n_k)\}^{\circledast} + p_{k+1}(n_{k+1}) + \cdots + p_{k+\ell}(n_{k+\ell})$. We use $\square^{+1}$ to denote the mlg $\{p_1(n_1+1), \cdots, p_k(n_k+1)\}^{\circledast} + p_{k+1}(n_{k+1}+1) + \cdots + p_{k+\ell}(n_{k+\ell}+1)$. That is, we replace each occurrence of a pair $p(n)$ in the representation of $\square$ by $p(n+1)$.

We are ready to define the function $Post_{Time}(\square_{in})$ for some input region generator $\square_{in}$. We start from $\square_{in}$ and perform an iteration, maintaining two sets V and W of region generators. Region generators in V are already analyzed and those in W are yet to be analyzed. We pick (also remove) a region generator $\square$ from W, add it to V (if it is not already included in V). We update W and V with new region generators according to the rules described below. We continue until W is empty. At this point we take $Post_{Time}(\square_{in}) = V$. Depending on the form of $\square$, we update W and V according to one of the following cases.

– If $\square$ is of the form $(\square_0, \square, \square_{\max})$, where $\square_0 \neq \square$. We add a region generator $(\square, young(\square_0) \bullet \square, \square_{\max} + old(\square_0))$ to W. This step corresponds to one rotation according to case 2 in the analysis above.
– If $\square$ is of the form $(\square, \square \bullet \square, \square_{max})$. Here the last element in the second component of the region generator is is an atomic expression (an mlg). We add the region generator $\left(\square^{+1}, \square, \square_{max}\right)$ to W. This step corresponds to one rotation according to case 1 in the analysis above.

– If $\square$ is of the form $(\square, \square \bullet \{\square_1, \ldots, \square_k\}^*, \square_{max})$. Here, the last expression in the second component of the region generator is a star expression. This case is similar to the previous one. However, the tokens corresponding to $\{\square_1, \ldots, \square_k\}^*$ now form an unbounded sequence with strictly increasing fractional parts. We add

$$\left(\square_i^{+1}, \left\{young(\square_1^{+1}), \ldots, young(\square_k^{+1})\right\}^* \bullet \square \bullet \{\square_1, \ldots, \square_k\}^*, \square_{\max} + Old^{\circledast}\right)$$

to V, and

$$\left(\square_i^{+1}, \left\{young(\square_1^{+1}), \ldots, young(\square_k^{+1})\right\}^* \bullet \square, \square_{\max} + Old^{\circledast}\right)$$

to W, for $i : 1 \leq i \leq k$. Here, Old is the union of the sets of symbols occurring in the set of mlgs $\left\{old(\square_1^{+1}), \ldots, old(\square_k^{+1})\right\}$. This step corresponds to performing a sequence of rotations of the forms of case 1 and case 2 above.

Notice that we add one of the newly generated region generators directly to V (and its "successor" to W). This is done in order to avoid an infinite loop where the same region generator is generated all the time.

– If $\square$ is of the form $(\square, \square, \square_{\max})$, i.e., all tokens have ages which are strictly greater than *max*, then we do not add any element to W.

The termination of this algorithm is guaranteed due to the fact that after a finite number of steps, we will eventually reach a point where we analyze region generators which will only characterize tokens with ages greater than *max* (i.e. will be of the form $(\square, \square, \square_{\max})$).

Post$_t$

For an input region generator $\square$, we compute (the downward closure of) the set of all markings which can be reached from a marking in $[\![\square]\!]$ by firing a discrete transition t, i.e we compute $Post_t(\square)$ as a finite set of region generators s.t $[\![Post(\square)]\!] = \{M' \mid \exists M \in [\![\square]\!].M \longrightarrow_t M'\}$.

Notice that from a downward closed set of markings, when we execute a timed transition, the set of markings reached is always downward-closed. But this is not the case for discrete transitions. Therefore, we consider the downward closure of the set of reachable markings in the following algorithm.

To give an algorithm for $Post_t$, we need to define an *addition* and a *subtraction* operation for region generators. An addition (subtraction) corresponds to adding (removing) a token in a certain age interval. These operations have hierarchical definitions reflecting the hierarchical structure of region generators.

We start by defining addition and subtraction for mlgs, defined over a finite set $P \times \{0, \ldots, \max\}$.

Given a *normal* mlg $\square = S^{\circledast} + a_1 + \cdots + a_\ell$ and a pair $p(n)$ where p is a place and n denotes the integral part of the age of a token in p, we define the *addition* $\square \oplus p(n)$ to be the mlg $\square + p(n)$.

The subtraction $\square \ominus p(n)$ is defined by the following three cases.

– If $p(n) \in S$, then $\square \ominus p(n) = \square$. Intuitively, the mlg $\square$ describes markings with an unbounded number of tokens each with an integral part equal to n, and each residing in place p. Therefore, after removing one such a token, we will still be left with an unbounded number of them.

- If $p(n) \not\in S$ and $a_i = p(n)$ for some $i : 1 \leq i \leq \ell$ then $□ \ominus p(n) = S^{\circledast} + a_1 + \cdots + a_{i-1} + a_{i+1} + \cdots + a_\ell$.
- Otherwise, the operation is undefined.

Addition and subtraction from mlgs over P is similar where instead of $p(n)$, we simply add (subtract) p.

Now, we extend the operations to wlgs defined over mlgs of the above form.

The addition $□ \oplus p(n)$ is a wlg $□$ consisting of the following three sets of wlgs.

1. For each $□_1$, $□_2$, and $□$ with $□ = □_1 \bullet □ \bullet □_2$, we have $□_1 \bullet (□ \oplus p(n)) \bullet □_2 \in (□ \oplus p(n))$.
2. For each $□_1$, $□_2$ and $□ = □_1 \bullet \{□_1, \cdots, □_k\}^* \bullet □_2$, we have for $i : 1 \leq i \leq k$, $□_1 \bullet \{□_1, \cdots, □_k\}^* \bullet (□_i \oplus p(n)) \bullet \{□_1, \cdots, □_k\}^* \bullet □_2 \in (□ \oplus p(n))$.
3. For each $□_1$ and $□_2$ with $□ = □_1 \bullet □_2$, we have $□_1 \bullet p(n) \bullet □_2 \in (□ \oplus p(n))$.

Intuitively, elements added according to the first two cases correspond to adding a token with a fractional part equal to that of some other token. In the third case the fractional part differs from all other tokens.

We define the subtraction $□ \ominus p(n)$, where $□$ is a wlg, to be a set of wlgs, according to the following two cases.

- If there is a star expression $e = \{□_1, \cdots, □_k\}^*$ containing the token we want to remove, i.e., if $□$ is of the form $□_1 \bullet e \bullet □_2$, and if any of the operations $□_i \ominus p(n)$ is defined for $i : 1 \leq i \leq k$, then $□ \ominus p(n) = \{□\}$.
- Otherwise, the set $□ \ominus p(n)$ contains wlgs of the form $□_1 \bullet □' \bullet □_2$ such that $□$ is of the form $□_1 \bullet □ \bullet □_2$ and $□' \in (□ \ominus p(n))$.

Now we describe how to use the addition and subtraction operations for computing $Post_t$. Addition and subtraction of pairs of the form $p(n)$ can be easily extended to pairs of the form $p(N)$ where $N \subseteq \{0, \ldots, max\}$, e.g $□ \ominus p(N) = \{□ \ominus p(n) |\ n \in N\}$.

We recall that, in a TPN, the effect of firing a transition is to remove tokens from the input places and add tokens to the output places. Furthermore, the tokens which are added or removed should have ages in the corresponding intervals. The effect of of firing transitions from the set of markings characterized by a region generator $□ = (□_0, □, □_{max})$ can therefore be defined by the following operations.

First, we assume an interval I of the form (x,y). The subtraction $□ \ominus p(I)$ is given by the union of the following sets of region generators.

- $(□_0 \ominus p(N), □, □_{\max})$ where each $n \in N$ is a natural number in the interval I. Intuitively, if the age of the token that is removed has a zero fractional part, then N contains the valid choices of integral part.
- $(□_0, □', □_{\max})$ such that $□' \in □ \ominus p(N)$, where $N = \{n |\ n \in \mathbb{N} \wedge x \leq n < y\}$ i.e., each n is a valid choice of integral part for the age of the token if it has a non-zero fractional part.
- $(□_0, □, □_{\max} \ominus p)$ if I is of the form $(x, □)$, i.e., the age of the token may be greater than max.

Addition is defined in a similar manner. The addition and subtraction operations will be similar if the interval is closed to the left. But if the interval is closed to the right, the last rule is undefined in that case.

We extend definition of subtraction and addition for subtracting a set of tuples $p(I)$ in the obvious manner. For a set of region generators □, we define $□ \oplus p(I) = \bigcup_{□ \in □} (□ \oplus p(I))$. Subtraction for a set of region generators is defined in a similar manner.

Let $A_{in}(t)$ be the set of input arcs given by $\{p(I) | \mathit{In}(t,p) = I\}$ and the set of output arcs $A_{out}(t)$ be given by $\{p(I) | \mathit{Out}(t,p) = I\}$.

We define,

$$\mathit{Post}_t(□) = (□ \ominus A_{in}(t)) \oplus A_{out}(t)$$

6 Acceleration

In this section, we explain how to accelerate the firing of a single transition interleaved with timed transitions from a region generator. We give a criterion which characterizes when acceleration can be applied. If the criterion is satisfied by an input region generator $□_{in}$ with respect to a transition t, then we compute a finite set $\mathit{Accel}_t(□_{in})$ of region generators such that $[\![\mathit{Accel}_t(□_{in})]\!] = \{M' | \exists M \in [\![□_{in}]\!].\ M(\longrightarrow_{\mathit{Time}} \cup \longrightarrow_t)^* M'\}$. We shall not compute the set $\mathit{Accel}_t(□_{in})$ in a single step. Instead, we will present a procedure Step_t with the following property: for each region generators $□_{in}$ there is an $n \geq 0$ such that $\mathit{Accel}_t(□) = \bigcup_{0 \leq i \leq n} (\mathit{Post}_{\mathit{Time}} \circ \mathit{Step}_t)^i(□)$. In other words, the set $\mathit{Accel}_t(□)$ will be fully generated through a finite number of applications of $\mathit{Post}_{\mathit{Time}}$ followed by Step_t. Since the reachability algorithm of Section 4 computes both $\mathit{Post}_{\mathit{Time}}$ and Step_t during each iteration, we are guaranteed that all region generators in $\mathit{Accel}_t(□_{in})$ will eventually be produced.

To define Step_t we need some preliminary definitions.

For a word atomic expression (mlg) $□ = \{a_1, \ldots, a_k\}^{\circledast} + a_{k+1} + \cdots + a_{k+\ell}$, we define $\mathit{sym}(□)$ as the set of symbols given by $\{a_1, \ldots, a_{k+\ell}\}$. For a word star expression $e = \{□_1, \ldots, □_k\}^*$, $\mathit{sym}(e) = \bigcup_i \mathit{sym}(□_i)$ for $i : 1 \leq i \leq k$.

Given a symbol $a \in A$ and an mlg □ over A of the form $S^{\circledast} + a_1 + \cdots + a_\ell$, we say that a is a $\circledast - \mathit{symbol}$ in □ if $a \in S$. Intuitively, a is a $\circledast - \mathit{symbol}$ in an mlg □ if it can occur arbitrarily many times in the multisets in □.
Given a wlg $□ = e_1 \bullet \cdots \bullet e_l$ over A, we say that a symbol $a \in A$ is a

- $\circledast - \mathit{symbol}$ in □ if there is an $i : 1 \leq i \leq l$ such that a is a $\circledast - \mathit{symbol}$ for some mlg □ occurring in wlg □.
- $* - \mathit{symbol}$ in □ if there is an $i : 1 \leq i \leq l$ such that $a \in \mathit{sym}(e_i)$ and e_i is a word star expression.

Intuitively, a is a $* - \mathit{symbol}$ in □ if it can occur an arbitrary number of times in arbitrarily many consecutive multisets in a word given by the wlg □.

In this section, we show how to perform acceleration when intervals are open, i.e of the form (x,y). It is straightforward to extend the algorithms to closed intervals (see [ADMN03] for details).

To compute the effect of acceleration, we define an operation $\uplus$.
Accelerated addition $\uplus$ corresponds to repeatedly adding an arbitrary number of tokens of the form $p(n)$ (with all possible fractional parts) to a region generator □.

First we define the operation $\uplus$ for mlgs. Given a mlg □ and a pair $p(n)$, the accelerated addition $□ \uplus p(n)$ is given by an mlg $□ + \{p(n)\}^{\circledast}$.

Given a wlg □, $□ \uplus p(n)$ can be inductively defined as follows.

- If $□ = □$, then $□ \uplus p(n) = \{\{p(n)\}^{\circledast}\}^*$.

– If $\Box = \Box \bullet \Box'$, then $\Box \uplus p(n) = \left\{\{p(n)\}^{\circledast}\right\}^{*} \bullet (\Box \uplus p(n)) \bullet (\Box' \uplus p(n))$
– If $\Box = \{\Box_1, \cdots, \Box_n\}^{*} \bullet \Box'$, then $\Box \uplus p(n) = \{\Box_1 \uplus p(n), \cdots, \Box_n \uplus p(n)\}^{*} \bullet (\Box' \uplus p(n))$

Accelerated addition can be extended to sets of pairs of the form $\{p(n_1), \ldots, p(n_k)\}$. Given a wlg $\Box$, we define $\Box \uplus \{p(n_1), \ldots, p(n_k)\} = \Box \uplus p(n_1) \uplus \cdots \uplus p(n_k)$.

Given a region generator $\Box = (\Box_0, \Box, \Box_{max})$ and a pair $p(I)$ where $I = (x,y)$, we define $\Box \uplus p(I) = \left(\Box_0 + S_1{}^{\circledast},\ \Box \uplus S_2,\ \Box_{max} + \{p_{max}\}^{\circledast}\right)$ where

– $S_1 = \{p(n) \mid n \in \mathbb{N} \wedge x < n < y\}$.
– $S_2 = \{p(n) \mid n \in \mathbb{N} \wedge x \leq n < y\}$.
– $p_{max} = p$ if $y = \Box$, $p_{max} = \Box$ otherwise.

For a set of pairs, $A = \{p_1(I_1), \cdots, p_k(I_k)\}$, we define $\Box \uplus A = \Box \uplus p_1(I_1) \uplus \cdots \uplus p_k(I_k)$.

Acceleration Criterion: For a discrete transition t, to check whether we can fire t arbitrarily many times interleaved with timed transitions, first we categorize the input places of t with respect to a region generator $\Box = (\Box_0, \Box, \Box_{max})$ and the transition t.

Type 1 place An input place p of t is said to be of *Type 1* if one of the following holds. Given $In(t,p) = (x,y)$,
  – there is an integer n such that $x < n < y$ and $p(n)$ is a $\circledast - symbol$ in $\Box_0$.
  – there is an integer n such that $x \leq n < y$ and $p(n)$ is a $\circledast - symbol$ or a $* - symbol$ in $\Box$.
  – p is a $\circledast - symbol$ in $\Box_{max}$ and $y = \Box$.

Intuitively, unbounded number of tokens with the ”right age” are available in an input place p of Type 1.

Type 2 place An input place p of t is of *Type 2* if it is not of Type 1, but it is an output place and both the following holds.
  1. Given $In(t,p) = I$, $\Box \ominus p(I) \neq \emptyset$. Intuitively, for a Type 2 place, there is initially at least one token of the ”right age” for firing t.
  2. $In(t,p) \cap Out(t,p)$ is a non-empty interval. Intuitively, a token generated as output in any firing may be re-used as an input for the next firing.

We accelerate if each input place of t is a Type 1 place or a Type 2 place.

Acceleration: Let $A_{in}(t), A_{out}(t)$ be the set of input and output arcs as defined in Section 5. Now, given a region generator $\Box$, we describe acceleration in steps.

– First we subtract input tokens from all input places. Then we add tokens to Type 2 places (places which always re-use an output token as an input for next firing). Formally we compute a set of region generators $\Box = (\Box \ominus A_{in}(t)) \oplus T_2$ where $T_2 = \{p(I) \mid p$ is of Type $2 \wedge p(I) \in A_{out}(t)\}$ is the set of output arcs from Type 2 places.
– Next, we accelerate addition for each region generator in $\Box$ and add tokens of all possible ages in the output places which are not of Type 2 (Type 2 places re-use input tokens, therefore do not accumulate tokens), i.e, we compute

$$Step_t(\Box) = \bigcup_{\Box' \in \Box} \Box' \uplus (A_{out}(t) \setminus T_2)$$

Theorem 5. *If the acceleration criterion holds from a region generator $\Box$ with respect to a transition t in a TPN, there is an $n \geq 0$ such that $Accel_t(\Box) = \bigcup_{0 \leq i \leq n} (Post_{Time} \circ Step_t)^i(\Box)$.*

7 Experimental Results

We have implemented a prototype based on our algorithm and used it to verify two protocols.

Parameterized Model of Fischer's protocol Fischer's protocol is intended to guarantee mutual exclusion in a concurrent system consisting of arbitrary number of processes trying to get access to a shared variable (see e.g. [AN01] for more details). The pro-

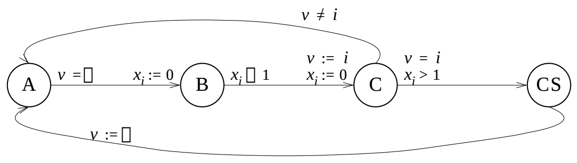

Fig. 2. One process in Fischer's Protocol for Mutual Exclusion

tocol consists of processes running a code, which is graphically described in Figure 2. Each process i has a local clock, x_i, and a control state, which assumes values in the set $\{A, B, C, CS\}$ where A is the initial state and CS is the critical section. The processes read from and write to a shared variable v, whose value is either $\perp$ or the index of one of the processes. Our aim is to show that this protocol allows at most one process to enter the critical section CS. Figure 2 shows a timed Petri net model [AN01] of the parameterized

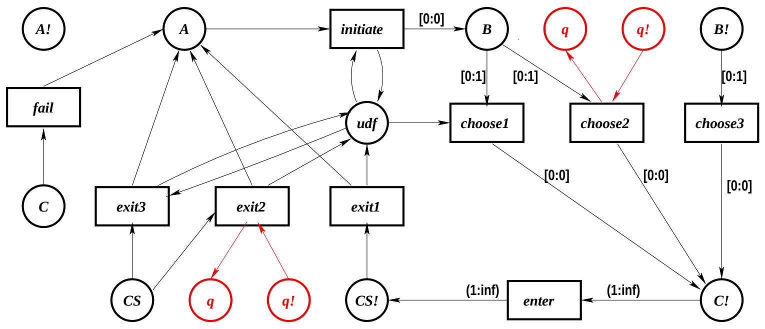

Fig. 3. TPN model for Parameterized version of Fischer's Protocol for Mutual Exclusion

protocol. Tokens in the places A, B, C, CS, $A!$, $B!$, $C!$ and $CS!$ represent processes running the protocol. In the figure, We use q to denote any of the process states A, B, C, CS. The places marked with ! represent that the value of the shared variable is the index of the process modeled by the token in that place. We use a place *udf* to represent that the value of the shared variable is $\perp$. The critical section is modelled by the places CS and $CS!$, so mutual exclusion is violated when the total number of tokens in those places is at least two.

In order to prove the mutual exclusion property, we specify markings with two tokens in $CS, CS!$ as the bad markings. We use $\left(\{A(0), A(1)\}^{\circledast} + udf(0), \{\{A(0)\}^{\circledast}\}^{*}, \{A\}^{\circledast}\right)$ as the initial region generator $\square_{init}$. $\square_{init}$ characterizes arbitrarily many processes in A having any clock value (age) and one token in udf with age 0. Furthermore, to prove that mutual exclusion is guaranteed, we checked the membership of the bad markings (characterizing an upward closed set of bad states) in the computed set of region generators.

Producer-Consumer System[NSS01] We consider the producer/consumer system mentioned in [NSS01][2]. Figure 3 shows a timed Petri net model of the producer/consumer

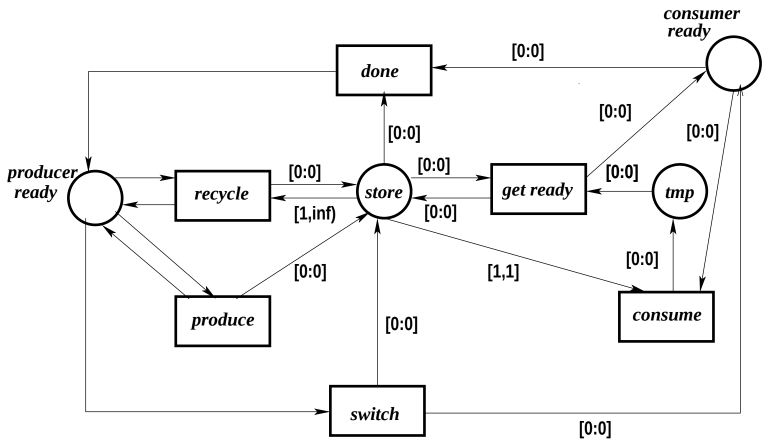

Fig. 4. TPN model for Producer/Consumer System

system. A token in the place *producer_ready* means that the producer can produce *items*; firing transition *produce* creates new *items* in place *store*. The consumer consumes items of age 1 if the place *consumer_ready* has a token. Old items (of age greater than 1) are recycled by the producer using the transition *recycle*. Control is switched between the producer and the consumer by transitions *switch* and *done*. We use $(producer\_ready(0), \square, \square)$ as the initial region generator $\square_{init}$ which characterizes a single token in place "*producer_ready*" with age 0.

Our program computes the reachability set for both protocols (the set of computed region generators are shown in Appendix). The procedure fails to terminate without the use of acceleration in both cases. It took 35MB memory and 26s to analyse Fischer's protocol, and 3.25MB memory and 3s to analyse producer/consumer system on a 1 GHz processor with 256 MB RAM.

Abstract Graph Using forward analysis of a TPN, our tool also generates a graph G which is a finite-state abstraction of the TPN. Each state in G corresponds to a region generator in the reachability set. Edges of G are created as follows. Consider two region generators $\square_1, \square_2$ in the reachability set. If there is a region generator $\square_2' \in Post_t(\square_1)$ such

[2] [NSS01] considers a TPN model with local time in each place.

that $\square'_2 \subseteq \square_2$, then we add an edge $\square_1 \xrightarrow{t} \square_2$ to G. Similarly, if there is a region generator $\square'_2 \in Post_{Time}(\square_1)$ such that $\square'_2 \subseteq \square_2$, then we add an edge $\square_1 \xrightarrow{\square} \square_2$. Notice that each region generator in the post-image should be included in some region generator in the computed set. It is straightforward to show that the abstract graph simulates the corresponding TPN model.

The graph obtained by the above analysis contains 12 states and 54 edges in the case of Fischer's protocol; and 11 states and 21 edges in the case of producer/consumer system. Furthermore, we use *The Concurrency Workbench* [CPS89] to minimize the abstract graphs modulo weak bisimilarity. Figure 4 shows the minimized finite state labelled transition systems for the above protocols.

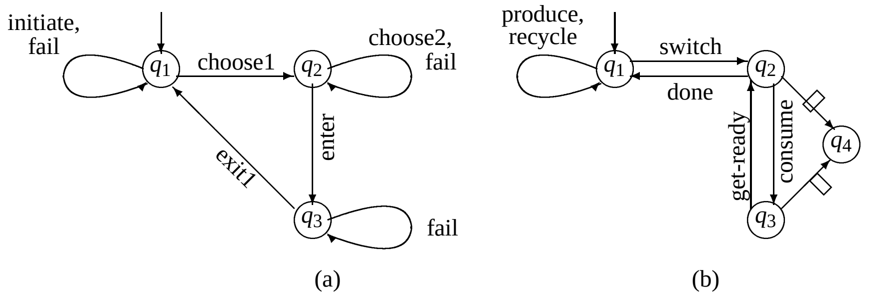

Fig. 5. Minimized abstract graph for (a) Fischer's protocol. (b) Producer/Consumer System.

8 Conclusions and Future Research

We have described how to perform forward analysis augmented with acceleration for timed Petri nets, using a symbolic representation called *region generators*. There are a number of interesting directions for future research. Firstly, we show how to accelerate with respect to single discrete transition interleaved with timed transitions. A remaining challenge is to extend the technique and consider accelerations of *sequences* of discrete transitions. It is not clear to us whether such accelerations are computable in the first place. Secondly, we assume a lazy behaviour of TPNS. It is well-known that checking safety properties is undecidable for TPNs with *urgent* behaviours even if the net is safe (bounded). Therefore, designing acceleration techniques is of particular interest for urgent TPNs. Notice that downward closure is no longer an exact abstraction if the behaviour is urgent. Thirdly, we use *region generators* for symbolic representation. We want to investigate designing efficient data structures (e.g . *zone generators* corresponding to a large number of region generators). *Zones* are widely used in existing tools for verification of timed automata [LPY97,Yov97]. Intuitively, a zone generator will correspond to a state in each minimized automaton in Figure 4. Finally, We aim at developing generic methods for building downward closed languages, in a similar manner to the methods we have developed for building upward closed languages in [AČJYK00]. This would give a general theory for forward analysis of infinite state systems, in the same way the work in [AČJYK00] is for backward analysis. Simple regular expressions of [ABJ98] and the region generators of this paper are examples of data structures which might be developed in a systematic manner within such a theory.

References

[ABJ98] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *Proc. CAV'98*, volume 1427 of *LNCS*, pages 305–318, 1998.

[AČJYK00] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160:109–127, 2000.

[AD90] R. Alur and D. Dill. Automata for modelling real-time systems. In *Proc. ICALP '90*, volume 443 of *LNCS*, pages 322–335, 1990.

[ADMN03] Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén. Forward reachability analysis of timed petri nets. Technical Report 2003-056, Dept. of Information Technology, Uppsala University, Sweden, 2003. http://user.it.uu.se/˜pritha/Papers/tpn.ps.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. Timed Petri nets and BQOs. In *Proc. ICATPN'2001*, volume 2075 of *LNCS*, pages 53 –70, 2001.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *Proc. CAV'96*, volume 1102 of *LNCS*, pages 1–12, 1996.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. In *Proc. ICALP'97*, volume 1256 of *LNCS*, 1997.

[BLO98] S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems automatically and compositionally. In *Proc. CAV'98*, volume 1427 of *LNCS*, pages 319–331, 1998.

[Bow96] F. D. J. Bowden. Modelling time in Petri nets. In *Proc. Second Australian-Japan Workshop on Stochastic Models*, 1996.

[CPS89] R. Cleaveland, J. Parrow, and B. Steffen. A semantics-based tool for the verification of finite-state systems. In Brinksma, Scollo, and Vissers, editors, *Protocol Specification, Testing, and Verification IX*, pages 287–302. North-Holland, 1989.

[DR00] G. Delzanno and J. F. Raskin. Symbolic representation of upward-closed sets. In *Proc. TACAS'00*, volume 1785 of *LNCS*, pages 426–440, 2000.

[FIS00] A. Finkel, S. Purushothaman Iyer, and G. Sutre. Well-abstracted transition systems. In *Proc. CONCUR'00*, pages 566–580, 2000.

[FRSB02] A. Finkel, J.-F. Raskin, M. Samuelides, and L. Van Begin. Monotonic extensions of petri nets: Forward and backward search revisited. In *Proc. Infinity'02*, 2002.

[God94] J.C. Godskesen. *Timed Modal Specifications*. PhD thesis, Aalborg University, 1994.

[KM69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and Systems Sciences*, 3(2):147–195, May 1969.

[LBBO01] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc. TACAS'01*, volume 2031 of *LNCS*, 2001.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Software Tools for Technology Transfer*, 1(1-2), 1997.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*, volume 1776 of *LNCS*, 2000.

[NSS01] M. Nielson, V. Sassone, and J. Srba. Towards a distributed time for petri nets. In *Proc. ICATPN'01*, pages 23–31, 2001.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. LICS'86*, pages 332–344, June 1986.

[Yov97] S. Yovine. Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1-2), 1997.