Using Forward Reachability Analysis for Verification 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, Lynch and Shavit's mutual exclusion protocol and a producer-consumer protocol. We also used the tool to extract finite-state abstractions of these protocols.

Key words: Timed Petri Nets, Model Checking, Reachability Analysis, Downward Closed Languages

ACM CCS Categories and Subject Descriptors: Verification, Computing Review Category: F.3.1 (Specifying and Verifying and Reasoning about Programs)

1. Introduction

Petri nets are one of the most widely used graphical models for analysis and verification of concurrent systems. A number of timed extensions of Petri nets [Razouk and Phelps 1985],[Merlin and Farber 1976],[Ramamoorthy and Ho 1980],[Holliday and Vernon 1987],[Coolahan and Roussopoulos 1983],[Ghezzi *et al.* 1991],[Abdulla and Nylén 2001] have been proposed in order to capture the timing aspects of the concurrent systems (see [Bowden 1996] for a survey). We consider the *timed Petri net (TPNs)* model of [Abdulla and Nylén 2001], in which each token has an "age" which is represented by a real valued clock.

As opposed to timed automata [Alur and Dill 1990], TPNs operate on a potentially unbounded number of clocks. This implies that TPNs can, among other things,

[*] A preliminary version of this paper has been published in Formats-FTRTFT 2004.

Received September 2005; revised June 2007.

model parameterized timed systems (systems consisting of an unbounded number of timed processes) [Abdulla and Nylén 2001].

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 [Vardi and Wolper 1986], 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 [Abdulla and Nylén 2001], 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. [Bensalem *et al.* 1998],[Lakhnech *et al.* 2001]).

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* [Alur and Dill 1990]. 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, Lynch and Shavit's protocol and also for a simple producer/consumer protocol. Also, we used the tool for generating finite state abstractions of these protocols.

Related Work: [Abdulla *et al.* 1998] 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 [Delzanno and Raskin 2000] and [Finkel *et al.* 2002] consider (untimed) Petri nets and give symbolic representations for upward closed sets and downward closed sets of markings, respectively. The works in [Finkel *et al.* 2000],[Boigelot and Godefroid 1996],[Bouajjani and Habermehl 1997] 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.

[Abdulla and Nylén 2001] 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 $\mathbb{N}$ and $\mathbb{R}^{\geq 0}$ to denote the sets of natural numbers and nonnegative reals respectively. We use a set *Intrv* of intervals. An open interval is written as (w, z) where $w \in \mathbb{N}$ and $z \in \mathbb{N} \cup \{\infty\}$. 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 $A^{\circledast}$ to denote the set of finite multisets over A. 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 ϵ to denote an empty multiset.

For a set A with ordering $\preceq$, we use A^* to denote the set of finite words over A. For a word $w \in A^*$, 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 use $w_1 \bullet w_2$ to denote the concatenation of the words w_1 and w_2. We define the ordering $\preceq^w$ on the set of words over A such that $w_1 \preceq^w w_2$ if there is a strictly increasing injection $h\ :\ \{1, \ldots, |w_1|\} \rightarrow \{1, \ldots, |w_2|\}$ where $w_1(i) \preceq w_2(h(i))$ for $i : 1 \leq i \leq |w_1|$. Later we will use $\leq^m$ for $\preceq$ in Section 3. We use ϵ to denote an empty word as well.

Timed Petri Nets A *Timed Petri Net (TPN)* is a tuple $N = (P, T, \mathit{In}, \mathit{Out})$ 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*.

We let *max* be the maximum natural number which appears (in the intervals) on the arcs of the TPN.

If $\mathit{In}(t,p)$ ($\mathit{Out}(t,p)$) 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. For example, if $M = [p_1(2.5)\ ,\ p_1(1.3)\ ,\ p_2(4.7)\ ,\ p_2(4.7)]$, then, in the marking M, there are two tokens with ages 2.5 and 1.3 in p_1, and two tokens each with age 4.7 in p_2. Abusing notation again, we define, for each place p, a multiset $M(p)$ over $\mathbb{R}^{\geq 0}$, where $M(p)(x) = M(p(x))$.

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 $M_1 \longrightarrow_{\mathit{Time}} M_2$ to denote that $M_1 \longrightarrow_x M_2$ for some $x \in \mathbb{R}^{\geq 0}$.

We define the set of *discrete transitions* $\longrightarrow_{\mathit{Disc}}$ 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 $\{p(\mathcal{I}) \mid \mathit{In}(t,p) = \mathcal{I}\}$ is of the form $\{p_1(\mathcal{I}_1), \ldots, p_k(\mathcal{I}_k)\}$, the set of output arcs $\{p(\mathcal{I}) \mid \mathit{Out}(t,p) = \mathcal{I}\}$ is of the form $\{q_1(\mathcal{J}_1), \ldots, q_\ell(\mathcal{J}_\ell)\}$, 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:
- $b_1 \leq^m M_1$
- $x_i \in \mathcal{I}_i$, for $i : 1 \leq i \leq k$.

[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 $p(\mathcal{I})$ instead of $(p, \mathcal{I})$ where $\mathcal{I} \in \mathit{Intrv}$.

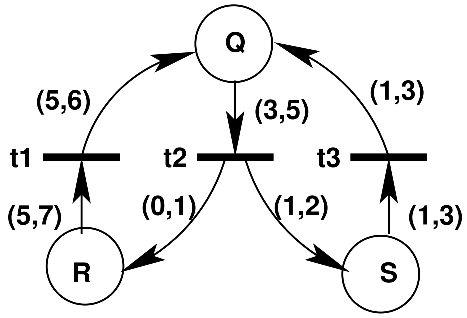


Fig. 2.1: A small timed Petri net.

- $y_i \in \mathcal{J}_i$, for $i : 1 \le i \le \ell$.
- $M_2 = (M_1 - b_1) + b_2$.

Intuitively, a transition t may be fired only if for each incoming arc to the transition, there 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 $\overset{*}{\longrightarrow}$ to denote the reflexive transitive closure of $\longrightarrow$. We say that M_2 is *reachable* from M_1 if $M_1 \overset{*}{\longrightarrow} M_2$. We define *Reach*$(M)$ to be the set $\left\{ M' \mid M \overset{*}{\longrightarrow} M' \right\}$.

Example 1. Figure 2.1 shows an example of a TPN where $P = \{Q, R, S\}$ and $T = \{t_1, t_2, t_3\}$. For instance, $In(t_2, Q) = (3, 5)$ and $Out(t_2, R) = (0, 1)$ and $Out(t_2, S) = (1, 2)$. A marking of the given net is $M_0 = [Q(2.0),\ R(4.3),\ R(3.5)]$. A timed transition from M_0 is given by $M_0 \longrightarrow_{1.5} M_1$ where $M_1 = [Q(3.5),\ R(5.8),\ R(5.0)]$. An example of a discrete transition is given by $M_1 \longrightarrow_{t_2} M_2$ where $M_2 = [R(0.2),\ S(1.6),\ R(5.8),\ R(5.0)]$.

Notice that untimed Petri nets are a special case in our model where all intervals are of the form $[0, \infty)$.

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".

In fact TPNs are infinite in two directions: they have unbounded number of tokens, and each token has a real-valued clock. The infiniteness due to real-valued clocks are handled by *regions*, introduced next.

Regions Regions were first designed for timed automata [Alur and Dill 1990] (automata operating on a finite number of clocks) and hence they are not powerful enough to capture the behaviour of TPNs. A *region* defines the integral parts of clock values up to *max* (the exact age of a token is not useful 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. We define an ordering on markings of TPNs which extends the equivalence relation on markings induced by the classical region graph construction of [Alur and Dill 1990].

Following Godskesen [Godskesen 1994] we represent a region for TPN by a triple (b_0, w, b_{max}) where

- $b_0 \in (P \times \{0, \dots, max\})^{\circledast}$ is a multiset of pairs. A pair of the form $p(n)$ represents a token with age exactly n in place p.
- $w \in \left((P \times \{0, \dots, max - 1\})^{\circledast}\right)^*$ is a word over the set $(P \times \{0, \dots, max - 1\})^{\circledast}$, i.e., w is a word where each element in the word is a multiset over $P \times \{0, \dots, max - 1\}$. 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.
- $b_{max} \in P^{\circledast}$ 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.

Assume a marking $M = [p_1(x_1), \dots, p_n(x_n)]$. We define a unique region $\mathcal{R}_M = (b_0, b_1 b_2 \cdots b_m, b_{m+1})$ such that M satisfies $\mathcal{R}_M$ (written as $M \models \mathcal{R}$) as follows:

Each b_j is of the form $\left[q_{j1}(y_{j1}), \dots, q_{j\ell_j}(y_{j\ell_j})\right]$ for $j : 0 \leq j \leq m$ and b_{m+1} is of the form $[q_{m+1\,1}, \dots, q_{m+1\,l_{m+1}}]$. We also define a bijection h from the set $\{1, \dots, n\}$ to the set of pairs $\left\{(j, k) \mid (0 \leq j \leq m + 1) \wedge (1 \leq k \leq \ell_j)\right\}$ 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 $\mathcal{R}$.
- $h(i) = (j, k)$ and $j = m + 1$ if $x_i > max$. Tokens older than max should correspond to elements in multiset b_{m+1}. The actual ages of these tokens are not relevant.
- $h(i) = (j, k)$ and $j \leq m$ if $x_i \leq max$ and $\lfloor x_i \rfloor = y_{j\,k}$. The integral part of the age of tokens should agree with the natural number specified by the corresponding elements in w.
- $h(i) = (0, k)$ for some k if $fract(x_i) = 0$ and $x_i \leq max$. Tokens with zero fractional parts correspond to elements in multiset b_0.
- $h(i_1) = (j_1, k_1)$ and $h(i_2) = (j_2, k_2)$ for $j_1 \leq j_2 \leq m$ if $x_{i_1}, x_{i_2} < max$ and $fract(x_{i_1}) \leq fract(x_{i_2})$. Tokens with equal fractional parts correspond to elements in the same multiset (unless they belong to b_{m+1}). The ordering among multisets inside $\mathcal{R}$ reflects the ordering among fractional parts in clock values.

From the above definitions, it is straightforward that each region characterizes an infinite set of markings. We let $[\![\mathcal{R}]\!] = \{M \mid \mathcal{R}_M = \mathcal{R}\} = \{M \mid M \models \mathcal{R}\}$.

The region construction defines an equivalence relation $\equiv$ on the set of markings such that $M_1 \equiv M_2$ iff $\mathcal{R}_{M_1} = \mathcal{R}_{M_2}$. Following [Alur and Dill 1990], it can be

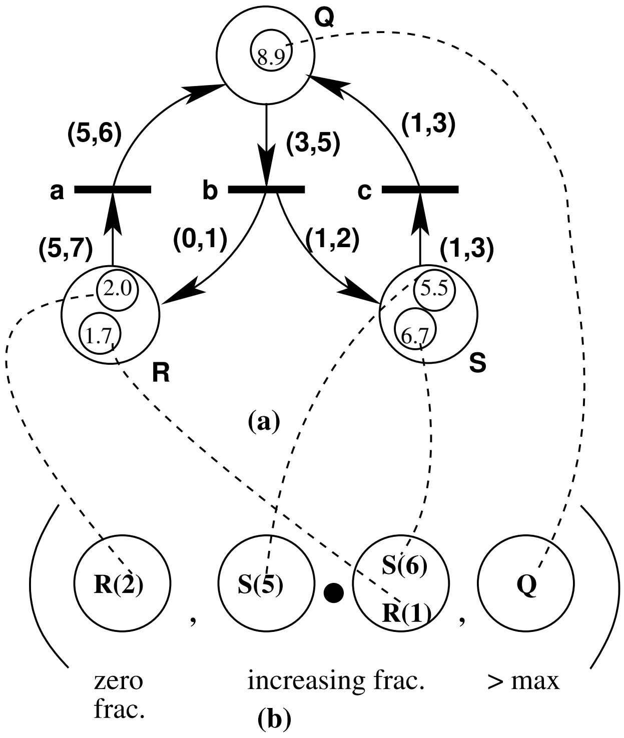


Fig. 2.2: Marking M in (a) satisfies region $\mathcal{R}$ in (b).

easily shown 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 $M_3 \longrightarrow M_4$.

Example 2. Consider the TPN N in Figure 2.1 with $max = 7$. Figure 2.2(a) shows a marking $M = [R(2.0)\ , S(5.5),\ R(1.7),\ S(6.7),\ Q(8.9)]$. Figure 2.2(b) shows the unique region $\mathcal{R}_M = ([R(2)]\ ,\ [S(5)]\ \bullet\ [R(1),\ S(6)]\ ,\ [Q])$. such that $M \models \mathcal{R}_M$. In Figure 2.2(b), each circle corresponds to a multiset of tokens of N with same fractional parts. Dotted lines show how the tokens of M in TPN corresponds to elements in the region $\mathcal{R}_M$.

Orderings First 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 $M_2' \leq^m M_2$. 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 $M_3 \longrightarrow M_4$.*

Proof. Suppose that $M_1 \preceq M_3$. By definition of $\preceq$ there is an M_3' with $M_1 \equiv M_3'$ and $M_3' \leq^m M_3$. From the definition of $\leq^m$ we know that there is an M_3'' such that $M_3 = M_3' + M_3''$. Since $\equiv$ is a congruence, there is M_4' such that $M_2 \equiv M_4'$ and $M_3' \longrightarrow M_4'$. We define $M_4 = M_4' + M_4''$ where $M_4'' = M_3''$ if $M_3' \longrightarrow_t M_4'$ for some discrete transition t, and $M_4'' = \left(M_3''\right)^{+x}$ if $M_3' \longrightarrow_x M_4'$ for some $x \in \mathbb{R}^{\geq 0}$. □

Example 3. Consider the TPN N in Figure 2.1 where $max = 7$. Here is an example of a region $\mathcal{R} = ([R(2)],\ [S(5)] \quad [R(1),\ S(5),\ S(2)],\ [Q])$. 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 $M_2 \leq^m M_3$ and $M_1 \equiv M_2$, we have $M_1 \preceq M_3$.

Next we define an ordering $\leq^r$ on regions such that if $\mathcal{R}_1 = \left(b_0^1, w^1, b_{max}^1\right)$ and $\mathcal{R}_2 = \left(b_0^2, w^2, b_{max}^2\right)$ then $\mathcal{R}_1 \leq^r \mathcal{R}_2$ iff $b_0^1 \leq^m b_0^2$, $w^1 \leq^w w^2$, and $b_{max}^1 \leq^m b_{max}^2$. We use $\mathcal{R}_1 \leq^r \mathcal{R}_2$ to mean that for each $M_1 \in [\![\mathcal{R}_1]\!]$ and $M_2 \in [\![\mathcal{R}_2]\!]$, $M_1 \preceq M_2$.

Upward (downward) closed sets of regions and upward (downward) closure of a set of regions with respect to $\leq^r$ can be defined in a similar manner to that for markings.

Coverability Problem for TPNs

Instance: A set of initial markings M_{init} and a finite set M_{fin} of final markings.

Question: $Reach(\mathsf{M}_{init}) \cap \left(\mathsf{M}_{fin} \uparrow\right) = \emptyset$?

The coverability problem is interesting from the verification point of view, since checking safety properties can often be reduced to coverability[Vardi and Wolper 1986]. 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 $\mathsf{M}_{fin} \uparrow$. (Notice that $\mathsf{M}_{fin} \uparrow$ is upward closed with respect to the ordering $\preceq$.)

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 $(Reach(\mathsf{M}_{init})) \downarrow \cap \mathsf{M} = \emptyset$.*

Proof. It is obvious that $Reach(\mathsf{M}_{init}) \cap \mathsf{M} = \emptyset$ if $(Reach(\mathsf{M}_{init})) \downarrow \cap \mathsf{M} = \emptyset$.

We show the other direction. Suppose that there is a marking $M \in (Reach(\mathsf{M}_{init})) \downarrow \cap \mathsf{M}$. This means that there is a marking $M' \in (Reach(\mathsf{M}_{init}))$ such that $M \preceq M'$ and $M \in \mathsf{M}$, i.e., $M' \in \mathsf{M}$, since M is upward closed. This implies $(Reach(\mathsf{M}_{init})) \cap \mathsf{M} \neq \emptyset$. Contradiction. □

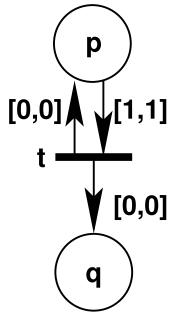


Fig. 2.3: A timed Petri net for which Karp-Miller algorithm cannot be applied.

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

Infeasibility of the Karp-Miller Algorithm The Karp-Miller algorithm [Karp and Miller 1969] is the classical method used for checking coverability in untimed Petri nets. However, it is not obvious how to extend the algorithm to TPNs. [Karp and Miller 1969] constructs a reachability tree starting from an initial marking. It detects paths in the reachability tree which lead 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 ω (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 $M_1 \leq^m M_2$ and $M_1 \neq M_2$) this conclusion cannot be drawn. For instance, consider the Petri net in Figure 2.3. If we start from marking $M_0 = [p(0)]$, we can let time pass by one unit, reach $M_1 = [p(1)]$, then fire transition t and reach a marking $M_2 = [p(0), q(0)]$. However, it is not the case that unboundedly many tokens with age $q(0)$ are generated, even though $M_0 \leq^m M_2$ and $M_0 \neq M_2$. In fact, in this case 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 ω, an unbounded number of tokens with ages less than max may still appear in the analysis. Termination is therefore not guaranteed.

3. Region Generators

TPNs are infinite in two directions: they have unbounded number of tokens, and each token has a real-valued clock. The infiniteness due to real-valued clocks are handled by regions (Section 2). However, to handle the infiniteness due to unbounded number of tokens, 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.

3.1 Multiset Language Generators (mlgs)

We define *multiset language generators (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 ordering $\leq^m$ on multisets (Section 2).

We define *(downward-closed) multiset language generators* (or *mlgs* for short) over the finite alphabet A. Each mlg ϕ over A defines a multiset language over A, denoted $L(\phi)$, 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 $a \in A$. $L(a) = \{[a]\ ,\ \epsilon\}$.
  - 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* ϕ is a (possibly empty) sequence $e_1 + \cdots + e_\ell$ of expressions. $L(\phi) = \{b_1 + \cdots + b_\ell \mid b_1 \in L(e_1), \cdots, b_\ell \in L(e_\ell)\}$. We denote an empty mlg by ϵ and define that $L(\epsilon) = \{\epsilon\}$ (which is a special case of $L(e_1 + \cdots + e_\ell)$ with $\ell = 0$).

We also consider sets of mlgs which we interpret as unions. If $\Phi = \{\phi_1, \cdots, \phi_m\}$ is a set of mlgs, then $L(\Phi) = L(\phi_1) \cup \cdots \cup L(\phi_m)$. We let $L(\emptyset) = \emptyset$.

THEOREM 1. *For each downward closed multiset language L over an alphabet A there is a set Φ of mlgs over A such that $L = L(\Phi)$.*

PROOF. See Appendix. □

Sometimes we identify mlgs with the languages they represent, so given two mlgs ϕ_1, ϕ_2, we write $\phi_1 \subseteq \phi_2$ (rather than $L(\phi_1) \subseteq L(\phi_2)$), and given a multiset b and an mlg ϕ, we write $b \in \phi$ (rather than $b \in L(\phi)$), etc.

Normal Form An mlg ϕ 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 $i : 1 \leq i \leq k$, $e_i \not\subseteq e$.

For each mlg ϕ, there is a unique (up to commutativity of the operators) normal mlg ϕ' such that $L(\phi') = L(\phi)$. We can derive ϕ' from ϕ by performing the following operations.

- Delete each atomic expression a from ϕ in case there is a star expression of the form $S^{\circledast}$ in ϕ such that $a \in S$. The language of the mlg is preserved since $L(a + S^{\circledast}) = L(S^{\circledast})$.
- Merge all star expressions using the property that $L({S_1}^{\circledast} + {S_2}^{\circledast}) = L((S_1 \cup S_2)^{\circledast})$.

A set of mlgs $\Phi = \{\phi_1, \cdots, \phi_m\}$ is said to be *normal* if each mlg ϕ_i is normal and $\phi_i \not\subseteq \phi_j$ for $1 \leq i \neq j \leq m$. We can transform each set of mlgs into normal form by transforming each member of Φ into normal form as described above, and by eliminating redundant members of Φ using the entailment algorithm described below.

From now on, (sets of) mlgs will always be assumed to be in a normal form.

Entailment In the following, we give an algorithm for computing entailment $\subseteq$ for (sets of) mlgs.

The relation $\subseteq$ is the least partial order on expressions satisfying

$$\begin{array}{lcll} a & \subseteq & S^{\circledast} & \text{IF } a \in S \\ {S_1}^{\circledast} & \subseteq & {S_2}^{\circledast} & \text{IF } S_1 \subseteq S_2 \end{array}$$

Given the algorithm for entailment of expressions, we can compute the entailment of mlgs as follows:

Consider the base cases. $\epsilon \subseteq \phi_2$ and $\phi_1 \not\subseteq \epsilon$ if $\phi_1 \neq \epsilon$. Given two non-empty mlgs $\phi_1 = e_1 + \phi_1'$ and $\phi_2 = e_2 + \phi_2'$, we have $\phi_1 \subseteq \phi_2$ iff one of the following holds.

(1) $e_1 = a$, $e_1 \not\subseteq e_2$ and $\phi_1 \subseteq \phi_2'$.

(2) $e_1 = e_2 = a$ and $\phi_1' \subseteq \phi_2'$.

(3) $e_2 = S^{\circledast}$, $e_1 \subseteq e_2$ and $\phi_1' \subseteq \phi_2$.

In a normal mlg, we assume that the atomic expressions in an mlg are sorted. This means that the entailment algorithm will have linear complexity.

Next, we consider entailment for sets of mlgs. We use the following lemma.

Lemma 3. *For mlgs $\phi, \phi_1, \ldots, \phi_m$, if $\phi \subseteq \{\phi_1, \cdots, \phi_m\}$, then $\phi \subseteq \phi_i$ for some $i \in \{1, \ldots, m\}$.*

Proof. See Appendix. □

From Lemma 3 and the algorithm for entailment of mlgs, it follows that

Theorem 2. *Entailment among mlgs can be checked in linear time and entailment of sets of mlgs can be checked in quadratic time.*

Example 4. Consider a finite alphabet $A = \{a, b, c\}$ and the set of multisets $A^{\circledast}$ over A. Given mlg $\phi_1 = \{a, b\}^{\circledast} + c$ (i.e., the multiset language over A containing at most one c and an arbitrary number of a's and b's), examples of multisets in $L(\phi_1)$ are $\left[a^2, b\right], [b, c], \left[a^3, b^2, c\right]$. Consider $\phi_2 = b + c$, i.e., the multiset language containing at most one b and one c. $L(\phi_2) = \{\epsilon, [c], [b], [b, c]\}$. Notice that ϕ_1, ϕ_2 are in normal form and $\phi_2 \subseteq \phi_1$. Furthermore $L(\phi_1)$ and $L(\phi_2)$ are both downward closed. Figure 3.4 graphically describes ϕ_1 and ϕ_2. Sets are drawn as ellipses and mlgs are shown as circles.

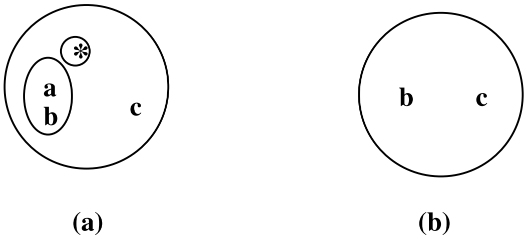


Fig. 3.4: mlgs (a) ϕ_1. (b) ϕ_2.

3.2 *Word Language Generators (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 $(A^{\circledast})^*$ (recall that for a set A, we use A^* to denote the set of finite words over A). The language is then a subset of $(A^{\circledast})^*$. Notice that the underlying alphabet, namely $A^{\circledast}$ is infinite.

From Section 2, recall that $|w|$ defines the size of the word and $\bullet$ is used for the concatenation of words. For a word $w \in L$, observe that $w(i)$ is a multiset over A.

From the definition of $\leq^w$ in Section 2, also observe that now the ordering $\leq^w$ on the set of words over A is defined 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 ϕ over A.
  - a *word star expression* of the form $\{\phi_1, \cdots, \phi_k\}^*$, where $\phi_1, \ldots, \phi_k$ are mlgs over A.
    $L(\{\phi_1, \cdots, \phi_k\}^*) =$
    $\{b_1 \bullet \cdots \bullet b_m \mid (m \geq 0) \text{ and } b_1, \ldots, b_m \in L(\phi_1) \cup \cdots L(\phi_k)\}$.
- A *word language generator (wlg)* ψ over A is a (possibly empty) concatenation $e_1 \bullet \cdots \bullet e_\ell$ of word expressions $e_1, \ldots, e_\ell$. $L(\psi) = \{w_1 \bullet \cdots \bullet w_\ell \mid w_1 \in L(e_1) \wedge \cdots \wedge w_\ell \in L(e_\ell)\}$.

Notice that the concatenation operator is associative, but not commutative (as is the operator + for multisets). Again, we denote the empty wlg by ϵ and define that $L(\epsilon) = \{\epsilon\}$ (a special case of . $L(e_1 \bullet \cdots \bullet e_\ell)$ with $\ell = 0$) and $L(\emptyset) = \emptyset$.

For a set $\Psi = \{\psi_1, \cdots, \psi_m\}$ of wlgs, we define $L(\Psi) = L(\psi_1) \cup \cdots \cup L(\psi_m)$. We also identify wlgs with word languages, as we did in case of mlgs and multiset languages.

Theorem 3. *For each downward closed word language L, there is a set Ψ of wlgs such that $L = L(\Psi)$.*

Proof. See Appendix. □

Normal Form

A word atomic expression e of the form ϕ is said to be in normal form if ϕ is a normal mlg. A word star expression $\{\phi_1, \ldots, \phi_k\}^*$ is said be in normal form if the set of mlgs $\{\phi_1, \ldots, \phi_k\}$ is in normal form.

A wlg $\psi = e_1 \bullet \cdots \bullet e_\ell$ is said to be *normal* if

- $e_1, \ldots, e_\ell$ are normal,
- $e_i \bullet e_{i+1} \not\subseteq e_i$ for each $i : 1 \leq i < \ell$, and
- $e_i \bullet e_{i+1} \not\subseteq e_{i+1}$, for each $i : 1 \leq i < \ell$.

A set of wlgs $\{\psi_1, \cdots, \psi_m\}$ is said to be *normal* if $\psi_1, \ldots, \psi_m$ are normal and $\psi_i \not\subseteq \psi_j$ for each $i, j : 1 \leq i \neq j \leq m$.

For each wlg ψ, there is a unique *normal* wlg ψ' such that $L(\psi) = L(\psi')$. We can derive ψ' from ψ using normalisation, checking entailment for mlgs and the entailment algorithm for wlgs described below. Normal form for sets of wlgs can be defined in a similar manner to mlgs. We can transform a set of wlgs $\Psi = \{\psi_1, \cdots, \psi_m\}$ into normal form using the normalization procedure above, and by eliminating redundant wlgs using the entailment algorithm below.

From now on, (sets of) wlgs will always be reduced to a normal form.

Entailment Now, we extend the algorithm for checking entailment of mlgs to check entailment of wlgs of the form $\psi \subseteq \psi'$.

First, we extend $\subseteq$ such that

- $\phi \subseteq \{\phi_1, \ldots, \phi_k\}^*$ if $\phi \subseteq \{\phi_1, \cdots, \phi_k\}$.
- $\{\phi_1, \cdots, \phi_k\}^* \subseteq \left\{\phi'_1, \cdots, \phi'_{k'}\right\}^*$ if $\{\phi_1, \cdots, \phi_k\} \subseteq \left\{\phi'_1, \cdots, \phi'_{k'}\right\}$.

The above entailment of word expressions can be computed using the entailment algorithm for multisets.

Entailment of wlgs is very similar to the entailment of mlgs. But, concatenation is not commutative. Therefore, given two non-empty wlgs $\psi_1 = e_1 \bullet \psi'_1$ and $\psi_2 = e_2 \bullet \psi'_2$, we have $\psi_1 \subseteq \psi_2$ iff one of the following holds.

- $e_1 \not\subseteq e_2$ and $\psi_1 \subseteq \psi'_2$.
- $e_1 \subseteq e_2$, e_2 is an atomic expression and $\psi'_1 \subseteq \psi'_2$
- $e_1 \subseteq e_2$, e_2 is a star expression, $\psi'_1 \subseteq \psi_2$.

For sets of wlgs, we use a lemma similar to Lemma 3.

Lemma 4. *For wlgs $\psi, \psi_1, \ldots, \psi_m$, if $\psi \subseteq \{\psi_1, \cdots, \psi_m\}$, then $\psi \subseteq \psi_i$ for some $i \in \{1, \ldots, m\}$.*

Proof. See Appendix. □

From Theorem 2, Lemma 4 and the above algorithm for computing entailment of wlgs, we conclude that

Theorem 4. *Entailment of wlgs can be computed in quadratic time and entailment of a set of wlgs can be computed in cubic time.*

Example 5. Consider the same alphabet A and mlgs $\phi_1 = \{a, b\}^{\circledast} + c$ and $\phi_2 = b + c$. Consider a wlg $\psi_1 = \{\phi_2\}^* \bullet \phi_1$. Example of a word in $L(\psi_1)$ is $[b, c] \bullet [b] \bullet \left[a^3\right]$. Consider a wlg $\psi_2 = \{\phi_1\}^* \bullet \phi_2$. Example of a word in $L(\psi_2)$ is $\left[a^2, b^3, c\right] \bullet \left[a^3, c\right] \bullet [b, c]$. Notice that $\psi_1 \subseteq \psi_2$, and $\psi_2 \not\subseteq \psi_1$. Figure 3.5 graphically describes ψ_1 and ψ_2.

*
b
c
●
⊛
a
b
c
*
⊛
a
b
c
●
b
c

(a)

(b)

Fig. 3.5: wlgs (a) ψ_1. (b) ψ_2.

3.3 Region Generators

A *region generator* θ is a triple $(\phi_0, \psi, \phi_{max})$ where ϕ_0 is an mlg over $P \times \{0, \ldots, max\}$, ψ is a wlg over $P \times \{0, \ldots, max - 1\}$, and ϕ_{max} is an mlg over P. The language $L(\theta)$ contains exactly each region of the form (b_0, w, b_{max}) where $b_0 \in L(\phi_0)$, $w \in L(\psi)$, and $b_{max} \in L(\phi_{max})$.

For a region generator θ, we define $[\![\theta]\!]^{\downarrow}$ to be $\cup_{\mathcal{R} \in L(\theta)} [\![\mathcal{R}]\!]$. In other words, a region generator θ:

- defines a language $L(\theta)$ of regions; and
- denotes a set of markings, namely all markings which belong to the denotation $[\![\mathcal{R}]\!]$ for some region $\mathcal{R} \in L(\theta)$.

A finite set $\Theta = \{\theta_1, \ldots, \theta_m\}$ of region generators is interpreted as the union of its elements, i.e, $[\![\Theta]\!]^{\downarrow} = \bigcup_{1 \le i \le m} [\![\theta_i]\!]^{\downarrow}$.

Given a marking M and a region generator θ, it is straightforward to check whether $M \in [\![\theta]\!]^{\downarrow}$ from the definition of $[\![\mathcal{R}]\!]$ and $[\![\theta]\!]^{\downarrow}$.

Here, we recall that in Section 2, we showed how to decide the entailment $\le^r$ on regions.

By Theorem 1 and Theorem 3 it follows that for each set $\mathbf{R}$ of regions which is downward closed with respect to $\leq^r$, there is a finite set of region generators Θ such that $L(\Theta) = \mathbf{R}$. Recall that $\mathcal{R}_1 \leq^r \mathcal{R}_2$ implies that for each $M_1 \in [\![\mathcal{R}_1]\!]$ and $M_2 \in [\![\mathcal{R}_2]\!]$, $M_1 \preceq M_2$.

From this we get the following.

Theorem 5. *For each set* M *of markings which is downward closed with respect to* $\preceq$ *there is a finite set of region generators* Θ *such that* $\mathsf{M} = [\![\Theta]\!]^{\downarrow}$.

Entailment: We observe that if $\theta_1 = \left(\phi_0^1, \psi^1, \phi_{max}^1\right)$ and $\theta_2 = \left(\phi_0^2, \psi^2, \phi_{max}^2\right)$ then $\theta_1 \sqsubseteq \theta_2$ iff $\phi_0^1 \sqsubseteq \phi_0^2$, $\psi^1 \sqsubseteq \psi^2$, and $\phi_{max}^1 \sqsubseteq \phi_{max}^2$. In other words entailment between region generators can be computed by checking entailment between the individual elements. Notice that $\theta_1 \sqsubseteq \theta_2$ means for each $M_1 \in [\![\theta_1]\!]^{\downarrow}$ and $M_2 \in [\![\theta_2]\!]^{\downarrow}$, $M_1 \preceq M_2$.

Example 6. Consider again the TPN in Figure 2.1 with $max = 7$. Examples of mlgs over $\{Q, R, S\} \times \{0, \dots, 7\}$ are $R(2)$, $S(2)$, $\{S(6), R(1)\}^{\circledast} + S(5)$, etc. $S(2) \bullet \left\{\{S(6), R(1)\}^{\circledast} + S(5)\right\}^{*}$ is an example of a wlg over $\{Q, R, S\} \times \{0, \dots, 7\}$ and $\{Q\}^{\circledast}$ is an mlg over $\{Q, R, S\}$. Finally, an example of region generator is given by $\theta = \left(R(2),\ S(2) \bullet \left\{\{S(6), R(1)\}^{\circledast} + S(5)\right\}^{*},\ \{Q\}^{\circledast}\right)$. Figure 3.6(a) shows the region generator graphically and Figure 3.6(b) shows an example of a region in the language of the region generator in Figure 3.6(a). Notice that the markings in $[\![\theta]\!]^{\downarrow}$ can have arbitrarily many tokens in places R (with age $x : 1 < x < 2$) and S (with age $y : 5 < y < 7$).

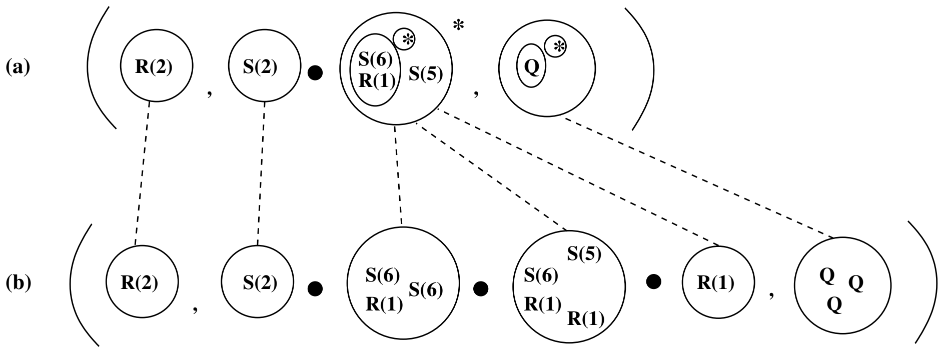


Fig. 3.6: (a) Region Generator θ. (b) A region $\mathcal{R} \in L(\theta)$.

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 Θ_{init} characterizing the set M_{init} of initial markings, and a set M_{fin} of final markings and tries to answer whether $[\![\Theta_{init}]\!]^{\downarrow} \cap \mathsf{M}_{fin} \uparrow = \emptyset$. The algorithm computes the sequence $\Theta_0, \Theta_1, \ldots$ of sets of region generators such that $\Theta_{i+1} = \Theta_i \cup succ(\Theta_i)$ with $\Theta_0 = \Theta_{init}$. If $[\![\Theta_i]\!]^{\downarrow} \cap \mathsf{M}_{fin} \uparrow \neq \emptyset$ (amounts to checking membership of elements of M_{fin} in $[\![\Theta]\!]^{\downarrow}$), or if $\Theta_{i+1} = \Theta_i$, then the procedure is terminated. We define $succ(\Theta)$ to be $Post_{Time}(\Theta) \ \cup \ \bigcup_{t \in T}(Post_t(\Theta) \cup Step_t(\Theta))$. $Post_{Time}$ 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 θ_1, θ_2 in a set of region generators such that $\theta_1 \subseteq \theta_2$, we remove θ_1 from the set.

Even if we know by Theorem 5 that there is a finite set Θ of region generators such that $Reach([\![\Theta_{init}]\!]^{\downarrow}) = [\![\Theta]\!]^{\downarrow}$, the following holds due to undecidability of structural termination for TPNs (shown in Mahata [2005]).

THEOREM 6. *Given a region generator θ_{init} we cannot in general compute a set Θ of region generators such that $Reach([\![\Theta_{init}]\!]^{\downarrow}) = [\![\Theta]\!]^{\downarrow}$.*

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

5. *Post*-Image of a Region Generator

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

5.1 Timed Post-image

To give the intuition about computing the post-image of a region generator with respect to timed transitions, first we show how to compute post-images of regions with respect to timed transitions.

5.1.1 $Post_{Time}$ for regions

We define $Post_{Time}$ such that it corresponds to letting time pass.

We compute the post-image of a region $\mathcal{R}$ with respect to time as a finite set of regions such that $[\![Post_{Time}(\mathcal{R})]\!] = \{M' \mid \exists M \in [\![\mathcal{R}]\!].\ M \longrightarrow_{Time} M'\}$. For a set $\mathbf{R}$ of regions $Post_{Time}(\mathbf{R}) = \bigcup_{\mathcal{R} \in \mathbf{R}} Post_{Time}(\mathcal{R})$.

First, we define a function *Rotate* such that given an input region $\mathcal{R}$, $Rotate(\mathcal{R})$ returns a region as described in the following. Later we use *Rotate* to define $Post_{Time}$.

Consider a marking M and a region $\mathcal{R} = (b_0, w, b_{max})$ such that $M \models \mathcal{R}$. Three cases are possible :

(1) If $b_0 = \epsilon$, 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 $\mathit{stable}(M)$. 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 $\mathcal{R}$. We call $\mathit{stable}(M)$ the *stable period* of M.

Suppose that time passes by an amount $\delta \in \mathit{stable}(M)$. If $M \longrightarrow_{T=\delta} M_1$ then $M_1 \models \mathcal{R}$, 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. This case does not yield a new marking by letting time pass.

Next consider $\delta = 1 - x$. 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 new marking M_2, where $M_2 \models \mathit{Rotate}(\mathcal{R})$ and $\mathit{Rotate}(\mathcal{R})$ 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 $b_0 \neq \epsilon$, 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 $\mathit{stable}(M)$ here is the point interval $[0 : 0]$. Suppose that we let time pass by an amount $\delta : 0 < \delta < 1 - x$, 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 $M \longrightarrow_{T=\delta} M_1$ then $M_1 \models \mathit{Rotate}(\mathcal{R})$ where $\mathit{Rotate}(\mathcal{R})$ is of the form $(\epsilon, \mathit{young} \bullet w, b_{max} + \mathit{old})$. Here, *young* and *old* are sub-multisets of b_0 such that $\mathit{young}(p(n)) = b_0(p(n))$ if $n < \mathit{max}$, and $\mathit{old}(p) = b_0(p(\mathit{max}))$, where $p(n) \in b_0$. 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 $b_0 = \epsilon, w = \epsilon$, all tokens have age greater than max. Now, if we let time pass by any amount $\delta \geq 0$ and $M \longrightarrow_{T=\delta} M_1$, then $M_1 \models \mathcal{R}$. When all tokens reach age of max, aging of tokens becomes irrelevant. This case yields only a marking which is equivalent to M with respect to $\equiv$.

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 the first two cases above correspond to "rotating" the multisets in b_0 and w, sometimes also moving them to b_{max}.

We define *Rotate** to be the reflexive transitive closure of *Rotate*. It computes the set of all regions which we can generate by letting time pass by any amount.

In *Rotate*$^*(\mathcal{R})$, we apply *Rotate* to each new region generated, except when a region is of the form (ϵ, ϵ, b). It is straightforward to verify that such a region will be eventually generated (by increasing the age of the tokens, all tokens will eventually become old). This gives us the following lemma.

Lemma 5. *Rotate** *is effectively constructible and* $Post_{Time} = Rotate^*$.

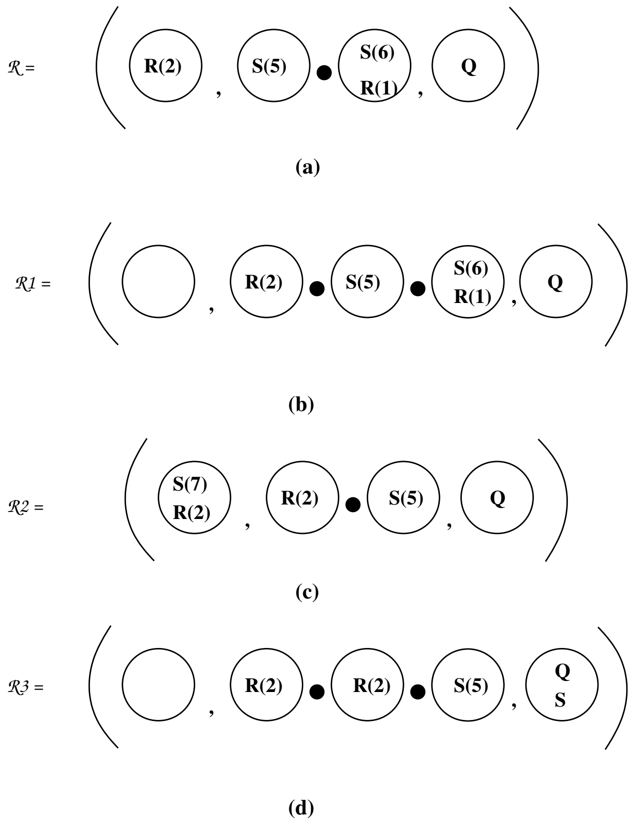


Fig. 5.7: A few regions in $Post_{Time}(\mathcal{R})$

Example 7. For the TPN in Figure 2.1, $max = 7$, consider a region $\mathcal{R}$ in Figure 5.7(a). $Post_{Time}(\mathcal{R})$ computes a number of regions. We show first three of them in Figure 5.7(b), (c) and (d). Consider the following markings $M = [R(2.0), S(5.5), R(1.7), S(6.7), Q(8.9)]$, $M_1 = [R(2.1), S(5.6), R(1.8), S(6.8), Q(9.0)]$, $M_2 = [R(2.3), S(5.8), R(2.0), S(7.0), Q(9.2)]$ and $M_3 = [R(2.4), S(5.9), R(2.1), S(7.1), Q(9.3)]$. Now, $M \longrightarrow_{0.1} M_1 \longrightarrow_{0.2} M_2 \longrightarrow_{0.1} M_3$ and $M \models \mathcal{R}$, $M_1 \models \mathcal{R}_1$, $M_2 \models \mathcal{R}_2$ and $M_3 \models \mathcal{R}_3$.

5.1.2 $Post_{Time}$ for region generators

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

First, we introduce some notations. Let ϕ 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 ϕ in this form. We define $\sharp\phi$ to be the pair (b,b') where $b = [a_1,\ldots,a_k]$ and $b' = [a_{k+1},\ldots,a_{k+\ell}]$.

Let ϕ be an mlg over $P \times \{0,\ldots,max\}$ with $\sharp\phi = (b,b')$. We define $young(\phi)$ and $old(\phi)$ to be mlgs over $P \times \{0,\ldots,max-1\}$ and P respectively such that the following holds: let $\sharp young(\phi) = \left(b_1, b_1'\right)$ and $\sharp old(\phi) = \left(b_2, b_2'\right)$ 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 ϕ, we obtain an mlg given by $young(\phi)$ which characterizes tokens younger than max and an mlg $old(\phi)$ which characterizes tokens older than max.

Let ϕ 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 ϕ^{+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 ϕ by $p(n+1)$.

We are now ready to define the function $Post_{Time}(\theta_{in})$ for some input region generator θ_{in}. We start from θ_{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 θ 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 $Post_{Time}(\theta_{in}) = V$. Depending on the form of θ, we update W and V according to one of the following cases.

- If θ is of the form $(\phi_0, \psi, \phi_{max})$, where $\phi_0 \neq \epsilon$. We add a region generator $(\epsilon, young(\phi_0) \bullet \psi, \phi_{max} + old(\phi_0))$ to W. This step corresponds to one rotation according to case 2 in the computation of *Rotate*.
- If θ is of the form $(\epsilon, \psi \bullet \phi, \phi_{max})$. Here the last element in the second component of the region generator is an atomic expression (an mlg). We add the region generator $\left(\phi^{+1}, \psi, \phi_{max}\right)$ to W. This step corresponds to one rotation according to case 1 for computation of *Rotate*.
- If θ is of the form $(\epsilon, \psi \bullet \{\phi_1,\ldots,\phi_k\}^*, \phi_{max})$. 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 $\{\phi_1,\ldots,\phi_k\}^*$ now form an unbounded sequence with strictly increasing frac-

tional parts. We add

$$\left(\phi_i^{+1}, \left\{young(\phi_1^{+1}), \ldots, young(\phi_k^{+1})\right\}^* \bullet \psi \ \bullet \ \{\phi_1, \ldots, \phi_k\}^*, \phi_{max} + Old^{\circledast}\right)$$

to V, and

$$\left(\phi_i^{+1}, \left\{young(\phi_1^{+1}), \ldots, young(\phi_k^{+1})\right\}^* \bullet \psi, \phi_{max} + Old^{\circledast}\right)$$

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 $\left\{old(\phi_1^{+1}), \ldots, old(\phi_k^{+1})\right\}$. This step corresponds to performing a sequence of rotations of the forms of case 1 and case 2 together for *Rotate*.

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 θ is of the form $(\epsilon, \epsilon, \phi_{max})$, 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 $(\epsilon, \epsilon, \phi_{max})$).

Example 8. For the TPN in Figure 2.1, where $max = 7$. Consider an input region generator $\theta = (Q(7) + R(6), \{S(6) + Q(5)\}^*, \epsilon)$ in Figure 5.8(a). We show a few region generators computed by $Post_{Time}(\theta)$ starting from θ in Figure 5.8(b) to Figure 5.8(g). Notice the rotation of first and second part of the region generators and sometimes moving to third part of the region generator, corresponding to the 'rotation' described in $Post_{Time}$ for regions. Also, notice that the region generator in (c) and (f) correspond to several 'rotations' of regions in their languages. Furthermore, we need to apply normalisation after computing $Post_{Time}$, since the region generator in Figure 5.8(f) is not in normal form. Also, observe that the region generator in Figure 5.8(g) is included in the region generator in Figure 5.8(f) and the former one will be removed during the forward analysis from the working set of the region generators.

5.2 *Discrete Post-image*

First we show how to compute the post-image of a region with respect to a discrete transition.

5.2.1 *$Post_t$ for regions*

We define $Post_{Disc}$ such that it corresponds to firing discrete transitions. $Post_{Disc}$ is computed as the union of $Post_t$ for all transitions t in the TPN where $Post_t$ characterizes the effect of running t once.

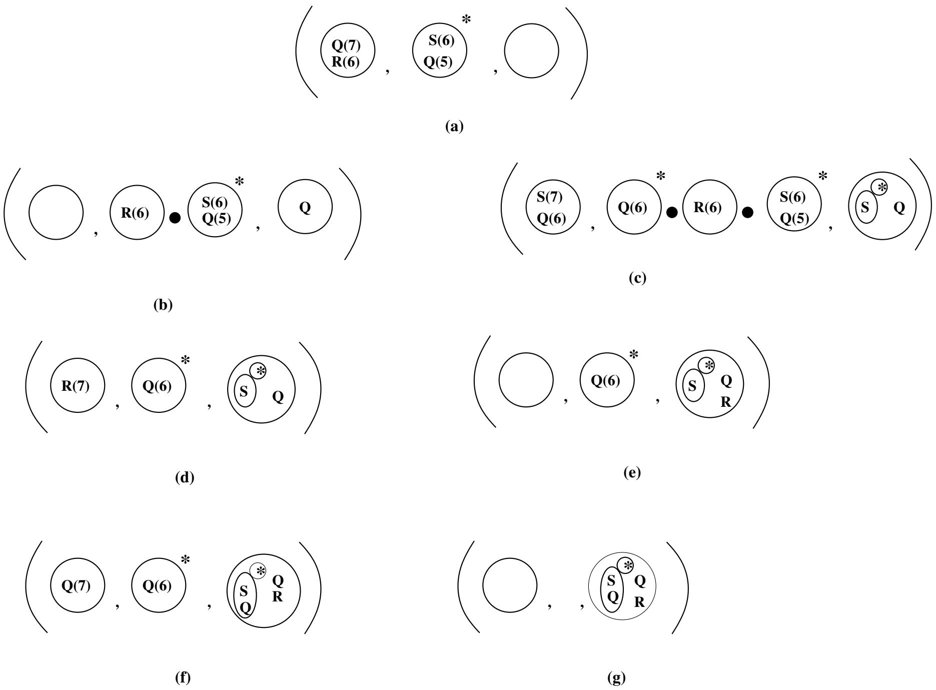


Fig. 5.8: Given θ in (a), (b), . . . ,(g) shows the region generators computed by $Post_{Time}(\theta)$.

For an input region $\mathcal{R}$, we shall characterize the set of all markings which can be reached from a marking in $[\![\mathcal{R}]\!]$ through execution of transition t. We shall compute $Post_t(\mathcal{R})$ as a finite set of regions such that $[\![Post_t(\mathcal{R})]\!] = \{M' \mid \exists M \in [\![\mathcal{R}]\!].\ M \longrightarrow_t M'\}$. For a set $\mathbf{R}$ of regions $Post_t(\mathbf{R}) = \bigcup_{\mathcal{R}\in\mathbf{R}} Post_t(\mathcal{R})$.

Let $\mathcal{R} = (b_0, w, b_{max})$. To give an algorithm for $Post_t$, we need to define an *addition* and a *subtraction* operation for regions.

An addition (subtraction) corresponds to adding (removing) a token in a certain age interval. Let $\mathcal{I}$ be an interval of the form $[w, z)$ and let $\mathcal{R}$ be a region of the form $\mathcal{R} = (b_0, b_1 \bullet \ldots \bullet b_m, b_{m+1})$. We define the *addition* $\mathcal{R} \oplus p(\mathcal{I})$ as a set of regions (the addition of other types of intervals can be defined in a similar manner).

We define $\mathcal{R} \oplus p(\mathcal{I})$ to be the union of the following four sets:

(1) A set containing $(b_0 + [p(n)], b_1 \bullet \ldots \bullet b_m, b_{m+1})$, for each $n : w \leq n < z$. This corresponds to adding tokens with zero fractional parts.

(2) A set containing regions $(b_0, b_1 \bullet \ldots \bullet b'_i \bullet \ldots \bullet b_m, b_{m+1})$, where $1 \leq i \leq m$ and $b'_i = b_i + [p(n)]$, for each n $w \leq n < z$, $n < max$. Elements added according to this case corresponds to adding a token with a fractional part equal to that of some other token.

(3) A set containing regions $(b_0, b_1 \bullet \ldots \bullet b_i \bullet [p(n)] \bullet \ldots \bullet b_m, b_{m+1})$, where n satisfies the same conditions as in 2. In this case, the fractional part differs from all other tokens.

(4) A singleton set containing $(b_0, b_1 \bullet \ldots \bullet b_m, b_{m+1} + [p])$ if $z = \infty$.

Given $\mathcal{R}$ and $\mathcal{I}$ of the above forms, we define the subtraction $\mathcal{R} \ominus p(\mathcal{I})$ as the union of the following sets:

(1) A singleton set containing $(b_0 - [p(n)], b_1 \bullet \ldots \bullet b_m, b_{m+1})$, for each $n : w \leq n < z$. This corresponds to subtracting tokens with zero fractional parts.

(2) A set containing regions $(b_0, b_1 \bullet \ldots b'_i \ldots \bullet b_m, b_{m+1})$ where $b'_i = b_i - [p(n)]$, where $1 \leq i \leq m$ and $w \leq n < z$, $n < max$. This corresponds to subtracting tokens with non-zero fractional parts.

(3) A singleton set containing $(b_0, b_1 \bullet \ldots \bullet b_m, b_{m+1} - [p])$ in case $z = \infty$. This corresponds to removing tokens with age greater than *max*.

(4) If all the above sets are empty, then $\mathcal{R} \ominus p(\mathcal{I})$ is undefined.

We extend $\oplus, \ominus$ to sets of regions in the obvious manner.

We also extend $\oplus, \ominus$ for a set $\mathcal{A}$ of pairs of the form $p(\mathcal{I})$ as follows. $\mathcal{R} \oplus \mathcal{A} = \bigcup_{p(\mathcal{I}) \in \mathcal{A}} \mathcal{R} \oplus p(\mathcal{I})$.

Let $\mathcal{A}_{in}(t)$ be the set of input arcs given by $\{p(\mathcal{I}) \mid In(t, p) = \mathcal{I}\}$ and the set of output arcs $\mathcal{A}_{out}(t)$ be given by $\{p(\mathcal{I}) \mid Out(t, p) = \mathcal{I}\}$.

We define

$$Post_t(\mathcal{R}) = (\mathcal{R} \ominus \mathcal{A}_{in}(t)) \oplus \mathcal{A}_{out}(t)$$

.

Example 9. For the TPN in Figure 2.1, consider a marking $M = [Q(3.5)]$ and a region $\mathcal{R} = (\epsilon, Q(3), \epsilon)$. $M \models \mathcal{R}$. Consider the transition t_2. We show the regions computed by $Post_t$ in Figure 5.9. Since, $\mathcal{R} \ominus [Q(3.5)] = (\epsilon, \epsilon, \epsilon)$, we show the result of $(\epsilon, \epsilon, \epsilon) \oplus R((0, 1)) \oplus S((1, 2)))$ by $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ which together covers all possible markings that can be created by firing t_2 from M.

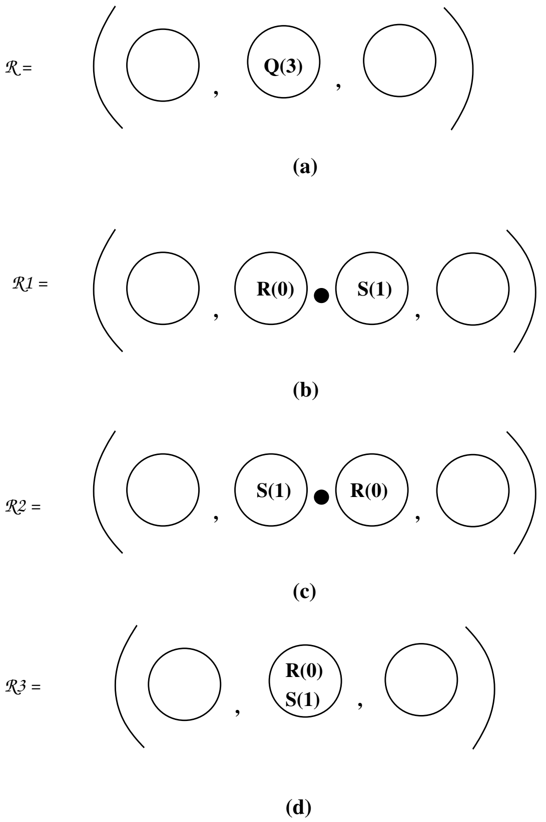
We let $Post = Post_{Time} \cup Post_{Disc}$. From the definition of $Post_{Time}, Post_t$, we get the following.

Lemma 6. *Given a region $\mathcal{R}$, $Post(\mathcal{R})$ is effectively constructible.*

5.2.2 $Post_t$ for region generators

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

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



(a)

(b)

(c)

(d)

Fig. 5.9: Regions in $Post_t(\mathcal{R})$

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 $\phi = S^{\circledast} + a_1 + \cdots + a_\ell$ 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* $\phi \oplus p(n)$ to be the mlg $\phi + p(n)$.

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

- If $p(n) \in S$, then $\phi \ominus p(n) = \phi$. Intuitively, the mlg ϕ 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 $p(n) \notin S$ and $a_i = p(n)$ for some $i : 1 \le i \le \ell$ then $\phi \ominus p(n) = S^{\circledast} + a_1 + \cdots + a_{i-1} + a_{i+1} + \cdots + a_\ell$.
- ○ 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 $\psi \oplus p(n)$ is a wlg ψ consisting of the following three sets of wlgs.

(1) For each ψ_1, ψ_2, and ϕ with $\psi = \psi_1 \bullet \phi \bullet \psi_2$, we have
$\psi_1 \bullet (\phi \oplus p(n)) \bullet \psi_2 \in (\psi \oplus p(n))$.

(2) For each ψ_1, ψ_2 and $\psi = \psi_1 \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet \psi_2$, we have for $i : 1 \le i \le k$,
$\psi_1 \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet (\phi_i \oplus p(n)) \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet \psi_2 \in (\psi \oplus p(n))$.

(3) For each ψ_1 and ψ_2 with $\psi = \psi_1 \bullet \psi_2$, we have
$\psi_1 \bullet p(n) \bullet \psi_2 \in (\psi \oplus p(n))$.

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 $\psi \ominus p(n)$, where ψ is a wlg, to be a set of wlgs, according to the following two cases.

- ○ If there is a star expression $e = \{\phi_1, \cdots, \phi_k\}^*$ containing the token we want to remove, i.e., if ψ is of the form $\psi_1 \bullet e \bullet \psi_2$, and if any of the operations $\phi_i \ominus p(n)$ is defined for $i : 1 \le i \le k$, then $\psi \ominus p(n) = \{\psi\}$.
- ○ Otherwise, the set $\psi \ominus p(n)$ contains wlgs of the form $\psi_1 \bullet \phi' \bullet \psi_2$ such that ψ is of the form $\psi_1 \bullet \phi \bullet \psi_2$ and $\phi' \in (\phi \ominus p(n))$.

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 $p(\mathtt{N})$ where $\mathtt{N} \subseteq \{0, \ldots, max\}$, e.g $\psi \ominus p(\mathtt{N}) = \{\psi \ominus p(n) \mid n \in \mathtt{N}\}$.

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 $\theta = (\phi_0, \psi, \phi_{max})$ can therefore be defined by the following operations.

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

- ○ $(\phi_0 \ominus p(\mathtt{N}), \psi, \phi_{max})$ where each $n \in \mathtt{N}$ is a natural number in the interval $\mathcal{I}$. Intuitively, if the age of the token that is removed has a zero fractional part, then $\mathtt{N}$ contains the valid choices of integral part.
- ○ $(\phi_0, \psi', \phi_{max})$ such that $\psi' \in \psi \ominus p(\mathtt{N})$, where $\mathtt{N} = \{n \mid n \in \mathbb{N} \wedge x \le n < y\}$ 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.

- ○ $(\phi_0, \psi, \phi_{max} \ominus p)$ if $\mathcal{I}$ 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 $p(\mathcal{I})$ in the obvious manner. For a set of region generators Θ, we define $\Theta \oplus p(\mathcal{I}) = \bigcup_{\theta\in\Theta} (\theta \oplus p(\mathcal{I}))$. Subtraction for a set of region generators is defined in a similar manner.

Let $\mathcal{A}_{in}(t)$ be the set of input arcs given by $\{p(\mathcal{I}) \mid In(t, p) = \mathcal{I}\}$ and the set of output arcs $\mathcal{A}_{out}(t)$ be given by $\{p(\mathcal{I}) \mid Out(t, p) = \mathcal{I}\}$.

We define,

$$Post_t(\theta) = (\theta \ominus \mathcal{A}_{in}(t)) \oplus \mathcal{A}_{out}(t)$$

Example 10. For the TPN in Figure 2.1, consider an input region generator $\theta = (\epsilon, \{R(6)\}^*, \epsilon)$ shown in Figure 5.10(a) and the transition t_1 of the TPN. We show the region generators computed by the above algorithm in Figure 5.10(b), (c) and (d). Notice that we have $\{R(6)\}^* \ominus R(6) = \{R(6)\}^*$.We show the result of $\theta \oplus Q((5, 6))$ in Figure 5.10. Notice that we normalised the resulting set of region generators after each operation.

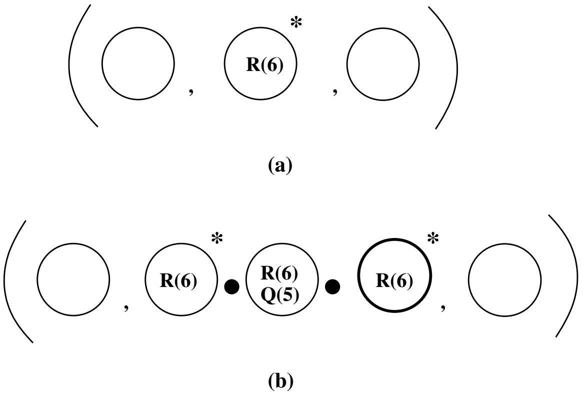


Fig. 5.10: Given θ in (a), (b) shows the region generator computed by $Post_t(\theta)$.

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 θ_{in} with respect to a transition t, then we compute a finite set $Accel_t(\theta_{in})$ of region generators such that $[\![Accel_t(\theta_{in})]\!]^{\downarrow} = \left\{M' \mid \exists M \in [\![\theta_{in}]\!]^{\downarrow}.\ M(\longrightarrow_{Time} \cup \longrightarrow_t)^* M'\right\} \downarrow$. We shall not compute the set $Accel_t(\theta_{in})$ in a single step. Instead, we will present a procedure $Step_t$ with the following property: for each region generator θ_{in} there is an $n \geq 0$ such that $Accel_t(\theta) = \bigcup_{0 \leq i \leq n} (Post_{Time} \circ Step_t)^i(\theta)$. In other words, the set $Accel_t(\theta)$ will be fully generated through a finite number of applications of $Post_{Time}$ followed by $Step_t$. Since the reachability algorithm of Section 4 computes both $Post_{Time}$ and $Step_t$ during each iteration, we are guaranteed that all region generators in $Accel_t(\theta_{in})$ will eventually be produced.

To define $Step_t$ we need some preliminary definitions.

For a word atomic expression (mlg) $\phi = \{a_1, \ldots, a_k\}^{\circledast} + a_{k+1} + \cdots + a_{k+\ell}$, we define $sym(\phi)$ as the set of symbols given by $\{a_1, \ldots, a_{k+\ell}\}$. For a word star expression $e = \{\phi_1, \ldots, \phi_k\}^*$, $sym(e) = \bigcup_i sym(\phi_i)$ for $i : 1 \leq i \leq k$.

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

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

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

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 [Abdulla *et al.* 2003] 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 θ.

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

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

- If $\psi = \epsilon$, then $\psi \uplus p(n) = \left\{\{p(n)\}^{\circledast}\right\}^*$.
- If $\psi = \phi \bullet \psi'$, then $\psi \uplus p(n) = \left\{\{p(n)\}^{\circledast}\right\}^* \bullet (\phi \uplus p(n)) \bullet (\psi' \uplus p(n))$
- If $\psi = \{\phi_1, \cdots, \phi_n\}^* \bullet \psi'$, then $\psi \uplus p(n) = \{\phi_1 \uplus p(n), \cdots, \phi_n \uplus p(n)\}^* \bullet (\psi' \uplus p(n))$

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

Given a region generator $\theta = (\phi_0, \psi, \phi_{max})$ and a pair $p(\mathcal{I})$ where $\mathcal{I} = (x, y)$, we define

$\theta \uplus p(\mathcal{I}) = (\phi_0 + S_1{}^{\circledast},\ \psi \uplus S_2,\ \phi_{max} + \mathsf{p}_{\mathsf{max}}{}^{\circledast})$ where

- $S_1 = \{p(n) \mid n \in \mathbb{N}\ \wedge\ x < n < y\}$.
- $S_2 = \{p(n) \mid n \in \mathbb{N}\ \wedge\ x \leq n < y\}$.
- $\mathsf{p}_{\mathsf{max}} = \{p\}$ if $y = \infty$, $\mathsf{p}_{\mathsf{max}} = \emptyset$ otherwise.

For a set of pairs, $\mathcal{A} = \{p_1(\mathcal{I}_1), \cdots, p_k(\mathcal{I}_k)\}$, we define $\theta \uplus \mathcal{A} = \theta \uplus p_1(\mathcal{I}_1) \uplus \cdots \uplus p_k(\mathcal{I}_k)$.

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 $\theta = (\phi_0, \psi, \phi_{max})$ 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 ϕ_0.
- there is an integer n such that $x \leq n < y$ and $p(n)$ is a $\circledast -$ *symbol* or a $* -$ *symbol* in ψ.
- p is a $\circledast -$ *symbol* in ϕ_{max} and $y = \infty$.

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 hold.

(1) Given $In(t,p) = \mathcal{I}$, $\theta\ \ominus\ p(\mathcal{I}) \neq \emptyset$. 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.

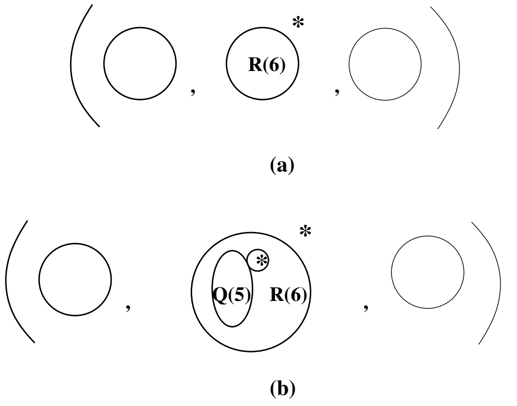
We accelerate if each input place of t is a Type 1 place or a Type 2 place.

Acceleration: Let $\mathcal{A}_{in}(t), \mathcal{A}_{out}(t)$ be the set of input and output arcs as defined in Section 5. Now, given a region generator θ, 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 $\Theta = (\theta\ \ominus\ \mathcal{A}_{in}(t))\ \oplus\ T_2$ where $T_2 = \{p(\mathcal{I}) \mid p \text{ IS OF TYPE } 2 \wedge p(\mathcal{I}) \in \mathcal{A}_{out}(t)\}$ is the set of output arcs from Type 2 places.
- Next, we accelerate addition for each region generator in Θ 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

$$Step_t(\theta) = \bigcup_{\theta' \in \Theta} \theta' \uplus (\mathcal{A}_{out}(t) \setminus T_2)$$

Example 11. Consider the TPN in Figure 2.1 and the region generator θ in Figure 6.11(a). Figure 6.11(b) illustrates the region generator computed by the acceleration algorithm from θ with respect to transition a. Notice that all region generators generated by $Post_t$ is entailed by the region generator in Figure 6.11(b).



(a)

(b)

Fig. 6.11: Given θ in (a), (b) shows the region generator computed by $Step_t(\theta)$.

Theorem 7. *If the acceleration criterion holds from a region generator θ with respect to a transition t in a TPN, there is an $n \geq 0$ such that $Accel_t(\theta) = \bigcup_{0 \leq i \leq n} (Post_{Time} \circ Step_t)^i(\theta)$.*

Proof. See Appendix. □

7. Experimental Results

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

7.1 Fischer's Protocol

First we describe a parameterized version of Fischer's protocol. The purpose of the protocol is to guarantee mutual exclusion in a concurrent system consisting of an arbitrary number of processes. The example was suggested by [Schneider *et al.* 1992].

The protocol consists of an arbitrary number of processes, each running the code graphically described in Figure 7.12. 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

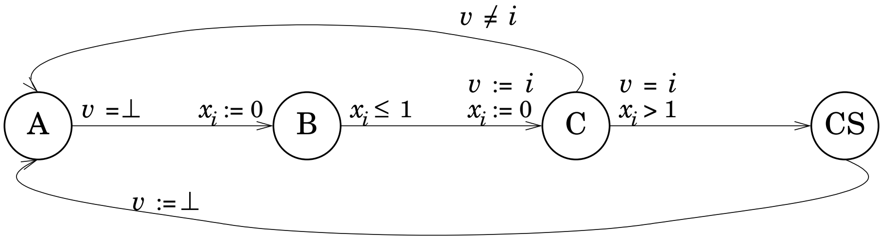


Fig. 7.12: Fischer's Protocol for Mutual Exclusion

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.

All processes start in state A. If the value of the shared variable is $\perp$, a process wishing to enter the critical section can proceed to state B and reset its local clock. From state B, the process can proceed to state C within one time unit or get stuck in B forever. When making the transition from B to C, the process resets its local clock and sets the value of the shared variable to its own index. The process now has to wait in state C for more than one time unit, a period of time which is strictly greater than the one used in the timeout of state B. If the value of the shared variable is still the index of the process, the process may enter the critical section, otherwise it may return to state A and start over again. When exiting the critical section, the process resets the shared variable to $\perp$.

[Abdulla and Nylén 2001] gives a model of the protocol in our TPN formalism. The processes running the protocol are modeled by tokens in the places A, B, C, CS, $A!$, $B!$, $C!$ and $CS!$. 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 $\perp$. A token in place *udf* means that the variable $v = \perp$ and an absence of a token in *udf* means that some process i has its id assigned to the variable.

A straightforward translation of the description in Figure 7.12 yields the timed Petri net model in Figure 7.13. q is used to denote an arbitrary process state. We illustrate translation of two transitions in Figure 7.12 by the transitions in of the TPN model in Figure 7.13 in the following. Translations of other transitions can be explained in a similar manner.

In Figure 7.12, a process in state A changes its state to B if the variable value is undefined. Furthermore, it resets its clock. This is translated to the transition *initiate* in the TPN model. The transition *initiate* is fired if there is a token in place A and a token in place udf (denoting that the variable is undefined). Firing of the transition removes the token from the place A, adds a token with age 0 to place B (corresponds to resetting the clock and changing state to B in Figure 7.12) and leaves the variable undefined by returning a token in place udf.

Secondly, in Figure 7.12, a process in state B changes its state to C if its clock value is less than 1 and it assigns its own process id i to the variable v and resets its

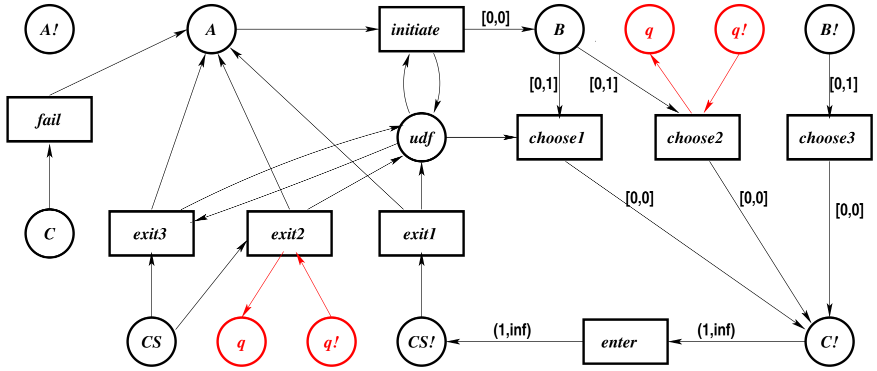


Fig. 7.13: TPN model of Fischer's Protocol for Mutual Exclusion

clock. This transition is translated to three transitions *choose1*, *choose2*, *choose3* in the TPN model. There are 3 cases.

$v = \bot$**.** If there is a token in *udf* (denoting $v = \bot$) and a token in B with age less than 1 (modelling a process in state B), firing transition *choose1* puts a token of age 0 in $C!$ denoting that a process in C modeled by the token in $C!$ has its id assigned to the shared variable and has reset it clock.

$v = j$ ***where*** $j \neq i$**.** If there is a token in place $q!$ (i.e, some other process has its id j assigned to the shared variable) and there is a token in place B (modelling a process in state B) with clock value less than 1, we fire *choose2* and change the state of the process in $q!$ to q by removing a token from place $q!$ and adding a token to q. Also, the token from place B is moved to $C!$ and the new age of the token is 0.

$v = i$**.** If there is a token in place $B!$ (modelling a process which already has its id assigned to the shared variable), we fire the transition *choose3*, remove the token from $B!$ and add a token to $C!$ with age 0.

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 $\left(\{A(0), A(1)\}^{\circledast} + udf(0), \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A\}^{\circledast}\right)$ as the initial region generator θ_{init}. θ_{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.

7.2 Lynch and Shavit's Mutual Exclusion Protocol

[Lynch and Shavit 1992] modified Fischer's protocol in such a way that mutual exclusion property becomes time-independent. The code for each process in Lynch and Shavit's protocol is shown in Figure 7.14 and the corresponding TPN model is shown in Figure 7.15. Each process i has a local clock x_i, and a control state, which assumes values in the set $\{A', B', C', A, B, C, CS, W, X\}$ where A' is the initial state and CS is the critical section. This protocol uses an integer variable v_1 (same as v in Fischer) and an extra boolean variable v_2, shared between processes. The code for each process can be explained as in the case of Fischer.

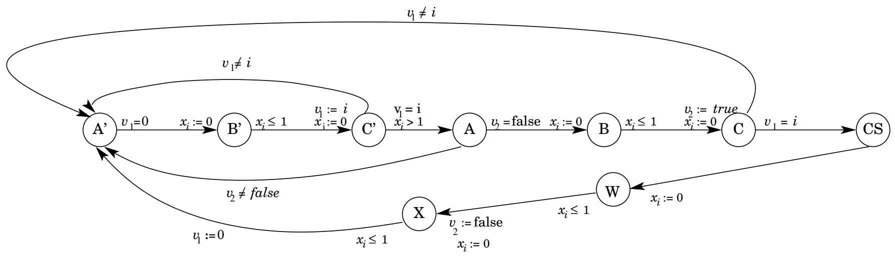


Fig. 7.14: One process running Lynch and Shavit's mutual exclusion protocol

In Figure 7.15, the processes running this protocol are modelled by tokens in the places A', B', C', A, B, C, CS, W, X, $A'!$, $B'!$, $C'!$ $A!$, $B!$, $C!$, $CS!$, $W!$ and $X!$. The places marked with ! represent that the value of the shared variable v_1 is the index of the process modelled by the token in that place. We use a place *udf* to represent that the value of the shared variable v_1 is undefined (0). We use two places $false$ and *true* to represent the variable v_2. A token in place *udf* means that the variable $v_1 = 0$ and an absence of a token in *udf* means that some process i has its id assigned to the variable. A token in place $false$ and no token in place *true* mean that the shared variable v_2 has value $false$. Shared variable v_2 with value *true* is represented in a similar manner. Also, we consider $q \in \{A', B', C', CS, A, B, C, W, X\}$ and $false' = false$. Notice that in this protocol, it is not compulsory to have a delay before entering the critical section $CS!$.

A straightforward translation of the description in Figure 7.14 yields the timed Petri net model in Figure 7.15. The specification of the bad markings is the similar to the case of Fischer's protocol. We use $\left(\{A'(0), A'(1)\}^{\circledast} + udf(0) + false(0), \left\{\{A'(0)\}^{\circledast}\right\}^{*}, \{A'\}^{\circledast}\right)$ as the initial region generator θ_{init}. θ_{init} characterizes arbitrarily many processes in A' having any clock value (age), one token in udf with age 0 and one token in $false$ with age 0 denoting that v_2 is false initially.

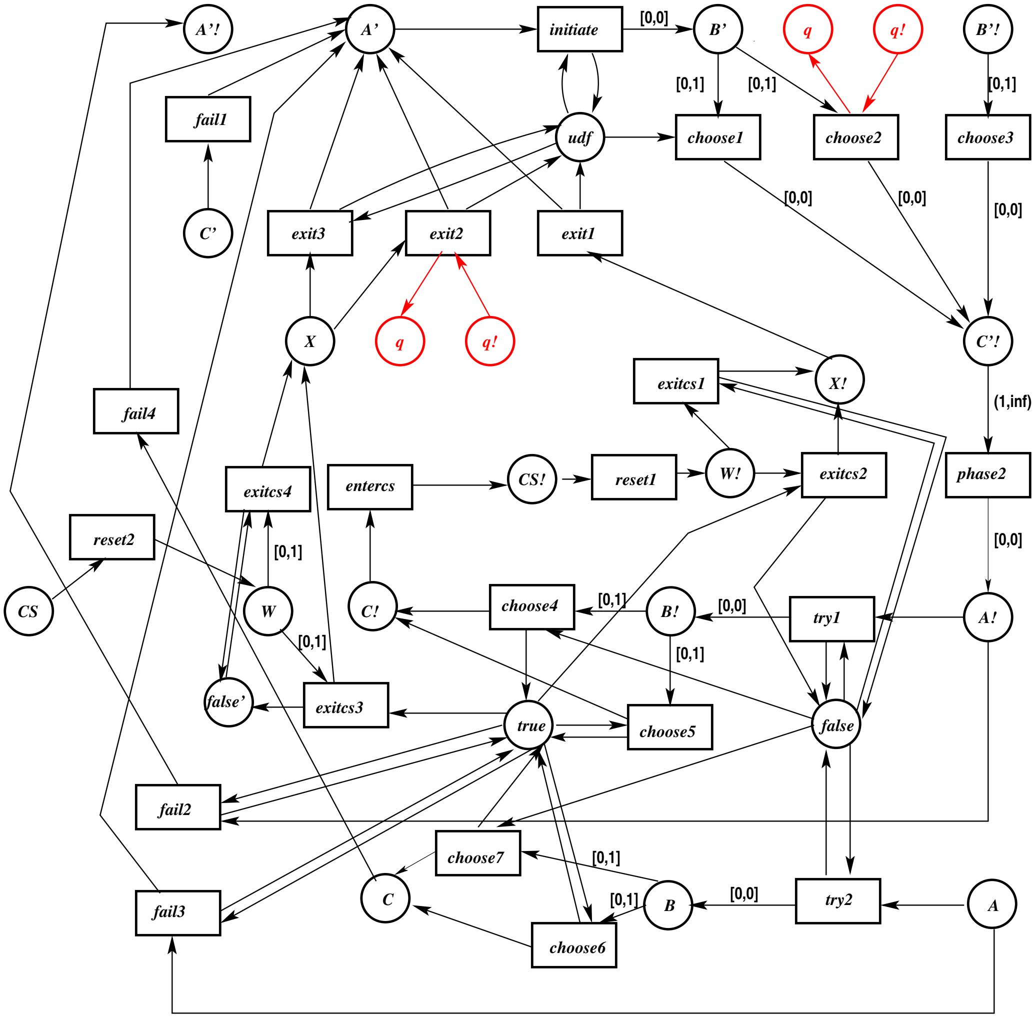


Fig. 7.15: TPN model for the parameterized version of Lynch and Shavit's protocol

7.3 Producer/Consumer System

In a traditional producer/consumer system, the producer produces items and stores them into a buffer, whereas the consumer consumes the items from the buffer. Figure 7.16 shows a timed Petri net model of the producer/consumer 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 by firing *consume* if the place *consumer_ready* has a token; firing *consume* also puts back a token in place *tmp*. A transition *get_ready* is used to move the token from *tmp* back to the place *consumer_ready* if there are still items of age 0 in *store*. To make this possible, old items (of age greater than 1) in *store* are recycled by the producer using the transition *recycle*. Firing *recycle* removes an old

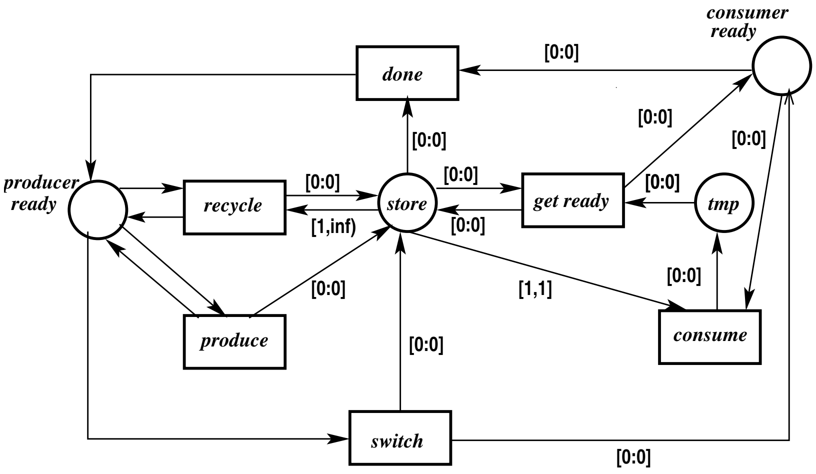


Fig. 7.16: TPN model for Producer/Consumer System

item from the *store* if *producer_ready* has a token and puts a fresh item (a token of age 0) back to *store*. The transition switch moves the control from the producer to the consumer by consuming a token from *producer_ready* and adding a token to *consumer_ready*. Also, the transition *done* switches the control back from the consumer to the producer by doing the reverse. These two transitions make sure that the items are not simultaneously accessed by the producer and the consumer.

We consider the producer/consumer system mentioned in [Nielson *et al.* 2001][2]. We use $(producer\_ready(0), \epsilon, \epsilon)$ as the initial region generator θ_{init} which characterizes a single token in place "*producer_ready*" with age 0.

7.4 Results

Our program computes the reachability set for all the protocols. The procedure fails to terminate without the use of acceleration in all the cases. It took 1.16MB memory and 2.12s to analyse Fischer's protocol, 187MB memory and 34mins to analyse Lynch and Shavit's protocol and 1.02MB memory and 1.25s to analyse producer/consumer system on a 1 GHz processor with 256 MB RAM.

7.5 Abstract Graph

Using forward analysis of a TPN, our tool also generates a graph $\mathcal{G}$ which is a finite-state abstraction of the TPN. Each state in $\mathcal{G}$ corresponds to a region generator in the reachability set. Edges of $\mathcal{G}$ are created as follows. Consider two region generators θ_1, θ_2 in the reachability set. If there is a region generator $\theta_2' \in Post_t(\theta_1)$ such that $\theta_2' \subseteq \theta_2$, then we add an edge $\theta_1 \xrightarrow{t} \theta_2$ to $\mathcal{G}$. Similarly, if there is a region

[2] [Nielson *et al.* 2001] considers a TPN model with local time in each place.

generator $\theta'_2 \in Post_{Time}(\theta_1)$ such that $\theta'_2 \subseteq \theta_2$, then we add an edge $\theta_1 \xrightarrow{\tau} \theta_2$. 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 10 states and 59 edges in the case of Fischer's protocol; 64 states and 478 edges in the case of Lynch and Shavit's protocol; and 11 states and 49 edges in the case of producer/consumer system. Furthermore, we use *The Concurrency Workbench* [Cleaveland *et al.* 1989] to minimize the abstract graphs modulo weak bisimilarity. Figure 7.17, Figure 7.18 and Figure 7.19 show the minimized finite state labelled transition systems for the above protocols.

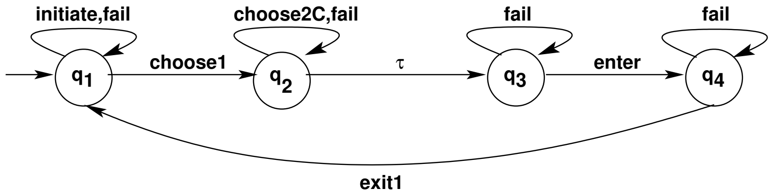


Fig. 7.17: Minimized abstract graph for Fischer's protocol.

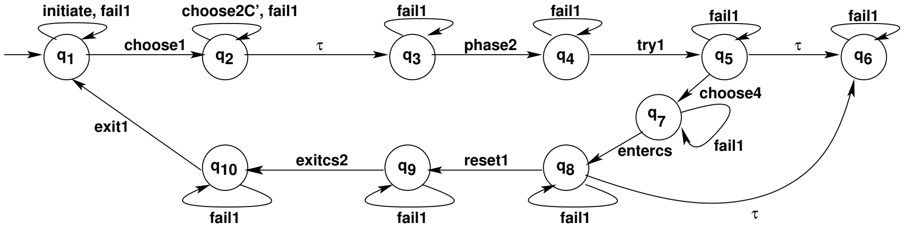


Fig. 7.18: Minimized abstract graph for Lynch and Shavit's protocol.

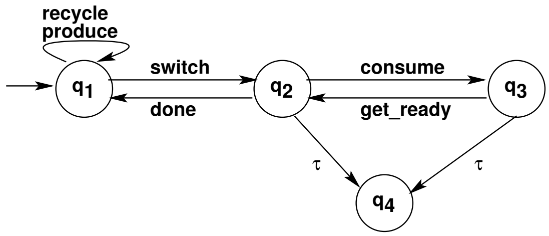


Fig. 7.19: Minimized abstract graph for producer/consumer protocol.

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) [Berthomieu and Diaz 1991]. 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 [Larsen *et al.* 1997]. Intuitively, a zone generator will correspond to a state in each minimized automaton in Figures 7.17, 7.18 and 7.19.
- 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 [Abdulla *et al.* 2000]. This would give a general theory for forward analysis of infinite state systems, in the same way the work in [Abdulla *et al.* 2000] is for backward analysis. Simple regular expressions of [Abdulla *et al.* 1998] and the region generators of this paper are examples of data structures which might be developed in a systematic manner within such a theory.

Acknowledgements

We are thankful to the anonymous reviewer for his/her valuable comments. We also thank Dr. Richard Mayr and Dr. Julien d'Orso for giving comments on some earlier versions of this paper.

References

Abdulla, P. A., Bouajjani, A., and Jonsson, B. 1998. On-the-Fly Analysis of Systems with Unbounded, Lossy FIFO Channels. In *Proc.* 10th *Int. Conf. on Computer Aided Verification*, Volume 1427 of *Lecture Notes in Computer Science*, 305–318.

Abdulla, P. A., Čerāns, K., Jonsson, B., and Yih-Kuen, T. 2000. Algorithmic Analysis of Programs with Well Quasi-Ordered Domains. *Information and Computation 160*, 109–127.

Abdulla, P. A., Deneux, J., Mahata, P., and Nylén, A. 2003. Forward Reachability Analysis of Timed Petri Nets. Tech. Report 2003-056, Dept. of Information Technology, Uppsala University, Sweden.

Abdulla, P. A. and Nylén, A. 2001. Timed Petri Nets and BQOs. In *Proc. ICATPN'2001: 22nd Int. Conf. on application and theory of Petri nets*, Volume 2075 of *Lecture Notes in Computer Science*, 53 –70.

Alur, R. and Dill, D. 1990. Automata for Modelling Real-Time Systems. In *Proc. ICALP '90*, Volume 443 of *Lecture Notes in Computer Science*, 322–335.

Bensalem, S., Lakhnech, Y., and Owre, S. 1998. Computing Abstractions of Infinite State Systems Automatically and Compositionally. In *Computer Aided Verification*, Volume 1427 of *Lecture Notes in Computer Science*. Springer-Verlag, 319–331.

Berthomieu, B. and Diaz, M. 1991. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. on Software Engineering 17*, 3, 259–273.

Boigelot, B. and Godefroid, P. 1996. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. In *Proc. 8th Int. Conf. on Computer Aided Verification*, Volume 1102 of *Lecture Notes in Computer Science*. Springer Verlag, 1–12.

Bouajjani, A. and Habermehl, P. 1997. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. In *Proc. ICALP '97, 24th International Colloquium on Automata, Lnaguages, and Programming*, Volume 1256 of *Lecture Notes in Computer Science*.

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

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

Coolahan, J. E. and Roussopoulos, N. 1983. Timing Requirements for Time Driven System Using Augmented Petri Nets. In *IEEE Transactions on Software Engineering*, Volume SE9.

Delzanno, G. and Raskin, J. F. 2000. Symbolic Representation of Upward-Closed Sets. In *Proc. TACAS '00, 6th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, Volume 1785 of *Lecture Notes in Computer Science*, 426–440.

Dickson, L. E. 1913. Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors. *Amercan Journal of Mathematics 35*, 413–422.

Finkel, A., Iyer, S. Purushothaman, and Sutre, G. 2000. Well-abstracted transition systems. In *Proc. CONCUR 2000, 11th Int. Conf. on Concurrency Theory*, 566–580.

Finkel, A., Raskin, J.-F., Samuelides, M., and Begin, L. Van. 2002. Monotonic Extensions of Petri Nets: Forward and Backward Search Revisited. In *Proc. Infinity'02*.

Ghezzi, C., Mandrioli, D., Morasca, S., and Pezzè, M. 1991. A Unified High-Level Petri net Formalism for Time-Critical Systems. *IEEE Trans. on Software Engineering 17*, 2, 160–172.

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

Higman, G. 1952. Ordering by Divisibility in Abstract Algebras. *Proc. London Math. Soc. 2*, 326–336.

Holliday, M. A. and Vernon, M. K. 1987. A Generalized Timed Petri Net Model for Performance Analysis. In *IEEE Transactions on Software Engineering*, Volume SE13.

Karp, R.M. and Miller, R.E. 1969. Parallel Program Schemata. *Journal of Computer and Systems Sciences 3*, 2 (May), 147–195.

Lakhnech, Y., Bensalem, S., Berezin, S., and Owre, S. 2001. Symbolic Techniques for Parametric Reasoning about Counter and Clock Systems. In *Proc. TACAS '01, 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, Volume 2031 of *Lecture Notes in Computer Science*. Springer Verlag.

Larsen, K.G., Pettersson, P., and Yi, W. 1997. UPPAAL in a Nutshell. *Software Tools for Technology Transfer 1*, 1-2, ?–?

Lynch, N. A. and Shavit, N. 1992. Timing-Based Mutual Exclusion. In *IEEE Real-Time Systems Symposium*, 2–11.

Mahata, Pritha. 2005. *Model Checking Parameterized Timed Systems*. PhD thesis, Dept. of Computer Systems, Uppsala University, Sweden, Uppsala, Sweden.

Merlin, P. and Farber, D.J. 1976. Recoverability of Communication Protocols - Implications of a Theoretical Study. *IEEE Trans. on Computers COM-24*, (Sept.), 1036–1043.

Nielson, M., Sassone, V., and Srba, J. 2001. Towards a Distributed Time for Petri nets. In *Proc. ICATPN'01*, Volume 2075 of *Lecture Notes in Computer Science*, 23–31.

Ramamoorthy, C. V. and Ho, G. S. 1980. Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets. In *IEEE Transactions on Software Engineering*, Volume SE6.

Razouk, R. and Phelps, C. 1985. Performance Analysis Using Timed Petri nets. In *Protocol Testing, Specification, and Verification*, 561–576.

Schneider, F. B., Bloom, B., and Marzullo, K. 1992. Putting Time Into Proof Outlines. In *Real-Time: Theory in Practice*, Volume 600 of *Lecture Notes in Computer Science*.

Vardi, M. Y. and Wolper, P. 1986. An automata-theoretic approach to automatic program verification. In *Proc. LICS'86*. IEEE Computer Society Press, 332–344.

Appendix A. Appendix - Proofs of Lemmas

Appendix A.1 Proof of Theorem 1

First, we show some auxiliary lemmas.

Lemma A.1 For a finite alphabet A and a multiset $\kappa \in A^{\circledast}$, there is a set of mlgs Φ such that a multiset $\varrho \in L(\Phi)$ iff $\kappa \not\leq^m \varrho$.

Proof. Let κ be of the form $\left[a_1^{l_1}, \ldots, a_m^{l_m}\right]$. Let e_1 be a star expression $\{b_1, \cdots, b_k\}^{\circledast}$ where $b_1, \ldots, b_k \in A \setminus \{a_1, \ldots, a_m\}$. Notice that $A = \{a_1, \ldots, a_m\}$ implies that $L(e_1) = \{\epsilon\}$. We define Φ as a set of mlgs ϕ_i of the form $e_1 + a_1^{l_1} + \cdots + a_i^{l_i - 1} + \cdots + a_m^{l_m}$ where $i : 1 \leq i \leq m$.

First we show that $\varrho \in L(\Phi) \rightarrow \kappa \not\leq^m \varrho$ by contraposition. Assume $\kappa \leq^m \varrho$. We prove that $\varrho \notin L(\phi_i)$ for each $i : 1 \leq i \leq m$. From the definition of $\leq^m$, we know that ϱ is of the form $\kappa + \vartheta$ where $\vartheta \in A^{\circledast}$. From definition of ϕ_i, $a_i^{l_i} \notin L(\phi_i)$ for each $i : 1 \leq i \leq m$. Therefore, $\kappa + \vartheta \notin L(\phi_i)$ for each i. Therefore, $\varrho \notin L(\Phi)$.

Next, we prove that $\kappa \not\leq^m \varrho \rightarrow \varrho \in L(\phi)$. Let κ' be the largest proper sub-multiset of κ which satisfies $\kappa' \leq^m \varrho$. This means that ϱ is of the form $\kappa' + \vartheta$ where ϑ is a multiset over $A \setminus \{a_1, \ldots, a_m\}$. Thus $\vartheta \in L(e_1)$. Since κ' is a proper submultiset of κ, $\kappa' \in L(a_1^{l_1} + \cdots + a_i^{l_i - 1} + \cdots + a_m^{l_m})$ where $i : 1 \leq i \leq m$. Thus, $\varrho \in L(\Phi)$. □

Lemma A.2 For set of mlgs Φ_1, Φ_2, there is a set of mlgs $\Phi_1 \cap \Phi_2$ such that $L(\Phi_1 \cap \Phi_2) = L(\Phi_1) \cap L(\Phi_2)$. Proof. First we consider the intersection of mlgs ϕ_1, ϕ_2.

In case, ϕ_1, ϕ_2 are atomic expressions, we have

- ○ if both are atomic expressions, then either of the following holds.
  (1) if $\phi_1 = \phi_2 = a$, then $\phi_1 \cap \phi_2 = a$.
  (2) $\phi_1 \cap \phi_2 = \epsilon$, otherwise.
- ○ If one of them is a star expression $\{a_1, \cdots, a_l\}^{\circledast}$ and the other one is a, then $\phi_1 \cap \phi_2 = a$ if $a \in \{a_1, \ldots, a_m\}$ for $a, a_1, \ldots, a_m \in A$. Otherwise, $\phi_1 \cap \phi_2 = \epsilon$.
- ○ If both of them are star expressions, i.e, $\phi_1 = \{a_1, \cdots, a_m\}^{\circledast}$ and $\phi_2 = \{b_1, \cdots, b_n\}^{\circledast}$, then $\phi_1 \cap \ \phi_2 = \{c_1, \cdots, c_k\}^{\circledast}$ where $\{c_1, \ldots, c_k\} = \{a_1, \ldots, a_m\} \ \cap \ \{b_1, \ldots, b_n\}$.

If ϕ_1 and ϕ_2 are mlgs, then if either of them is empty, their intersection is also empty. Suppose that we are given two non-empty mlgs $\phi_1 = e_{11} + \cdots + e_{1k}$ and $\phi_2 = e_{21} + \cdots + e_{2m}$. Define ϕ_{1i} to be the result of deleting the expression e_{1i} from ϕ_1. Define ϕ_{2j} in a similar manner. Then, $\phi_1 \cap \phi_2$ is the union of all sets of mlgs ϕ_{ij}, for $i : 1 \leq i \leq k$ and $j : 1 \leq j \leq m$, computed according to one of the following four cases.

(1) if e_{1i} and e_{2j} are atomic expressions.
$\phi_{ij} \ = \ (e_{1i} \cap e_{2j}) \ + \ (\phi_{1i} \cap \phi_{2j})$.

(2) if e_{1i} is an atomic expression and e_{2j} is a star expression.
$\phi_{ij} \ = \ (e_{1i} \cap e_{2j}) \ + \ (\phi_{1i} \cap \phi_2)$.

(3) if e_{1i} is a star expression and e_{2j} is an atomic expression.
$\phi_{ij} \ = \ (e_{1i} \cap e_{2j}) \ + \ (\phi_1 \cap \phi_{2j})$.

(4) if e_{1i} and e_{2j} are star expressions.
$\phi_{ij} \ = \ \left\{(e_{1i} \cap e_{2j}) \ + \ (\phi_{1i} \cap \phi_2) \ , \ (e_{1i} \cap e_{2j}) \ + \ (\phi_1 \cap \phi_{2j})\right\}$

Intuitively, due to commutativity of multiset addition, we intersect all pairs of expressions in two mlgs and repeat the intersection with the rest of the two mlgs. Notice that, if one of e_{1i}, e_{2j} is a star expression, (say, e_{2j}), then we consider whole of ϕ_2 as the "rest" of the mlg. Also notice that we assume that + can be distributed over sets of mlgs.

Now, if $\Phi_1 = \{\phi_1, \cdots, \phi_m\}$ and $\Phi_2 = \left\{\phi'_1, \cdots, \phi'_n\right\}$, then $\Phi_1 \cap \Phi_2 = \{\phi_{11}, \cdots, \phi_{n_1 n_2}\}$ where $\phi_{ij} = \phi_i \cap \phi'_j$ for each $i : 1 \leq i \leq m$ and $j : 1 \leq j \leq n$. □

Main proof of Theorem 1:

Finally, we assume a downward closed language L. If $L = \emptyset$, then $L = L(\Phi)$ where $\Phi = \emptyset$. Otherwise, complement of L is upward closed and can be characterized by a finite set of multisets $\{M_1, \ldots, M_n\}$ over A by Dickson's Lemma Dickson [1913]. Thus, a multiset $\varrho \in L$, if and only if $M_i \not\leq^m \varrho$ for any $i : 1 \leq i \leq n$. $M_i \in A^{\circledast}$ for each $i : 1 \leq i \leq n$ and by Lemma A.1 and Lemma A.2, it follows that there are sets of mlgs, $\Phi_1, \ldots, \Phi_n$ such that $L = L(\Phi_1) \ \cap \cdots \cap \ L(\Phi_n)$.

Appendix A.2 Proof of Lemma 3

We prove the lemma by contraposition. Assume $\phi \not\subseteq \phi'$ for any $\phi' \in \left\{\phi'_1, \ldots, \phi'_n\right\}$.

We show that there is a multiset $M \in L(\phi)$, but $M \notin L(\phi')$ for any ϕ' such that $\phi' \in \left\{\phi'_1, \ldots, \phi'_n\right\}$. Thus $M \notin L(\left\{\phi'_1, \cdots, \phi'_n\right\})$. Therefore, $\phi \not\subseteq \left\{\phi'_1, \cdots, \phi'_n\right\}$.

Let ϕ' be of the form $e'_1 + \cdots + e'_k$.

Induction hypothesis **(IH):** For a mlg $\phi = e_1 + \cdots + e_m$ with $m \geq 1$, we have $e_1 + \cdots + e_m \not\subseteq \phi' \rightarrow M_1 + \cdots + M_m \notin L(\phi')$ where multisets $M_i \in L(e_i)$ for $i : 1 \leq i \leq m$ and $M = M_1 + \ldots + M_m$.

Base case ($m = 1$):

First, we prove the claim where ϕ is an atomic expression a. In that case, we define M to be a multiset containing a singleton element a. $a \notin L(e'_i)$ for any $i : 1 \leq i \leq k$. Hence, $a \notin L(\phi')$.

Second, we prove the claim where ϕ is a star expression $\{a_1, \ldots, a_l\}^{\circledast}$. In this case, we define M such that $M(a) = k + 1$ for all $a \in \{a_1, \cdots, a_l\}$, i.e $M = \left[a_1^{k+1}, \ldots, a_l^{k+1}\right]$ and $l > 0$. We use induction on k to show that $M \notin L(\phi')$. The base case ($k = 0$) is trivial. For the induction step, we assume $k > 0$. For each $i : 1 \leq i \leq k$, assuming $\phi' = e'_i + \phi''_i$, we show the claim. There are two cases.

- ○ e'_i is atomic. By the induction hypothesis, we have that $\left[a_1^k, \ldots, a_l^k\right] \notin L(\phi''_i)$. Since e'_i is atomic (contains a singleton), $\left[a_1^{k+1}, \ldots, a_l^{k+1}\right] \notin L(\phi')$.
- ○ e'_i is star expression. We know that $\{a_1, \ldots, a_l\}^{\circledast} \not\subseteq e'_i$ (otherwise, $\phi \subseteq \phi'$ and that is contradiction). Since e'_i is a star expression and $\{a_1, \ldots, a_l\}^{\circledast} \not\subseteq e'_i$, there must be a symbol $a \in \{a_1, \ldots, a_l\}$ such that $a \notin L(e'_i)$. This implies that $[a_1, \ldots, a_l] \notin L(e'_i)$. By the induction hypothesis, we have that $\left[a_1^k, \ldots, a_l^k\right] \notin L(\phi''_i)$. This implies that $\left[a_1^{k+1}, \ldots, a_l^{k+1}\right] \notin L(\phi')$.

Inductive Step ($m > 1$): Let ϕ be of the form $e_1 + \cdots + e_m$. We define $M = M_1 + \cdots + M_m$ where M_i is derived from e_i in the same manner to derivation of M from expressions in the special case above. We show that M satisfies the claim. We use induction on m. If $e_1 \not\subseteq \phi'$, then this case reduces to the case above. Otherwise, we know that $m > 1$ and $e_1 \subseteq \phi''$ such that $\phi' = \phi'' + \phi'''$ where ϕ'' is a minimum mlg (i.e, a mlg consisting of the least number of expressions) which satisfies $e_1 \subseteq \phi''$. Assume that ϕ'' is of the form $e'_i + \phi'_i$ where $i : 1 \leq i \leq n$ where n is the number of expressions in ϕ''. For each i, we have two possible cases.

- ◦ If e'_i is atomic. Since ϕ'' is a minimum mlg which satisfies $e_1 \subseteq \phi''$, $M_1 \notin L(\phi'_i)$. Furthermore, we know that $e_2 + \cdots + e_m \not\subseteq \phi'''$ (otherwise, $\phi \subseteq \phi'$, contradiction). By induction hypothesis, it follows that $M_2 + \cdots + M_m \notin L(\phi''')$. Since e'_i is atomic, we infer that $M_1 + \cdots + M_m \notin L(\phi')$.
- ◦ If e'_i is a star expression. Since ϕ'' is a minimum mlg which satisfies $e_1 \subseteq \phi''$, $M_1 \notin L(\phi'_i)$. Furthermore, since e'_i is a star expression, we know that $e_2 + \cdots + e_m \not\subseteq e'_i + \phi'''$ (otherwise, $\phi \subseteq \phi'$, contradiction). By induction hypothesis, it follows that $M_2 + \cdots + M_m \notin L(e'_i + \phi''')$. We infer that $M_1 + \cdots + M_m \notin L(\phi')$.

Appendix A.3 Proof of Theorem 3

First, we show some auxiliary lemmas.

Lemma A.3 For an infinite alphabet $A^{\circledast}$ and a non-empty word $\mu \in (A^{\circledast})^*$, there is a wlg ψ such that a word $\nu \in L(\psi)$ iff $\mu \not\leq^w \nu$.

Proof. Let μ be of the form $M_1 \bullet M_2 \bullet \ldots \bullet M_m$ where $M_1, \ldots, M_m$ are multisets over a finite alphabet A. Let e_i be a star expression Φ_i^* where Φ_i is obtained from multiset M_i for each $i : 1 \leq i \leq m$ as shown in Lemma A.1, satisfying that a multiset $M \in L(\Phi_i)$ if $M_i \not\leq^m M$ for $i : 1 \leq i \leq m$.

On the other hand, for each multiset M_i, it is easy to construct a smallest mlg ϕ_i such that $M_i \in L(\phi_i)$ for each $i : 1 \leq i \leq m$.

We define wlg ψ by $e_1 \bullet \phi_1 \bullet \cdots \bullet e_{m-1} \bullet \phi_{m-1} \bullet e_m$.

First we show that $\nu \in L(\psi) \rightarrow \mu \not\leq^w \nu$ by contraposition. Assume $\mu \leq^w \nu$. We prove that $\nu \notin L(\psi)$. From the definition of $\leq^w$, we know that ν is of the form $\nu_1 \bullet M_1 \bullet \nu_2 \bullet \cdots M_m \bullet \nu_{m+1}$ where $\nu_i \in (A^{\circledast})^*$. From definition of e_i, we know that $M_i \notin L(e_i)$ and hence, $\nu_i \bullet M_i \notin L(e_i)$ for each $i : 1 \leq i \leq m$. This implies that $\nu_1 \bullet M_1 \bullet \nu_2 \bullet \cdots \bullet \nu_m \bullet M_m \notin L(e_1 \bullet \phi_1 \bullet \cdots \bullet \phi_{m-1} \bullet e_m) = L(\psi)$, i.e $\nu \notin L(\psi)$.

Next, we prove that $\mu \not\leq^w \nu \rightarrow \nu \in L(\psi)$. Let l be the largest natural number such that $M_1 \bullet \ldots \bullet M_l \leq^w \nu$. Obviously, $0 \leq l < m$. This means that ν is of the form $\nu_0 \bullet M_1 \bullet \nu_1 \bullet M_2 \cdots \bullet \nu_{l-1} \bullet M_l \bullet \nu_l$, where ν_i is a word over $A^{\circledast} \setminus (M_{i+1} \uparrow)$ for $i : 0 \leq i < l$, where $M_{i+1} \uparrow$ denotes the upward closure of multiset M_{i+1}. Furthermore, we know that M_{l+1} does not occur in ν_l (otherwise, we will have $M_1 \bullet \cdots \bullet M_{l+1} \leq^w \nu$ violating the maximality of l). This implies that $\nu_i \in L(e_{i+1})$ for each $i : 0 \leq i \leq l$. From this and the fact that $M_i \in L(\phi_i)$, we have $\nu \in L(\psi)$. □

Lemma A.4 For wlgs ψ_1, ψ_2, there is a set of wlgs $\psi_1 \cap \psi_2$ such that $L(\psi_1 \cap \psi_2) = L(\psi_1) \cap L(\psi_2)$.

Proof. In case ψ_1 and ψ_2 are atomic expressions (mlgs), $\psi_1 \cap \psi_2$ is same as intersection of two mlgs. In case, one of them is a star expression, i.e, $\psi_1 = \{\phi_1, \ldots, \phi_k\}^*$ and $\psi_2 = \phi_2$, then $\psi_1 \cap \psi_2 = \{\phi_1 \cap \phi_2, \ldots, \phi_k \cap \phi_2\}$. If both of them are star expressions, i.e, $\psi_1 = \{\phi_1, \ldots, \phi_k\}^*$ and $\psi_2 = \left\{\phi'_1, \ldots, \phi'_m\right\}^*$, then $\psi_1 \cap \psi_2 = \left\{\phi_1 \cap \phi'_1, \ldots, \phi_k \cap \phi'_m\right\}^*$. Let $\psi_1 = e_1 \bullet \psi'_1$ and $\psi_2 = e_2 \bullet \psi'_2$ be non-empty wlgs. We have four cases depending on the form of e_1 and e_2.

(1) e_1 and e_2 are atomic expressions,

$$\Psi = \left\{(e_1 \cap e_2) \bullet (\psi'_1 \cap \psi'_2)\ ,\ (\psi_1 \cap \psi'_2)\ ,\ (\psi'_1 \cap \psi_2)\right\}$$

(2) e_1 is an atomic expression and e_2 is a star expression.
$\Psi = \left\{(e_1 \cap e_2) \bullet (\psi'_1 \cap \psi_2)\ ,\ (\psi_1 \cap \psi'_2)\right\}$

(3) e_1 is a star expression and e_2 is an atomic expression.
$\Psi = \left\{(e_1 \cap e_2) \bullet (\psi_1 \cap \psi'_2)\ ,\ (\psi'_1 \cap \psi_2)\right\}$

(4) e_1 and e_2 are star expressions.
$\Psi = \left\{(e_1 \cap e_2) \bullet (\psi_1 \cap \psi'_2)\ ,\ (e_1 \cap e_2) \bullet (\psi'_1 \cap \psi_2)\right\}$

Notice that we assume the operator $\bullet$ can be distributed over sets of wlgs. □

Main Proof of Theorem 3:

Consider a downward closed language L of words over multisets. If $L = \emptyset$, then $L = L(\Psi)$ where $\Psi = \emptyset$. Otherwise, complement of L is upward closed and can be characterized by a finite set of words over multisets given by $\{w_1, \ldots, w_n\}$ (by Higman's theoremHigman [1952]). Thus, a word $v \in L$, if and only if $w_i \not\leq^w v$ for any $i : 1 \leq i \leq n$. $w_i \in (A^{\circledast})^*$ for each $i : 1 \leq i \leq n$ and by Lemma A.3 and Lemma A.4, it follows that there are wlgs, $\psi_1, \ldots, \psi_n$ such that $L = L(\psi_1) \ \cap \cdots \cap \ L(\psi_n)$.

Appendix A.4 Proof of Lemma 4

Assume $\psi \not\subseteq \psi'$ for any $\psi' \in \left\{\psi'_1, \ldots, \psi'_n\right\}$. We show that there is a word $w \in L(\psi)$ such that $w \notin L(\psi')$ for any $\psi' \in \left\{\psi'_1, \ldots, \psi'_n\right\}$ which implies that $w \not\subseteq L(\left\{\psi'_1, \cdots, \psi'_n\right\})$. This proves that $\psi \not\subseteq \left\{\psi'_1, \cdots, \psi'_n\right\}$.

Let k be the number of expressions in wlg $\psi' \in \left\{\psi'_1, \cdots, \psi'_n\right\}$, i.e, $\psi' = e'_1 \ \bullet \ \ldots \ \bullet \ e'_k$.

Induction hypothesis (**IH**): For a wlg $\psi = e_1 \ \bullet \ \cdots \ \bullet \ e_m$ with $m \geq 1$, we have $e_1 \ \bullet \ \cdots \ \bullet \ e_m \not\subseteq \psi' \to w_1 \ \bullet \ \cdots \ \bullet \ w_m \notin L(\psi')$ where words $w_i \in L(e_i)$ for $i : 1 \leq i \leq m$ and $w = w_1 \ \bullet \ \cdots \ \bullet \ w_m$.

Base case ($m = 1$):

First, we prove the claim where ψ is an atomic expression, i.e a mlg ϕ. In that case, we follow the proof steps in the general case of Lemma 3 and define w to be a word containing a single multiset M derived from ϕ for some natural number k_m where k_m is the length of longest mlg among all mlgs in $e'_1, \cdots, e'_k$. Given, $\psi \not\subseteq \psi'$ and ψ is atomic, $M \notin L(e'_i)$ for any $i : 1 \leq i \leq k$. Hence, $w \notin L(\psi')$.

Second, we prove the claim where ψ is a star expression $e = \{\phi_1, \cdots, \phi_l\}^*$ with ϕ_i is a mlg for $i : 1 \leq i \leq l$. In this case, we define w such that $w = (M_1 \ \bullet \ \ldots \ \bullet \ M_l)^{k+1}$ and M_i is derived from ϕ_i as before. We use induction on k (length of ψ') to show that $w \notin L(\psi')$. The base case ($k = 0$) is trivial. For the induction step, we assume $k > 0$. There are two cases.

- ○ e'_k is atomic. By the induction hypothesis, we have that $(M_1 \ \bullet \ \ldots \ \bullet \ M_l)^k \notin L(e'_1 \ \bullet \ \cdots \ \bullet \ e'_{k-1})$. Since e'_k is atomic, $(M_1 \ \bullet \ \ldots \ \bullet \ M_l)^{k+1} \notin L(\psi')$.
- ○ e'_k is star expression. We know that $e \not\subseteq e'_k$ (otherwise, $\psi \subseteq \psi'$ and that is contradiction). Since e'_k is a star expression and $e \not\subseteq e'_k$, there must be a mlg ϕ_i in e such that $i : 1 \leq i \leq l$ and $M_i \notin L(e'_k)$. This implies that $M_1 \ \bullet \ \ldots \ \bullet \ M_l \notin L(e'_k)$. By the induction hypothesis, we have that $(M_1 \ \bullet \ \ldots \ \bullet \ M_l)^k \notin L(e'_1 \ \bullet \ \cdots \ \bullet \ e'_{k-1})$. This implies that $(M_1 \ \bullet \ \ldots \ \bullet \ M_l)^{k+1} \notin L(\psi')$.

Inductive Step ($m > 1$): Let ψ be of the form $e_1 \bullet \cdots \bullet e_m$. We define $w = w_1 \bullet \cdots \bullet w_m$ where w_i is derived from e_i in the same manner to derivation of w from expressions in the special case above. We show that w satisfies the claim. We use induction on m. If $e_1 \not\subseteq \psi'$, then this case reduces to the case above. Otherwise, we know that $m > 1$. Let k_1 be the minimum natural number such that $e_1 \subseteq e'_1 \bullet \cdots \bullet e'_{k_1}$. Now, we have two possible cases.

- ○ If e'_{k_1} is atomic. Since k_1 is the minimum natural number satisfying $e_1 \subseteq e'_1 \bullet \cdots \bullet e'_{k_1}$, $w_1 \notin L(e'_1 \bullet \cdots \bullet e'_{k_1-1})$. Furthermore, we know that $e_2 \bullet \cdots \bullet e_m \not\subseteq e'_{k_1+1} \bullet \cdots \bullet e'_k$. (otherwise, $\psi \subseteq \psi'$, contradiction). By induction hypothesis, it follows that $w_2 \bullet \cdots \bullet w_m \notin L(e'_{k_1+1} \bullet \cdots \bullet e'_k)$. Since e'_{k_1} is atomic, we infer that $w_1 \bullet \cdots \bullet w_m \notin L(\psi')$.
- ○ If e'_{k_1} is a star expression. Since k_1 is the minimum natural number satisfying $e_1 \subseteq e'_1 \bullet \cdots \bullet e'_{k_1}$, $w_1 \notin L(e'_1 \bullet \cdots \bullet e'_{k_1-1})$. Furthermore, since e'_{k_1} is a star expression, we know that $e_2 \bullet \cdots \bullet e_m \not\subseteq e'_{k_1} \bullet \cdots \bullet e'_k$ (otherwise, $\psi \subseteq \psi'$, contradiction). By induction hypothesis, it follows that $w_2 \bullet \cdots \bullet w_m \notin L(e'_{k_1} \bullet \cdots \bullet e'_k)$. We infer that $w_1 \bullet \cdots \bullet w_m \notin L(\psi')$.

Proof of Theorem 7

Given that the acceleration criterion holds at a region generator θ with respect to a transition t, we show that for each sequence $\theta_1, \theta_2, \ldots$ of region generators in $Accel_t(\theta)$ such that $\theta_i \in (Post_{Time} \circ Step_t)(\theta_{i-1})$ for $i > 0$, there is an integer n such that $\theta_n \subseteq \bigcup_{0 \le i \le n-1} (Post_{Time} \circ Step_t)^i(\theta)$, i.e we prove that

the set of region generators computed by $Accel_t(\theta)$ is finite.

First we introduce some notations. We overload || operator for mlgs, wlgs and region generators, respectively to quantify the symbols in a region generator.

For a mlg $\phi = \{a_1, \ldots, a_k\}^{\circledast} + a_{k+1} + \cdots + a_{k+\ell}$, $|\phi| = k + \ell$. Notice that $|\epsilon| = 0$.

For a word star expression $e = \{\phi_1, \ldots, \phi_k\}^*$, $|e| = |\phi_1| + \cdots + |\phi_k|$ and for a wlg $\psi = e_1 \bullet \cdots \bullet e_\ell$, $|\psi| = |e_1| + \cdots + |e_\ell|$.

Now, for a region generator $\theta = (\phi_0, \psi, \phi_{max})$, we have $|\theta| = |\phi_0| + |\psi| + |\phi_{max}|$.

Lemma A.5 There is a bound K such that for all region generator $\theta' \in Accel_t(\theta)$, $|\theta'| \le K$.

Notice that Theorem 7 directly follows from Lemma A.5.

Now, we show that there is indeed such a bound K as claimed in Lemma A.5.

First, we give a measure of the maximum number of word expressions that can be introduced in a region generator during the computation of $Accel_t(\theta)$. The symbols in the region generator can belong to Type 2 places and each of them can be removed and inserted again as a new atomic word expression anywhere in the region generator. This corresponds to the case when the new token has totally different fractional part than all other tokens: e.g., given $\psi = \psi_1 \bullet \psi_2$, we have $\psi_1 \bullet p(n) \bullet \psi_2 \in (\psi \oplus p(n))$. Furthermore, the new tokens in Type 2 place can have a fractional part common with some other token belonging to some star expression. This case also increases the number of word expressions. Recall that $\{\phi_1, \cdots, \phi_k\}^* \oplus p(n) = \{\phi_1, \cdots, \phi_k\}^* \bullet (\phi_i \oplus p(n)) \bullet \{\phi_1, \cdots, \phi_k\}^*$ for $i : 1 \le i \le k$.

Therefore total number of word expressions in any θ' is governed by the size $|\theta|$. At most, all symbols can reappear as atomic expressions and we add star expressions in accelerated addition before and after each such atomic expression. Since accelerated addition to a word star expression does not increase the number of word expressions (by normalisation), the maximum possible number of word expressions, k_1 is $2 * |\theta| + 1$.

Secondly, we give a measure of the maximum number of symbols in each word expression of any region generator θ' in $Accel_t(\theta)$. Each word expression of θ can contain tokens from Type 2 places. In some firing of t, such tokens can be placed together in a single atomic word expression. Moreover, each such atomic expression will also have tokens added during accelerated addition and these new tokens will be placed with some of the old tokens added. Thus maximum number of symbols in each of the word expressions of θ' is given by $|\theta| + max * s$, where s is given by the size of the set $\mathcal{A}_{out}(t) \setminus T_2$ (tokens put during accelerated addition) and $max * s$ is the maximum number of symbols for old and newly added tokens during accelerated addition in a word expression. This is due to the fact that the accelerated addition to a multiset star expression is bounded by the number of places and the value of max. Therefore, we can say that the maximum number of symbols in each word expression of any θ' is maximally bounded by $k_2 = |\theta| + |P| * (max + 1)$ where P is the set of places in TPN.

Given that the maximum number of word expressions in any θ' is k_1 and the maximum number of symbols in each word expression is k_2, we have the bound

$$K = (k_1 + 2) * k_2$$

where 2 corresponds to the first and the third part of θ'. This implies that $K = O(|\theta|^2)$.