Forward Reachability Analysis of Timed Petri Nets

Parosh Abdulla, Johann Deneux, Pritha Mahata and Aletta Nylén

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2003-056
December 2003
ISSN 1404-3203

Forward Reachability Analysis of Timed Petri Nets

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

Dept. of Information Technology
Uppsala University
Sweden
{parosh,johannd,pritha,aletta}@it.uu.se

Abstract. We consider verification of safety properties for concurrent real-timed systems modelled by 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. Downward closed languages 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. We also used the tool to generate a finite-state abstraction of the protocol.

1 Introduction

Background: Petri nets are a popular model in the verification of concurrent programs. Several classes of *Timed Petri Nets (TPNs)* have been introduced in the literature for verification of real-time systems; e.g. [RP85,MF76,BD91,GMMP91] (see [Bow96] for a survey). A TPN is an extension of a Petri net in the sense that each token is assumed to have an "age" which is represented by a real valued clock. Besides the firing of transitions (which is now controlled by the ages of the tokens), a TPN may perform timed transitions describing the passage of time. TPNs are computationally more powerful than timed automata [AD90] since they operate on a potentially unbounded number of clocks. This implies that TPNs can for instance model parameterized timed systems (systems consisting of an unbounded number of timed processes) [AN01,AJ98].

A fundamental problem for TPNs (and also for standard Petri nets) is that of *coverability*: check whether an upward closed set of *final markings* is reachable from a set of initial markings. Using standard techniques [VW86,GW93], we can reduce several classes of safety properties for TPNs into the coverability problem. More precisely, the final markings will correspond to "bad markings" which 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 several classes of models, including TPNs, the set of backward reachable states is computable [AN01]. However, this is not the case for forward analysis. For instance, it is not obvious how to extend the Karp-Miller algorithm from standard (untimed) Petri nets to TPNs. In fact, the set of forward reachable states in TPNs is in general not computable, and therefore any procedure for performing forward reachability analysis on TPNs is necessarily incomplete. On the other hand, forward analysis is practically very attractive. Often, it turns out [ABLS01,AAB00,ABJ98] that it is possible to design semi-algorithms for forward analysis which terminate on almost all interesting examples. Furthermore, 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 may arise during the execution of the system, while backward closure only describes the set from which the system can fail. Also, forward analysis can often be used for constructing *abstract automaton* (*symbolic graph*) which constitute finite abstractions of the systems, and on which standard techniques for finite state model checking can be applied [ABLS01,AAB$^+$99,BLO98,LBBO01]. More interestingly, forward analysis can often be more efficient on practical examples than backward analysis [FRSB02]. Finally, forward analysis can be used in conjunction with backward analysis. For instance, we can use forward analysis to derive invariants which can then be used to prune the backward reachable state space making backward analysis more efficient [FRSB02].

Contribution: In this paper, we consider performing forward reachability analysis for TPNs. We provide an abstraction of the set of reachable markings by taking its *downward closure*. There are at least two reasons why downward closed sets are advantageous. First, this abstraction is exact with respect to coverability and consequently with respect to safety properties. In other words, a given TPN satisfies any safety property exactly when the downward closure of the set of reachable states satisfies the same property. Second, the downward closure has usually a much simpler structure than the exact set of reachable states, since it does not contain as much information about system behaviour, and is therefore much easier to analyze. This is the case also for (non-timed) Petri nets. For instance, downward closed sets cannot express constrains of the form "a certain place contains at least as many tokens as another place". Such constraints are irrelevant from the point of view of safety properties.

Since the set of reachable markings (and its downward closure) is in general infinite, we need to devise a symbolic representation for downward closed sets. Our representations is a variant of that of regions for timed automata [AD90]. A region roughly encodes information about the integral parts of clock values and ordering among fractional parts of clock values. Regions are designed for systems with finitely many clocks, and hence are not capable of capturing the behaviour of TPNs. Therefore, we introduce a more powerful symbolic representation which we call a *region generator*. Each region generator denotes the union of an infinite number of regions. The region generators are defined 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. The set of markings characterized by a region generator is downward closed. In fact, we show that any downward closed set of markings can be characterized by region generators.

We also show that region generators allow the basic operations in forward analysis, such as checking membership, entailment, and computing the post-images of a single transition. While these three operations are sufficient for checking the validity of invariants, they are are in general too weak to make the forward analysis procedure terminate. Therefore, we give an acceleration scheme to speed up the forward reachability analysis. The acceleration 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 a forward reachability procedure and verified the mutual exclusion property for a parameterized version of Fischer's protocol. Our acceleration scheme allowed us to generate the (downward closure of the) full reachability set for Fischer's protocol. From the analysis, we also generate an abstract automaton with 12 states and 54 transitions, which is weakly bisimilar to the TPN model of the protocol. Minimizing this abstract automaton using *The Concurrency Workbench* [CPS93], yields an automaton with only 3 states and 8 transitions.

Related Work: In [ABJ98,ABB01], we present *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 an alphabet).

Both [DR00] and [FRSB02] consider (untimed) Petri nets and give symbolic representations for upward closed sets and downward closed sets of markings, respectively. The works in [FIS00,BG96,BGWW97,BH97] give symbolic representation for FIFO automata. All these representations are designed for weaker models (Petri nets and FIFO automata) and cannot capture the behaviour of timed Petri nets.

In [AN01,AN00,AJ98] we consider timed Petri nets. The symbolic representations in these papers characterize upward closed sets of markings, and can be used for backward analysis, but not forward analysis.

Outline: In the next section, we introduce timed Petri nets. In Section 3 we define an ordering on markings and in Section 4, we define the coverability problem for TPNS. The next three sections are devoted to the hierarchical definition of region generators. In Section 5 we define *multiset language generators (mlgs)*. Each mlg characterizes a language whose elements are multisets (rather than words) on a finite alphabet. In Section 6 we define *word language generators*. The underlying alphabet is infinite; namely the set of multisets over a finite alphabet. In Section 7 we introduce *region generators*. Section 8 gives the forward reachability algorithm. Section 9 and Section 10 give algorithms for computing post-images and acceleration respectively. In Section 12 we report on some experiments with our implementation. Finally, we give conclusions and directions for future research in Section 13.

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 a subinterval of natural numbers. When firing a transition, tokens which are removed (added) from (to) places should have their ages lying in the intervals of corresponding arcs.

We use $\mathbb{N}, \mathbb{R}^{\geq 0}$ to denote the sets of natural numbers, nonnegative reals respectively.

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

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

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

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 $[p_1(x_1 + x)\ ,\ \ldots\ ,\ p_n(x_n + x)]$.

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

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

We define the set of *discrete transitions* $\longrightarrow_{Disc}$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of firing the transition t. More precisely, $M_1 \longrightarrow_t M_2$ if the set of input arcs $\{p(I) | \, In(t,p) = I\}$ is of the form $\{p_1(I_1), \ldots, p_k(I_k)\}$, the set of output arcs $\{p(I) | \, Out(t,p) = I\}$ is of the form $\{q_1(\mathcal{J}_1), \ldots, q_\ell(\mathcal{J}_\ell)\}$, 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:
- $B_1 \leq^m M_1$
- $x_i \in I_i$, for $i : 1 \leq i \leq k$.
- $y_i \in \mathcal{J}_i$, for $i : 1 \leq i \leq l$.
- $M_2 = (M_1 \setminus B_1) \odot B_2$.

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. Also, the newly produced tokens may only have ages which belong to the relevant intervals.

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

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

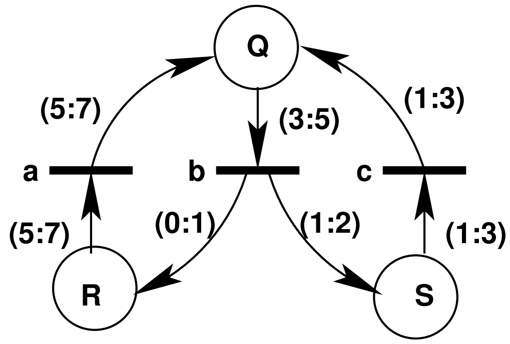

Fig. 1. A small timed Petri net.

Example 1. Figure 1 shows an example of a TPN where $P = \{Q, R, S\}$ and $T = \{a, b, c\}$. For instance, $In(b,Q) = (3:5)$ and $Out(b,R) = (0:1)$ and $Out(b,S) = (1:2)$. 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 $M_1 \longrightarrow_b M_2$ where $M_2 = [R(0.2),\ S(1.6),\ R(5.8),\ R(5.0)]$.

3 Ordering

In this section, we define an ordering on markings which extends the *region* equivalence relation of [AD90].

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

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

Each region characterizes an infinite set of markings as follows. Assume a marking $M = [p_1(x_1), \ldots, p_n(x_n)]$ and a region $r = (B_0, B_1 B_2 \cdots B_m, B_{m+1})$. Let B_j is of the form $\left[q_{j1}(y_{j1}), \ldots, q_{j\ell_j}(y_{j\ell_j})\right]$ for $j : 0 \leq j \leq m$ and B_{m+1} is of the form $\left[q_{m+1\,1}, \ldots, q_{m+1\,l_{m+1}}\right]$. We say that M *satisfies* r, written $M \models r$, if there is a bijection h from the set $\{1, \ldots, n\}$ to the set of pairs $\left\{(j,k) \mid (0 \leq j \leq m+1) \wedge (1 \leq k \leq \ell_j)\right\}$ 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 r.
- If $h(i) = (j,k)$ then $j = m+1$ iff $x_i > max$. Tokens older than *max* should correspond to elements in multiset B_{m+1}. The actual ages of these tokens are not relevant.
- If $x_i \leq max$ and $h(i) = (j,k)$ then $\lfloor x_i \rfloor = y_{j\,k}$. The integral part of the age of tokens should agree with the natural number specified by the corresponding elements in w.
- If $x_i \leq max$ and $h(i) = (j,k)$ then $fract(x_i) = 0$ iff $j = 0$. Tokens with zero fractional parts correspond to elements in multiset B_0.
- If $x_{i_1}, x_{i_2} < max$, $h(i_1) = (j_1, k_1)$ and $h(i_2) = (j_2, k_2)$ then $fract(x_{i_1}) \leq fract(x_{i_2})$ iff $j_1 \leq j_2$. Tokens with equal fractional parts correspond to elements in the same multiset (unless they belong to B_{m+1}). The ordering among multisets inside r reflects the ordering among fractional parts in clock values.

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

The region construction defines an equivalence relation $\equiv$ on the set of markings such that $M_1 \equiv M_2$ if, for each region r, it is the case that $M_1 \in [\![r]\!]$ iff $M_2 \in [\![r]\!]$.

It is well-known [AD90] that $\equiv$ is a congruence on the set of markings. In other words, if $M_1 \longrightarrow M_2$ and $M_1 \equiv M_3$ then there is an M_4 such that $M_2 \equiv M_4$ and $M_3 \longrightarrow M_4$.

Ordering Next 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 $\{M|\ \exists M' \in \mathsf{M} :\ M' \leq^m M\}$. 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 $M_3 = M_3' \odot M_3''$. Since $\equiv$ is a congruence, there is M_4' such that $M_2 \equiv M_4'$ and $M_3' \longrightarrow M_4'$. We define $M_4 = M_4' \odot M_4''$ where $M_4'' = M_3''$ if $M_3' \longrightarrow_{Disc} M_4'$, and $M_4'' = (M_3'')^{+x}$ if $M_3' \longrightarrow_x M_4'$.

Example 2. For the TPN in Figure 1, we have $max = 7$. Following is an example of a region :

$$r = ([R(2)],\ [S(1)]\ \ [R(1),\ S(5)],\ [Q])$$

Marking $M_1 = [R(2.0)\ ,S(1.5),\ R(1.7),\ S(5.7), Q(8.9)]$ and $M_2 = [R(2.0)\ ,S(1.7),\ R(1.8),\ S(5.8), Q(9.9)]$ satisfy the above region. Notice that $M_1 \equiv M_2$. Let $M_3 = M_2 \odot [R(1.2),\ Q(14.2)]$. Since $M_2 \leq^m M_3$ and $M_1 \equiv M_2$, we have $M_1 \preceq M_3$.

4 Coverability

The *coverability problem* is defined as follows:

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

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

In [AČJYK00]) it is shown that if the safety property is specified as a regular set of traces then the set of bad states will be the upward closure of a finite set (here M_{fin}) of markings.

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

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

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

Infeasibility of the Karp Miller Algorithm The Karp Miller algorithm [KM69] is the classical method used for checking coverability in standard (untimed) Petri nets. However, it is not obvious how to extend the algorithm to TPNs. The Karp Miller algorithm constructs a reachability tree staring from an initial marking. The main step is to detect paths in the reachability tree which leads from a marking M_1 to a larger marking M_2. In such a case, we make 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 $M_1 < M_2$) the only useful 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 can therefore not be guaranteed.

5 Multiset Language Generators (mlgs)

In this section, we perform the first step in giving the hierarchical definition of region generators. We define *multiset language generators(mlgs)*. Each mlg characterizes a language which contains multisets (rather than words) over a finite alphabet.

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

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 $\mathcal{L}(\phi)$, 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 $a \in A$. $\mathcal{L}(a) = \{[a]\ ,\ \varepsilon\}$.

  - a *star expression* of the form $S^{\circledast}$ where $S \subseteq A$. $\mathcal{L}(S^{\circledast}) = \{[a_1\ ,\ \dots\ ,\ a_m] \mid m \geq 0 \wedge a_1, \dots, a_m \in S\}$.
- An *mlg* ϕ is a (possibly empty) sequence $e_1 \odot \cdots \odot e_\ell$ of expressions. $\mathcal{L}(\phi) = \{B_1 \odot \cdots \odot B_\ell \mid B_1 \in \mathcal{L}(e_1), \cdots, B_\ell \in \mathcal{L}(e_\ell)\}$. We denote an empty mlg by ε and assume that $\mathcal{L}(\varepsilon) = \{\varepsilon\}$.

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

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

Sometimes we identify mlgs with the languages they represent, so we write $\phi_1 \subseteq \phi_2$ (rather than $\mathcal{L}(\phi_1) \subseteq \mathcal{L}(\phi_2)$), and $B \in \phi$ (rather than $B \in \mathcal{L}(\phi)$), etc.

Normal Form An mlg ϕ is said to be normal if it is of the form $e \odot e_1 \odot \cdots \odot e_k$ where

- e is a star expression and $e_1, \dots, e_k$ are atomic expressions and
- for each $i : 1 \leq i \leq k$, $e_i \not\subseteq e$.

A set of mlgs $\{\phi_1, \cdots, \phi_m\}$ is said to be *normal* if ϕ_i is *normal* for each $i : 1 \leq i \leq m$, and for $i, j : 1 \leq i \neq j \leq m$, $\phi_i \not\subseteq \phi_j$.

For each mlg ϕ, there is a unique (up to commutativity of the operators) normal mlg ϕ' such that $\mathcal{L}(\phi') = \mathcal{L}(\phi)$. 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 $\mathcal{L}(a \odot S^{\circledast}) = \mathcal{L}(S^{\circledast})$.
- Merge all star expressions using the property that $\mathcal{L}\left({S_1}^{\circledast} \odot {S_2}^{\circledast}\right) = \mathcal{L}\left((S_1 \cup S_2)^{\circledast}\right)$.

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 from 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 reduced to a normal form.

Entailment In the following, we give an algorithm for computing entailment 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. $\varepsilon \subseteq \phi_2$ and $\phi_1 \not\subseteq \varepsilon$ if $\phi_1 \neq \varepsilon$. Given two non-empty mlgs $\phi_1 = e_1 \odot \phi_1'$ and $\phi_2 = e_2 \odot \phi_2'$, 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 $i \in \{1, \ldots, m\}$.*

From Lemma 3 and the algorithm for entailment of mlgs, we get

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

Example 3. Consider a finite alphabet $A = \{a, b, c\}$ and the set of multisets $A^{\circledast}$ over A. Given mlg $\phi_1 = \{a, b\}^{\circledast} \odot c$, examples of multisets in $\mathcal{L}(\phi_1)$ are $[a^2, b], [b, c], [a^3, c]$. Next consider $\phi_2 = b \odot c$. $\mathcal{L}(\phi_2) = \{\varepsilon, [c], [b], [b, c]\}$. Notice that ϕ_1, ϕ_2 are in normal form and $\phi_1 \subseteq \phi_2$.

6 Word Language Generators (wlgs)

In this section, we describe the second step in the definition of region generators. We consider languages where each word is a sequence of multisets over a finite alphabet A, i.e., each word is a member of $(A^{\circledast})^*$. The language is then a subset of $(A^{\circledast})^*$. Notice that the underlying alphabet, namely $A^{\circledast}$ is infinite.

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

We define the ordering $\leq^w$ on set of words such that $w_1 \leq^w w_2$ if there is a strictly monotonic injection $h : \{1, \ldots, |w_1|\} \to \{1, \ldots, |w_2|\}$ where $w_1(i) \leq^m w_2(h(i))$ for $i : 1 \leq i \leq |w_1|$. Recall that $\leq^m$ is the ordering defined on mutlisets (Section 2).

We shall consider languages which are downward closed with respect to $\leq^w$. In a similar manner to Section 5, we define downward closed *word language generators (wlgs)* and (word) languages as follows.

- An *expression* over A is of one of the following two forms:
  - an *atomic expression* is an mlg ϕ over A. $\mathcal{L}(\phi) = \{B | \, B \in \phi\} \cup \{\varepsilon\}$.
  - a *star expression* of the form $\{\phi_1, \cdots, \phi_k\}^*$, where $\phi_1, \ldots, \phi_k$ are mlgs over A. $\mathcal{L}(\{\phi_1, \cdots, \phi_k\}^*) = \{B_1 \bullet \cdots \bullet B_m | \, (m \geq 0) \text{ and } B_1, \ldots, B_m \in \mathcal{L}(\phi_1) \cup \cdots \mathcal{L}(\phi_k)\}$.
- A *word language generator (wlg)* ψ over A is a (possibly empty) concatenation $e_1 \bullet \cdots \bullet e_\ell$ of word expressions $e_1, \ldots, e_\ell$. $\mathcal{L}(\psi) = \{w_1 \bullet \cdots \bullet w_\ell | \, w_1 \in \mathcal{L}(e_1) \wedge \cdots \wedge w_\ell \in \mathcal{L}(e_\ell)\}$.

Notice that the concatenation operator is associative, but not commutative (as is the operator $\odot$ for multisets). Again, we denote the empty wlg by ε. For a set $\Psi = \{\psi_1, \cdots, \psi_m\}$ of wlgs, we define $\mathcal{L}(\Psi) = \mathcal{L}(\psi_1) \cup \cdots \cup \mathcal{L}(\psi_m)$.

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

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.
- $e_i \bullet e_{i+1} \not\subseteq e_i$ 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.
- $\psi_i \not\subseteq \psi_j$ for each $i, j : 1 \leq i \neq j < m$.

For each wlg ψ, there is a unique *normal* wlg ψ' such that $\mathcal{L}(\psi) = \mathcal{L}(\psi')$. 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 of wlgs Now, we extend the algorithm for checking entailment of mlgs to check entailment of wlgs of the form $\psi \subseteq \psi'$.

First, we extend $\subseteq$ as the least reflexive, partial order on word expressions in a similar manner.

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

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 $i \in \{1, \ldots, m\}$.*

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 4. Consider the same alphabet A and mlgs ϕ_1, ϕ_2 as in Example 3. Given a wlg $\psi_1 = \{\phi_2\}^* \bullet \phi_1$, example of word in $\mathcal{L}(\psi_1)$ is $[b,c] \bullet [b] \bullet \left[a^3\right]$. For a wlg $\psi_2 = \{\phi_1\}^* \bullet \phi_2$, example of a word in $\mathcal{L}(\psi_2)$ is $\left[a^2, b^3, c\right] \bullet \left[a^3, c\right] \bullet [b,c]$. Notice that $\psi_1 \subseteq \psi_2$, but and $\psi_2 \not\subseteq \psi_1$.

7 Region Generators

We introduce region generators and describe how each region generator characterizes a potentially infinite set (language) of regions.

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 $\phi_{\max}$ is an mlg over P. The language $\mathcal{L}(\theta)$ contains exactly each region of the form (B_0, w, B_{max}) where $B_0 \in \phi_0$, $w \in \psi$, and $B_{max} \in \phi_{\max}$. 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 $\theta_1 \subseteq \theta_2$ iff $\phi_0^1 \subseteq \phi_0^2$, $\psi^1 \subseteq \psi^2$, and $\phi_{\max}^1 \subseteq \phi_{\max}^2$. In other words entailment between region generators can be computed by checking entailment between the individual elements.

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

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

A finite set $\Theta = \{\theta_1, \ldots, \theta_m\}$ of region generators denotes the union of its elements, i.e, $[\![\Theta]\!] = \bigcup_{1 \leq i \leq m} [\![\theta_i]\!]$.

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

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

By Theorem 1 and Theorem 3 it follows that for each set R of regions which is downward closed with respect to $\leq^r$, there is a set of region generators Θ such that $\mathcal{L}(\Theta) = R$. Also, by definition, it follows that if $r_1 \leq^r r_2$ then $M_1 \preceq M_2$ for each $M_1 \in [\![r_1]\!]$ and $M_2 \in [\![r_2]\!]$.

From this we get the following

Theorem 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* $\mathsf{M} = [\![\Theta]\!]$.

8 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 $[\![\Theta_{init}]\!] \cap \mathsf{M}_{fin} \uparrow = \emptyset$. The algorithm computes the sequence of sets of regions generators $\Theta_{i+1} = \Theta_i \cup succ(\Theta_i)$ with $\Theta_0 = \Theta_{init}$. If $[\![\Theta_i]\!] \cap \mathsf{M}_{fin} \uparrow \neq \emptyset$ (amounts to checking membership of elements of M_{fin} in $[\![\Theta]\!]$), 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 9, compute the effect of timed and discrete transitions respectively. $Step_t$, defined in section 10, implements acceleration.

Even if we know by Theorem 5 that there is finite set Θ of region generators such that $Reach([\![\Theta_{init}]\!]) = [\![\Theta]\!]$, it is quite straightforward to apply the techniques presented in [May00] to show the following.

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

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

9 Computing *Post*

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

9.1 $Post_{Time}$

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

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

1. If $B_0 = \varepsilon$, i.e., there are no tokens in M with ages whose fractional parts are equal to zero. Let w be of the form $w_1 \bullet B_1$. The behaviour of the TPN from M due to

passage of time is decided by a certain subinterval of $\mathbb{R}^{\geq 0}$ which we denote by $stable(M)$. This interval is defined by $[0:1-x)$ where x is the highest fractional part among the tokens whose ages are less than *max*. Those tokens correspond to B_1 in the definition of r. We call $stable(M)$ the *stable period* of M.

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

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

2. If $B_0 \neq \varepsilon$, i.e., there are some tokens whose ages do not exceed *max* and whose fractional parts are equal to zero. We divide the tokens in B_0 into two multisets: *young tokens* whose ages are strictly less than *max*, and *old tokens* whose ages are equal to *max*). The stable period $stable(M)$ here is the point interval $[0:0]$. Suppose that we let time pass by an amount $\delta: 0 < \delta < 1-x$, 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 $M \longrightarrow_\delta M_1$ then $M_1 \models r_1$ where r_1 is of the form $(\varepsilon, young \bullet w, B_{max} \odot old)$. Here, *young* and *old* are sub-multisets of B_0 such that $young(p(n)) = B_0(p(n))$ if $n < max$, and $old(p) = B_0(p(max))$. Since the fractional parts of the tokens in *young* are smaller than all other tokens, we put *young* first in the second component of the region. Also, the ages of the tokens in *old* are now strictly greater than *max*, so they are added to the third component of the region.
3. If $B_0 = \varepsilon, w = \varepsilon$, all tokens have age greater than *max*. Now, if we let time pass by any amount $\delta \geq 0$ and $M \longrightarrow_\delta M_1$, then $M_1 \models r$. When all tokens reach age of *max*, aging of tokens becomes irrelevant.

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

Example 5. For the TPN in Figure 1, $max = 7$. Consider a region $r = ([Q(7), R(6)], [S(6), Q(5)] \bullet [S(6), Q(5)], \varepsilon)$. We show a few regions which are reached by letting time pass.

– $(\varepsilon, [R(6)] \bullet [S(6), Q(5)] \bullet [S(6), Q(5)], [Q])$
– $([S(7), Q(6)], [R(6)] \bullet [S(6), Q(5)], [Q])$
– $(\varepsilon, [Q(6)] \bullet [R(6)] \bullet [S(6), Q(5)], [Q, S])$

– $([S(7),Q(6)],[Q(6)] \bullet [R(6)],[Q,S])$, etc

Now, we formalize the informal analysis above as a number of computation rules on region generators. First, we introduce some notations. Let ϕ be an mlg of the form $\{a_1,\ldots,a_k\}^{\circledast} \odot a_{k+1} \odot \cdots \odot a_{k+\ell}$. Notice that, by the normal form defined in Section 5, we can always write ϕ in this form. We define $\#\phi$ 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 $\#\phi = (B,B')$. 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 $\#young(\phi) = (B_1,B_1')$ and $\#old(\phi) = (B_2,B_2')$, then

– $B(p(n)) = B_1(p(n))$ 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 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 $\{p_1(n_1),\ldots p_k(n_k)\}^{\circledast} \odot p_{k+1}(n_{k+1}) \odot \cdots \odot p_{k+\ell}(n_{k+\ell})$. We use ϕ^{+1} to denote the mlg $\{p_1(n_1+1),\cdots,p_k(n_k+1)\}^{\circledast} \odot p_{k+1}(n_{k+1}+1) \odot \cdots \odot p_{k+\ell}(n_{k+\ell}+1)$. That is, we replace each occurrence of a pair $p(n)$ in the representation of ϕ by $p(n+1)$.

For a word atomic expression (an mlg) $\phi = \{a_1,\ldots,a_k\}^{\circledast} \odot a_{k+1} \odot \cdots \odot a_{k+\ell}$, 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 $i : 1 \le i \le k$.

We are 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 $(\varepsilon, \psi \bullet \phi, \phi_{max})$. Here the last element in the second component of the region generator is is an atomic expression (an mlg). We add the region generator $\left(\phi^{+1}, \psi, \phi_{max}\right)$ to W. This step corresponds to one rotation according to case 1 in the analysis above.
– If θ is of the form $(\varepsilon, \psi \bullet \{\phi_1,\ldots,\phi_k\}^*, \phi_{max})$. 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 fractional parts. We add

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

to V, and

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

to W, for $i : 1 \leq i \leq k$. Here, *Old* is the union of the sets of symbols occurring in the mlgs $old(\phi_1^{+1}), \ldots, old(\phi_k^{+1})$, i.e $Old = sym(old(\phi_1^{+1})) \cup \cdots \cup sym(old(\phi_k^{+1}))$. This step corresponds to performing a sequence of rotations of the forms of case 1 and case 2 above.
Notice that we add one of the newly generated region generators directly to V (and its "successor" to W). This is done in order to avoid an infinite loop where the same region generator is generated all the time.

- If θ is of the form $(\phi_0, \psi, \phi_{\max})$, where $\phi_0 \neq \varepsilon$. We add a region generator $(\varepsilon, young(\phi_0) \bullet \psi, \phi_{\max} \odot old(\phi_0))$ to W. This step corresponds to one rotation according to case 2 in the analysis above.
- If θ is of the form $(\varepsilon, \varepsilon, \phi_{\max})$, i.e., all tokens have ages which are strictly greater than *max*, then we do not add any elements 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 of case 4).

Example 6. For the TPN in Figure 1, where $max = 7$. Consider an input region generator $\theta = (Q(7) \odot R(6), \{S(6), Q(5)\}^*, \varepsilon)$. We show a few region generators computed by $Post_{Time}(\theta)$.

- $(\varepsilon, R(6) \bullet \{S(6) \odot Q(5)\}^*, Q)$
- $(S(7) \odot Q(6), \{Q(6)\}^* \bullet R(6) \bullet \{S(6) \odot Q(5)\}^*, \{S\}^{\circledast} \odot Q)$
- $(R(7), \{Q(6)\}^*, \{S\}^{\circledast} \odot Q)$
- $(\varepsilon, \{Q(6)\}^*, \{S\}^{\circledast} \odot Q \odot R)$, etc

9.2 $Post_t$

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 $[\![\theta]\!]$ by firing a discrete transition t, i.e we compute a function $[\![Post(\theta)]\!] = \{M' \mid \exists M \in [\![\theta]\!]. M \longrightarrow_t M'\}$.

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

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

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

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

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

- If $p(n) \in S$, then $\phi - p(n) = \phi$. 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 $\phi - p(n) = S^{\circledast} \odot a_1 \odot \cdots \odot a_{i-1} \odot a_{i+1} \odot \cdots \odot a_\ell$.
- 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 $\psi + p(n)$ 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 $\psi_1 \bullet (\phi + p(n)) \bullet \psi_2 \in (\psi + p(n))$.
2. For each ψ_1, ψ_2 and $\psi = \psi_1 \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet \psi_2$, we have for $i : 1 \le i \le k$, $\psi_1 \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet (\phi_i + p(n)) \bullet \{\phi_1, \cdots, \phi_k\}^* \bullet \psi_2 \in (\psi + p(n))$.
3. For each ψ_1 and ψ_2 with $\psi = \psi_1 \bullet \psi_2$, we have $\psi_1 \bullet p(n) \bullet \psi_2 \in (\psi + p(n))$.

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 $\psi - p(n)$, 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 $\phi_i - p(n)$ is defined for $i : 1 \le i \le k$, then $\psi - p(n) = \{\psi\}$.
- Otherwise, the set $\psi - p(n)$ 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 $\phi' \in (\phi - p(n))$. Notice that ψ may be empty.

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

We recall that, in a TPN, the effect of firing a transition is to remove tokens from the input places and add tokens to the output places. Furthermore, the tokens which are added or removed should have ages in the corresponding intervals. The effect of of firing transitions from the set of markings characterized by a region generator $\theta = (\phi_0, \psi, \phi_{max})$ can therefore be defined by the following operations.

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

- $(\phi_0 - p(N), \psi, \phi_{\max})$ where each $n \in N$ is a natural number in the interval I. Intuitively, if the age of the token that is removed has a zero fractional part, then N contains the valid choices of integral part.
- $(\phi_0, \psi', \phi_{\max})$ such that $\psi' \in \psi - p(N)$, where $N = \{n | \, n \in \mathbb{N} \wedge x \leq n < y\}$ i.e., each n is a valid choice of integral part for the age of the token if it has a non-zero fractional part.
- $(\phi_0, \psi, \phi_{\max} - p)$ if I is of the form $(x : \infty)$, i.e., the age of the token may be greater than *max*.

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

We extend definition of subtraction and addition for subtracting a set of tuples $p(I)$ in the obvious manner. For a set of region generators Θ, we define $\Theta + p(I) = \bigcup_{\theta \in \Theta} (\theta + p(I))$. 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 $\{p(I) | \, In(t,p) = I\}$ and the set of output arcs $\mathcal{A}_{out}(t)$ be given by $\{p(I) | \, Out(t,p) = I\}$.

We define, $Post_t(\theta) = (\theta - \mathcal{A}_{in}(t)) + \mathcal{A}_{out}(t)$.

Example 7. For the TPN in Figure 1, consider an input region generator $(\varepsilon, Q(3), \varepsilon)$ and a transition b. We show the region generators computed by the above algorithm.

- $(\varepsilon, R(0) \bullet S(1), \varepsilon)$
- $(\varepsilon, S(1) \bullet R(0), \varepsilon)$
- $(\varepsilon, S(1) \odot R(0), \varepsilon)$

10 Acceleration

In this section, we explain how to accelerate the firing of a single transition interleaved with timed transitions from a region generator θ. Given an input region generator θ_{in} and a transition t, we want to compute a finite set $Accel_t(\theta_{in})$ of region generators such that $[\![Accel_t(\theta_{in})]\!] = \{M' | \, \exists M \in [\![\theta_{in}]\!]. \, M(\longrightarrow_{Time} \cup \longrightarrow_t)^* M'\}$. 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 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 8 computes both $Post_{Time}$ and $Step_t$ during each iteration, we are guaranteed that all region generators in $Accel_t(\theta)$ will eventually be produced.

To define $Step_t$ we need some preliminary definitions.

Given a symbol $a \in A$ and an mlg ϕ over A of the form $S^{\circledast} \odot a_1 \odot \cdots \odot a_\ell$, we say that a is a $\circledast - symbol$ in ϕ if $a \in S$. Recall that, due to existence of normal forms (Section 5),

we can always write ϕ of the above form. 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 $a \in sym(e_i)$ (as defined in Section 9) and e_i is a word star expression.

Intuitively, a is a $*-symbol$ in ψ if it can occur 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 $(x : y)$. We will extend the algorithms to closed intervals in the next subsections.

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 $\phi \odot \{p(n)\}^{\circledast}$.

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

- If $\psi = \varepsilon$, 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 $p(I)$ where $I = (x : y)$, we define $\theta \uplus p(I) = \left(\phi_0 \odot S_1{}^{\circledast},\ \psi \uplus S_2,\ \phi_{max} \odot \{p_{max}\}^{\circledast}\right)$ where

- $S_1 = \{p(n) |\ n \in \mathbb{N}\ \wedge\ x < n < y\}$.
- $S_2 = \{p(n) |\ n \in \mathbb{N}\ \wedge\ x \leq n < y\}$.
- $p_{max} = p$ if $y = \infty$, $p_{max} = \varepsilon$ otherwise.

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

Acceleration Criteria: 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 holds.

1. Given $In(t,p)=I$, $\theta - p(I) \neq \emptyset$. Intuitively, for a Type 2 place, there is initially at least one token of the "right age" for firing t.
2. $In(t,p) \cap Out(t,p)$ is a non-empty interval. Intuitively, a token generated as output in any firing may be re-used as an input for the next firing.

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

Acceleration: Let $\mathcal{A}_{in}(t), \mathcal{A}_{out}(t)$ be the set of input and output arcs as defined in Section 9. 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 $\Theta = (\theta - \mathcal{A}_{in}(t)) + T_2$ where $T_2 = \{p(I) | \ p \text{ is of Type } 2 \wedge p(I) \in \mathcal{A}_{out}(t)\}$ 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). Finally, we compute

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

Example 8. Consider the TPN in Figure 1. Let $\theta = \left(\{Q(4)\}^{\circledast}, S(2), \varepsilon\right)$ and consider transition b. Then, $Step_b(\theta)$ gives a new region generator $\left(\{Q(4)\}^{\circledast}, \left\{\{R(0),S(1)\}^{\circledast}\right\}^{*} \bullet (S(2) \odot R(0) \odot S(1)) \bullet \left\{\{R(0),S(1)\}^{\circledast}\right\}^{*}, \varepsilon\right)$.

General Case

Now, we will extend the above algorithm for the case when some of the intervals are closed to the left, (or) and closed to the right. We say that an interval is trivial if it is of the form $[x:x]$.

First, we show the result of $\theta \uplus p(I)$ for non-trivial intervals. In all the cases, $\theta \uplus p(I)$ is similar to the case when I is an open interval, except the form of S_1. For each non-trivial interval I, S_1 is given by the set $\{p(n) | \ n \in I\}$.

Given a set $\mathcal{A} = \{p_1(I_1), \ldots, p_k(I_k)\}$ of non-trivial intervals of any form, we define $\theta \uplus \mathcal{A} = \theta \uplus p_1(I_1) \uplus \ldots \uplus p_k(I_k)$.

Next consider a trivial interval. The operation $\uplus$ for mlgs and wlgs remains same as described before. But, for a region generator θ, we also consider the passage of little bit (explained later) of time in the operation $\uplus$.

In fact, $\theta \uplus p(I)$ is given by one of the following forms if $I = [x : x]$ (trivial). Let $S = \{p(x)\}$ and $\phi' = S^{\circledast}$. If $x < max$, we let $\phi'_1 = \phi'$ and $\phi'_2 = \varepsilon$. Otherwise, (if $x = max$) we let $\phi'_1 = \varepsilon$ and $\phi'_2 = \{p\}^{\circledast}$.

1. If $\theta = \left(\phi_i^{+1}, \{young(\phi_1^{+1}), \ldots, young(\phi_k^{+1})\}^* \bullet \psi \bullet \{\phi_1, \ldots, \phi_k\}^*, \phi_{max}\right)$ where $\{sym(old(\phi_1^{+1}) \cup \cdots \cup old(\phi_k^{+1})\}^{\circledast} \subseteq \phi_{max}$ and $i : 1 \le i \le k$, then we have

   $$\theta \uplus p(I) = \left(\phi_i^{+1} \odot \phi', \{young(\phi_1^{+1}) \odot \phi'_1, \ldots, young(\phi_k^{+1}) \odot \phi'_1\}^* \bullet \psi \bullet \{\phi_1, \ldots, \phi_k\}^*, \phi_{max} \odot \phi'_2\right)$$

   Intuitively, θ is the result of $Post_{Time}$ and there is a star expression at the end of the wlg. From such a region generator, there can be an an unbounded sequence of firing t and timed transitions. Consider a marking $M \in \theta$ which has tokens satisfying ψ. The amount of time δ passed in above-mentioned timed transition is given by a positive real number $\delta < 1 - y$ where y is the highest fractional part of the tokens in ψ. The operation $\uplus$ adds unboundedly many tokens $p(x)$ at each instant $\delta : 0 \le \delta < 1 - y$. Such sequence can be executed due to the assumption of non-trivial input intervals. Tokens in Type 2 places do not get disabled due to letting time pass and we can still continue such sequence of transitions.
2. If θ is not of the above form, but $\theta = (\phi_0, \psi, \phi_{max})$. Moreover, $\phi_0 \neq \varepsilon$. Then we have

   $$\theta \uplus p(I) = \left(\phi_0 \odot \phi', \psi, \phi_{max}\right)$$

   Intuitively, we are not allowed to let time pass between firing transitions, since this will destroy the form of the region generator. Therefore, we can only add tokens of age exactly x.
3. If $\theta = (\phi_0, \psi, \phi_{max})$ are not of the above forms and $\phi_0 = \varepsilon$, then we have

   $$\theta \uplus p(I) = \left(\phi', \{\phi'_1\}^* \bullet \psi, \phi_{max} \odot \phi'_2\right)$$

   Intuitively, we can fire unboundedly many discrete transitions followed by a letting non-zero amount of time pass such that the tokens with maximum fractional part do not get their integral parts changed.
   Notice that either $\phi'_1 = \varepsilon$ (when $x = max$) or $\phi'_2 = \varepsilon$ (when $x < max$).

We also need to extend the definition of $\uplus$ for a set of arcs with trivial intervals. This is due to the fact that each of the token in a trivial interval needs to be put together in the first part of the region generator, before elapsing any time. Thus, for a set of trivial intervals $\mathcal{A}$, we define a variant of $\uplus$ as $\uplus'$. Given $\mathcal{A} = \{p_1([n_1 : n_1]), \ldots, p_k([n_k : n_k])\}$, we define $\theta \uplus' \mathcal{A}$ as follows. Let $S = \{p(n) |\ p([n : n]) \in \mathcal{A}\}$, $S_1 = \{p(n) |\ p([n : n]) \in \mathcal{A} \wedge n < max\}$ and $S_2 = \{p |\ p([max : max]) \in \mathcal{A}\}$. The computation rules for $\theta \uplus' \mathcal{A}$ with $\mathcal{A}$ being a set of trivial intervals are same as those in the definition of $\theta \uplus p([x : x])$, but the definitions of ϕ', ϕ'_1, ϕ_2 change slightly. For $\uplus'$, we have $\phi' = S^{\circledast}$, $\phi'_1 = S_1{}^{\circledast}$ and $\phi'_2 = S_2{}^{\circledast}$.

Next, we extend $\uplus$ for a set $\mathcal{A}$ of pairs of the form $p(I)$ where I may be trivial or non-trivial. First we divide the set $\mathcal{A}$ according to the triviality of the intervals into two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{A}_1 = \{p(I) |\ I$ is trivial and $p(I) \in \mathcal{A}\}$ and $\mathcal{A}_2 = \mathcal{A} \setminus \mathcal{A}_1$. We define

$$\theta \uplus \mathcal{A} = (\theta \uplus' \mathcal{A}_1) \uplus \mathcal{A}_2$$

Acceleration Criteria: It is trivial to change the acceleration criteria (only first criterion of an input place to be of Type 1 needs change), depending on the form of the interval on the input arcs. Notice that the input intervals still have to be non-trivial.

Acceleration: The function $Step_t$ is same as that for the open intervals.

11 Termination of $Accel_t$

In this section, we show that the algorithm $Accel_t$ always terminates.

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

Lemma 5. *The set of region generators computed by $Accel_t(\theta)$ is finite.*

Rest of the section is devoted to the proof of the above lemma.

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

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

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 6. *There is a bound N such that for all region generator $\theta' \in Accel_t(\theta)$, $|\theta'| \le N$.*

Notice that Lemma 5 directly follows from Lemma 6.

Now, we show that there is indeed such a bound N as claimed in Lemma 6.

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 first and the second part of the region generator can belong to Type 2 places and each of them can be removed and inserted again as a new atomic expression anywhere in the region generator. In particular, the new tokens in Type 2 place can have a fractional part common with some other token belonging to some star expression. Recall that $\{\phi_1, \cdots, \phi_k\}^* + p(n) = \{\phi_1, \cdots, \phi_k\}^* \bullet (\phi_i + p(n)) \bullet \{\phi_1, \cdots, \phi_k\}^*$ for $i : 1 \le i \le k$. It may also be the case that the new token has totally different fractional part than all other tokens, such that given $\psi = \psi_1 \bullet \psi_2$, we have $\psi_1 \bullet p(n) \bullet \psi_2 \in (\psi + p(n))$. 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. Thus, we define the maximum possible number of word expressions, k_1 as $2*|\theta|+1$.

Now, there can be symbols in the different word expressions of θ which are 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 output tokens added during accelerated addition and these new output tokens will be placed with some of the old output tokens. Thus maximum number of symbols in each of the word expressions of any region generator $\theta' \in Accel_t(\theta)$ 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 new output tokens which can be present together in a word atomic expression.

Therefore, we can say that maximum number of symbols in each word expression of any θ' is maximally (roughly) 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

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

where 2 corresponds to the first and the third part of θ'.

This is really the maximum bound due to the fact that we always keep normalised mlgs and wlgs, and the accelerated addition to a star expression (word or multiset) does not increase the number of word expressions.

12 Experimental Results

We have implemented a prototype based on our algorithm and used our prototype to verify a parameterized version of the Fischer's protocol [SBM92].

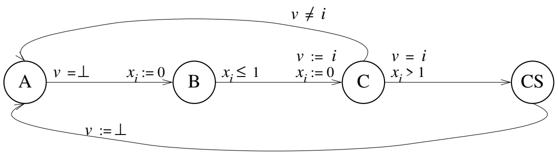

Fig. 2. One process in Fischer's Protocol for Mutual Exclusion

Protocol Description: The protocol consists of processes running a code, which is graphically described in Figure 2. Each process i has a local clock, x_i, and a control state, which assumes values in the set $\{A, B, C, CS\}$ where A is the initial state and CS is the critical section. The processes read from and write to a shared variable v, whose value is either $\perp$ or the index of one of the processes.

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 $\perp$.

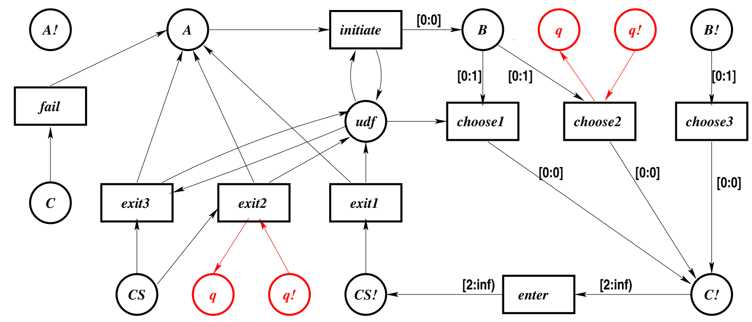

Fig. 3. TPN model for Parameterized version of Fischer's Protocol for Mutual Exclusion

Parameterized Model of Fischer's Protocol [AN01]: Fischer's protocol is intended to guarantee mutual exclusion in a concurrent system consisting of arbitrary number of processes. Figure 3 shows a timed Petri net model of the protocol. Tokens in the places A, B, C, CS, $A!$, $B!$, $C!$ and $CS!$ represent processes running the protocol. We use q to denote any of the process states A, B, C, CS. The places marked with ! represent that the value of the shared variable is the index of the process modeled by the token in that place. We use a place *udf* to represent that the value of the shared variable is $\perp$. The critical section is modelled by the places CS and $CS!$, so mutual exclusion is satisfied when the total number of tokens in those places is less than two.

Verification Result and Abstract Graph: Our program computed the reachability set for Fischer's protocol and proved that mutual exclusion is guaranteed in the protocol. In order to prove the mutual exclusion property, we checked the membership of a set of markings M_{fin} (characterizing an upward closed set of bad states) in the set of reachable states. We specify markings with two tokens in $CS, CS!$ as the bad markings. We use $\left(\{A(0), A(1)\}^{\circledast} \odot udf(0), \{\{A(0)\}^{\circledast}\}^{*}, \{A\}^{\circledast}\right)$ 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. Table 1 shows the region generators which are reachable from θ_{init}. The implementation takes 26s and 35MB memory to compute the invariant consisting of 12 region generators and to prove mutual exclusion, on an Intel Pentium III 1Ghz processor with 256MB RAM.

s_1	$\left(\{A(0),A(1),B(0),B(1)\}^{\circledast} \odot udf(0), \left\{\{A(0),B(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_2	$\left(\{A(0),A(1),B(0),B(1)\}^{\circledast} \odot udf(1), \left\{\{A(0),B(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_3	$\left(\{A(0),A(1),B(0),B(1)\}^{\circledast}, \left\{\{A(0),B(0)\}^{\circledast}\right\}^{*} \bullet (\{A(0),B(0)\}^{\circledast} \odot udf(0)) \bullet \left\{\{A(0),B(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_4	$\left(\{A(0),A(1),B(0),B(1)\}^{\circledast}, \left\{\{A(0),B(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast} \odot udf\right)$
s_5	$\left(\{A(0),A(1),B(0),B(1),C(0),C(1)\}^{\circledast} \odot C!(0), \left\{\{A(0),B(0),C(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_6	$\left(\{A(0),A(1),B(1),C(1)\}^{\circledast}, \left\{\{A(0)\}^{\circledast}\right\}^{*} \bullet \{A(0),B(0),C(0)\}^{\circledast} \odot C!(0) \bullet \left\{\{A(0),B(0),C(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_7	$\left(\{A(0),A(1),B(1),C(1)\}^{\circledast} \odot C!(1), \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_8	$\left(\{A(0),A(1)\}^{\circledast}, \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast} \odot C!\right)$
s_9	$\left(\{A(0),A(1)\}^{\circledast} \odot CS!(0), \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_{10}	$\left(\{A(0),A(1)\}^{\circledast} \odot CS!(1), \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_{11}	$\left(\{A(0),A(1)\}^{\circledast}, \left\{\{A(0)\}^{\circledast}\right\}^{*} \bullet \{A(0)\}^{\circledast} \odot CS!(0) \bullet \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast}\right)$
s_{12}	$\left(\{A(0),A(1)\}^{\circledast}, \left\{\{A(0)\}^{\circledast}\right\}^{*}, \{A,B,C\}^{\circledast} \odot CS!\right)$

Table 1. Set of reachable region generators in Fischer's protocol.

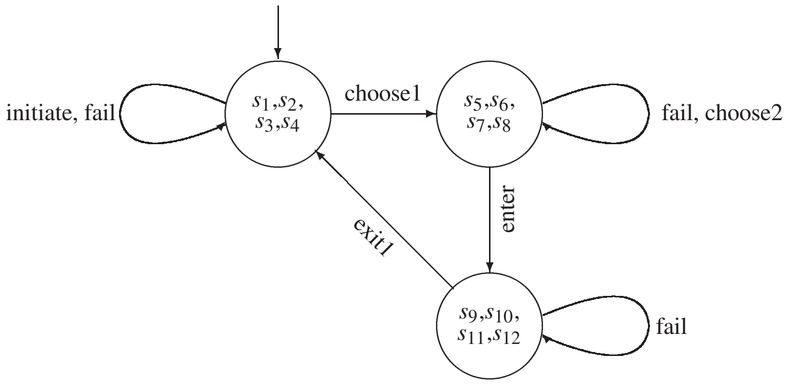

Fig. 4. Minimized version of the abstract graph for Fischer's protocol

We employed our tool to generate the abstract graph (a finite-state abstract model) of the TPN model of Fischer's protocol. This graph consists of 12 states (corresponding to region generators $s_1,\ldots,s_{12}$) and 54 transitions. We used *The Concurrency Workbench* [CPS93] to minimize the abstract graph according to the weak bisimilarity relation where the set of observable actions are given by the discrete transitions in the Fischer's protocol and timed transitions are considered as unobservable actions. Figure 4 shows the minimized finite state labelled transition system with 3 states and 8 transitions. Each state in the minimized version of the abstract graph is annotated by the region generators which are weakly bisimilar.

13 Conclusions and Future Research

We have proposed a formalism, called *region generators* for representing (potentially infinite) downward closed sets of markings of timed Petri nets. We show that region generators allow computing the basic operations involved in the forward reachability analysis. We have augmented the reachability analysis procedure with acceleration in order to make it terminate more often on practical examples. We have built a prototype based on the representation and used the prototype for verifying a parameterized description of Fischer's protocol.

There are a number of interesting directions for future research:

- In this paper, 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.
- In our model of TPNs we assume a lazy behaviour. It is well-known that checking safety properties is undecidable for TPNs with *urgent* behaviours even if the net is safe (bounded). Therefore, designing acceleration techniques is of particular interest for urgent TPNs. Notice that downward closure is no longer an exact abstraction if the behaviour is urgent.
- The symbolic representation we use is that of *region generators*. We intend to investigate designing data structures for more efficient application of our methodology. As a first step we intend to investigate using *zone generators* for analyzing behaviours of timed Petri nets. *Zones* are widely used in existing tools for verification of timed automata [LPY97,BDL$^+$01,Yov97]. Each zone generator will then correspond to a large number of region generators (e.g, a state in the minimized abstract automaton of Figure 4), and therefore gives a much more compact representation.
- We aim at developing generic methods for building downward closed languages, in a similar manner to the methods we have developed for building upward closed languages in [AČJYK00,AN00]. This would give a general theory for forward analysis of infinite state systems, in the same way the works [AČJYK00,AN00] are for backward analysis. The simple regular expressions of [ABJ98,ABB01] and the region generators of this paper are examples of data structures which might be developed in a systematic manner within such a theory.

References

[AAB$^+$99] Parosh Aziz Abdulla, Aurore Annichini, Saddek Bensalem, Ahmed Bouajjani, Peter Habermehl, and Yassine Lakhnech. Verification of infinite-state systems by combining abstraction and reachability analysis. In *Proc.* 11th *Int. Conf. on Computer*

Aided Verification, volume 1633 of *Lecture Notes in Computer Science*, pages 146–160, 1999.

[AAB00] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc.* 12^{th} *Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

[ABB01] Parosh Aziz Abdulla, Luc Boasson, and Ahmed Bouajjani. Effective lossy queue languages. In *Proc. ICALP '2001,* 28^{th} *International Colloquium on Automata, Lnaguages, and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 639–651, 2001.

[ABJ98] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *Proc.* 10^{th} *Int. Conf. on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318, 1998.

[ABLS01] A. Annichini, A. Bouajjani, Y. Lakhnech, and M. Sighireanu. Analyzing fair parametric extended automata. In P. Cousot, editor, *Proceedings of the International Symposium on Static Analysis*, volume 2126 of *Lecture Notes in Computer Science*, 2001.

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

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

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

[AJ98] Parosh Aziz Abdulla and Bengt Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Proc. TACAS '98,* 4^{th} *Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 298–312, 1998.

[AN00] Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. LICS' 00* 16^{th} *IEEE Int. Symp. on Logic in Computer Science*, pages 132–140, 2000.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. 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*, pages 53 –70, 2001.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Engineering*, 17(3):259–273, 1991.

[BDL$^+$01] G. Behrmann, A. David, K. G. Larsen, O. Möller, Paul Pettersson, and Wang Yi. UPPAAL - present and future. In *Proc. CDC'2001, the 40th IEEE Conference on Decision and Control*, 2001.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In Alur and Henzinger, editors, *Proc.* 8^{th} *Int. Conf. on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, 1996.

[BGWW97] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *Proc. of the Fourth International Static Analysis Symposium*, Lecture Notes in Computer Science. Springer Verlag, 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. In *Proc. ICALP '97, 24^{th} International Colloquium on Automata, Lnaguages, and Programming*, volume 1256 of *Lecture Notes in Computer Science*, 1997.

[BLO98] S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems automatically and compositionally. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 1998.

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

[CPS93] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. A semantics based tool for the verification of concurrency systems. *ACM Trans. on Programming Languages and Systems*, 15:36–72, 1993.

[DR00] G. Delzanno and J. F. Raskin. Symbolic representation of upward-closed sets. In *Proc. TACAS '00, 6^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *Lecture Notes in Computer Science*, pages 426–440, 2000.

[FIS00] A. Finkel, S. Purushothaman Iyer, and G. Sutre. Well-abstracted transition systems. In *Proc. CONCUR 2000, 11^{th} Int. Conf. on Concurrency Theory*, pages 566–580, 2000.

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

[GMMP91] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzè. A unified high-level Petri net formalism for time-critical systems. *IEEE Trans. on Software Engineering*, 17(2):160–172, 1991.

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

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, 1993.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

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

[LBBO01] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc. TACAS '01, 7^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

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

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Theoretical Informatics (LATIN'2000)*, number 1776 in Lecture Notes in Computer Science, 2000.

[MF76] P. Merlin and D.J. Farber. Recoverability of communication protocols - implications of a theoretical study. *IEEE Trans. on Computers*, COM-24:1036–1043, Sept. 1976.

[RP85] R. Razouk and C. Phelps. Performance analysis using timed Petri nets. In *Protocol Testing, Specification, and Verification*, pages 561–576, 1985.

[SBM92] F. B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. In de Bakker, Huizing, de Roever, and Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, 1992.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. LICS '86, 1^{st} IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

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

A Appendix - Proofs of Lemmas

Proof of Theorem 1

First, we show some auxiliary lemmas.

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

Proof. Let x be of the form $\left[a_1^{l_1}, \ldots, a_m^{l_m}\right]$ where a^l denotes l number of a's. 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 $\mathcal{L}(e_1) = \varepsilon$. We define Φ as a set of mlgs ϕ_i of the form $e_1 \odot a_1^{l_1} \odot \cdots \odot a_i^{l_i - 1} \odot \cdots \odot a_m^{l_m}$ where $i : 1 \leq i \leq m$.

First we show that $y \in \mathcal{L}(\Phi) \implies x \not\leq^m y$ by contraposition. Assume $x \leq^m y$. We prove that $y \notin \mathcal{L}(\phi_i)$ for each $i : 1 \leq i \leq m$. From the definition of $\leq^m$, we know that y is of the form $x \odot z$ where $z \in A^{\circledast}$. From definition of ϕ_i, $a_i^{l_i} \notin \mathcal{L}(\phi_i)$ for each $i : 1 \leq i \leq m$. Therefore, $x \odot z \notin \mathcal{L}(\phi_i)$ for each i. Therefore, $y \notin \mathcal{L}(\Phi)$.

Next, we prove that $x \not\leq^m y \implies y \in \mathcal{L}(\phi)$. Let x' be the largest proper sub-multiset of x which satisfies $x' \leq^m y$. This means that y is of the form $x' \odot z$ where z is a multiset over $A \setminus \{a_1, \ldots, a_m\}$. Thus $z \in \mathcal{L}(e_1)$. Since x' is a proper submultiset of x, $x' \in \mathcal{L}\left(a_1^{l_1} \odot \cdots \odot a_i^{l_i - 1} \odot \cdots \odot a_m^{l_m}\right)$ where $i : 1 \leq i \leq m$. Thus, $y \in \mathcal{L}(\Phi)$.

Lemma A.2 For set of mlgs Φ_1, Φ_2, there is a set of mlgs $\Phi_1 \cap \Phi_2$ such that $\mathcal{L}(\Phi_1 \cap \Phi_2) = \mathcal{L}(\Phi_1) \cap \mathcal{L}(\Phi_2)$.

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. $\phi_1 \cap \phi_2 = \varepsilon$, 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, $\phi_1 \cap \phi_2 = \varepsilon$.
- 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 $\phi_1 = e_{11} \odot \cdots \odot e_{1k}$ and $\phi_2 = e_{21} \odot \cdots \odot e_{2m}$. 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 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.
   $\phi_{ij} = (e_{1i} \cap e_{2j}) \odot (\phi_{1i} \cap \phi_{2j})$.
2. if e_{1i} is an atomic expression and e_{2j} is a star expression.
   $\phi_{ij} = (e_{1i} \cap e_{2j}) \odot (\phi_{1i} \cap \phi_2)$.
3. if e_{1i} is a star expression and e_{2j} is an atomic expression.
   $\phi_{ij} = (e_{1i} \cap e_{2j}) \odot (\phi_1 \cap \phi_{2j})$.

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

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.

Now, if $\Phi_1 = \{\phi_1, \cdots, \phi_m\}$ and $\Phi_2 = \{\phi'_1, \cdots, \phi'_n\}$, 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 $j_1 \leq j \leq n$.

To prove theorems 1 and 3, we use Higman's theorem [Hig52] which includes the following:

- For any finite alphabet A and for any infinite sequence of $x_1, x_2, ...$ of multisets over A, there are $i < j$ such that $x_i \leq^m x_j$. $(A^{\circledast}, \leq^m)$ is called a well-quasi-ordered set.
- Words and multisets over a well-quasi-ordered set are also well-quasi-ordered.

Finally, we assume a downward closed language L. If $L = \emptyset$, then $L = \mathcal{L}(\Phi)$ 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 from Higman's theorem [Hig52]. Thus, a multiset $y \in L$, if and only if $M_i \not\leq^m y$ 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, it follows that there are sets of mlgs, $\Phi_1, \ldots, \Phi_n$ such that $L = \mathcal{L}(\Phi_1) \cap \cdots \cap \mathcal{L}(\Phi_n)$. The result follows from Lemma A.2.

Proof of Lemma 3

We want to prove that $\phi \subseteq \{\phi'_1, \cdots, \phi'_m\}$ iff $\phi \subseteq \phi'_i$ for some $i : 1 \leq i \leq m$.

Assume $\phi \not\subseteq \phi'$ for any ϕ' such that $\phi' \in \{\phi'_1, \ldots, \phi'_n\}$.

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

Let k_m be the number of expressions in the longest mlg among $\phi'_1, \cdots, \phi'_n$.

For any mlg ϕ' of the form $e'_1 \odot \cdots \odot e'_k$ for some natural number k where $k \leq k_m$, our induction hypothesis is :

IH: For a mlg $\phi = e_1 \odot \cdots \odot e_m$ with $m \geq 1$, we have $e_1 \odot \cdots \odot e_m \not\subseteq \phi' \implies M_1 \odot \cdots \odot M_m \notin \mathcal{L}(\phi')$ where multisets $M_i \in \mathcal{L}(e_i)$ for $i : 1 \leq i \leq m$.

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. $a \notin \mathcal{L}(e'_i)$ for any $i : 1 \leq i \leq k$. Hence, $a \notin \mathcal{L}(\phi')$.

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 $M(a) = k_m + 1$ for all $a \in \{a_1, \cdots, a_l\}$, i.e $M = [a_1^{k_m+1}, \ldots, a_l^{k_m+1}]$ and $l > 0$. We need to use induction only on k (length of ϕ') to show that $M \notin \mathcal{L}(\phi')$. The base case ($k = 0$) is trivial. For the induction step, we asssume $k > 0$. For each $i : 1 \leq i \leq k$, assuming $\phi' = e'_i \odot \phi''_i$, we show the claim. There are two cases.

- e'_i is atomic. By the induction hypothesis, we have that $[a_1^k, \ldots, a_l^k] \notin \mathcal{L}(\phi''_i)$. Since e'_i is atomic (contains a singleton), $[a_1^{k+1}; \ldots, a_l^{k+1}] \notin \mathcal{L}(\phi')$.
- e'_i is star expression. We know that $\{a_!, \ldots, a_l\}^{\circledast} \not\subseteq e'_i$ (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 $a \notin \mathcal{L}(e'_i)$. This implies that $[a_1, \ldots, a_l] \notin \mathcal{L}(e'_i)$. By the induction hypothesis, we have that $\left[a_1^k, \ldots, a_l^k\right] \notin \mathcal{L}(\phi''_i)$. This implies that $\left[a_1^{k+1}, \ldots, a_l^{k+1}\right] \notin \mathcal{L}(\phi')$.

Therefore, $\left[a_1^{k_m+1}, \ldots, a_l^{k_m+1}\right] \notin \mathcal{L}(\phi')$.

Inductive Step ($m > 1$): Let ϕ be of the form $e_1 \odot \cdots \odot e_m$. We define $M = M_1 \odot \cdots \odot M_m$ 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 $\phi' = \phi'' \odot \phi'''$ where ϕ'' is a minimum mlg which satisfies $e_1 \subseteq \phi''$. Assume that ϕ'' is of the form $e'_i \odot \phi'_i$ 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 $e_1 \subseteq \phi''$, $M_1 \notin \mathcal{L}(\phi'_i)$. Furthermore, we know that $e_2 \odot \cdots \odot e_m \not\subseteq \phi'''$ (otherwise, $\phi \subseteq \phi'$, contradiction). By induction hypothesis, it follows that $M_2 \odot \cdots \odot M_m \notin \mathcal{L}(\phi''')$. Since e'_i is atomic, we infer that $M_1 \odot \cdots \odot M_m \notin \mathcal{L}(\phi')$.
- If e'_i is a star expression. Since ϕ'' is a minimum mlg which satisfies $e_1 \subseteq \phi''$, $M_1 \notin \mathcal{L}(\phi'_i)$. Furthermore, since e'_i is a star expression, we know that $e_2 \odot \cdots \odot e_m \not\subseteq e'_i \odot \phi'''$ (otherwise, $\phi \subseteq \phi'$, contradiction). By induction hypothesis, it follows that $M_2 \odot \cdots \odot M_m \notin \mathcal{L}(e'_i \odot \phi''')$. We infer that $M_1 \odot \cdots \odot M_m \notin \mathcal{L}(\phi')$.

Proof of Theorem 3

First, we show some auxiliary lemmas.

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

Proof. Let x be of the form $M_1 M_2 \ldots M_m$ where $M_1, \ldots, M_m$ are multisets over a finite alphabet A. Let e_i be a star expression $\left\{\phi_1^i, \ldots, \phi_k^i\right\}^*$ where $\phi_1^i, \ldots, \phi_k^i$ are obtained from multiset M_i for each $i : 1 \leq i \leq m$ as shown in Lemma A.1, satisfying that a multiset $M \in \mathcal{L}\left(\left\{\phi_1^i, \ldots, \phi_k^i\right\}\right)$ 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 multiset product ϕ_i such that $M_i \in \mathcal{L}(\phi_i)$ for each $i : 1 \leq i \leq m$.

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

First we show that $y \in \mathcal{L}(\psi) \implies x \not\leq^w y$ by contraposition. Assume $x \leq^w y$. We prove that $y \notin \mathcal{L}(\psi)$. From the definition of $\leq^w$, we know that y is of the form $y_1 \bullet M_1 \bullet y_2 \bullet \cdots M_m \bullet y_{m+1}$ where $y_i \in (A^{\circledast})^*$. From definition of e_i, we know that $M_i \notin \mathcal{L}(e_i)$ and hence, $y_i \bullet M_i \notin \mathcal{L}(e_i)$ for each $i : 1 \leq i \leq m$. This implies that $y_1 \bullet M_1 \bullet y_2 \bullet \cdots \bullet y_m \bullet M_m \notin \mathcal{L}(e_1 \bullet \phi_1 \bullet \cdots \bullet \phi_{m-1} \bullet e_m) = \mathcal{L}(\psi)$, i.e $y \notin \mathcal{L}(\psi)$.

Next, we prove that $x \not\leq^w y \implies y \in \mathcal{L}(\psi)$. Let l be the largest natural number such that $M_1 \ldots M_l \leq^w y$. Obviously, $0 \leq l < m$. This means that y is of the form $y_0 \bullet M_1 \bullet y_1 \bullet M_2 \cdots y_{l-1} \bullet M_l \bullet y_l$, where y_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 y_l (otherwise, we will have $M_1 \cdots M_{l+1} \leq^w y$ violating the maximality of l). This implies that $y_i \in \mathcal{L}(e_{i+1})$ for each $i : 0 \leq i \leq l$, and hence $y \in \mathcal{L}(\psi)$.

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

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 $\psi_2 = \{\phi'_1, \ldots, \phi'_m\}^*$, then $\psi_1 \cap \psi_2 = (\phi_1 \cap \phi'_1 \cup \ldots \cup \phi_k \cap \phi'_m)^*$. 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,
   $\Psi = \{(e_1 \cap e_2) \bullet (\psi'_1 \cap \psi'_2)\ ,\ (\psi_1 \cap \psi'_2)\ ,\ (\psi'_1 \cap \psi_2)\}$
2. e_1 is an atomic expression and e_2 is a star expression.
   $\Psi = \{(e_1 \cap e_2) \bullet (\psi'_1 \cap \psi_2)\ ,\ (\psi_1 \cap \psi'_2)\}$
3. e_1 is a star expression and e_2 is an atomic expression.
   $\Psi = \{(e_1 \cap e_2) \bullet (\psi_1 \cap \psi'_2)\ \ (\psi'_1 \cap \psi_2)\}$
4. e_1 and e_2 are star expressions.
   $\Psi = \{(e_1 \cap e_2) \bullet (\psi_1 \cap \psi'_2)\ ,\ (e_1 \cap e_2) \bullet (\psi'_1 \cap \psi_2)\}$

Consider a downward closed language L of words over multisets. If $L = \emptyset$, then $L = \mathcal{L}(\Psi)$ where $\Psi = \emptyset$. Otherwise, complement of L is upward closed and can be chatacterized by a finite set of words over multisets given by $\{w_1, \ldots, w_n\}$ (from the second condition of Higman's theorem). Thus, a word $y \in L$, if and only if $w_i \not\preceq^w y$ 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, it follows that there are wlgs, $\psi_1, \ldots, \psi_n$ such that $L = \mathcal{L}(\psi_1) \cap \cdots \cap \mathcal{L}(\psi_n)$. The result follows from Lemma A.4.

Proof of Lemma 4

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

Let k' be the number of expressions in the longest wlg among $\psi'_1, \cdots, \psi'_n$. For any word wlg ψ' of the form $e'_1 \bullet \cdots \bullet e'_k$ for some natural number k where $k \leq k'$, we do the following.

Induction hypothesis is :

IH: For a wlg $\psi = e_1 \bullet \cdots \bullet e_m$ with $m \geq 1$, we have $e_1 \bullet \cdots \bullet e_m \not\subseteq \psi' \implies w_1 \bullet \cdots \bullet w_m \notin \mathcal{L}(\psi')$ where words $w_i \in \mathcal{L}(e_i)$ for $i : 1 \leq i \leq m$.

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, $M \notin \mathcal{L}(e'_i)$ for any $i : 1 \leq i \leq k$. Hence, $w \notin \mathcal{L}(\psi')$.

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 need to use induction only on k (length of ψ') to show that $w \notin \mathcal{L}(\psi')$. The base case ($k = 0$) is trivial. For the induction step, we asssume $k > 0$. There are two cases.

- e'_k is atomic. By the induction hypothesis, we have that $(M_1 \bullet \ldots \bullet M_l)^k \notin \mathcal{L}(e'_1 \bullet \cdots \bullet e'_{k-1})$. Since e'_k is atomic, $(M_1 \bullet \ldots \bullet M_l)^{k+1} \notin \mathcal{L}(\psi')$.
- 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 $M_i \notin \mathcal{L}(e'_k)$. This implies that $M_1 \bullet \ldots \bullet M_l \notin \mathcal{L}(e'_k)$. By the induction hypothesis, we have that $(M_1 \bullet \ldots \bullet M_l)^k \notin \mathcal{L}(e'_1 \bullet \cdots \bullet e'_{k-1})$. This implies that $(M_1 \bullet \ldots \bullet M_l)^{k+1} \notin \mathcal{L}(\psi')$.

Therefore, $(M_1 \bullet \ldots \bullet M_l)^{k'+1} \notin \mathcal{L}(\psi')$.

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 $e_1 \nsubseteq \psi'$, 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 $e_1 \subseteq e'_1 \bullet \cdots \bullet e'_{k_1}$, $w_1 \notin \mathcal{L}\left(e'_1 \bullet \cdots \bullet e'_{k_1-1}\right)$. Furthermore, we know that $e_2 \bullet \cdots \bullet e_m \nsubseteq e'_{k_1+1} \bullet \cdots \bullet e'_k$. (otherwise, $\psi \subseteq \psi'$, contradiction). By induction hypothesis, it follows that $w_2 \bullet \cdots \bullet w_m \notin \mathcal{L}\left(e'_{k_1+1} \bullet \cdots \bullet e'_k\right)$. Since e'_{k_1} is atomic, we infer that $w_1 \bullet \cdots \bullet w_m \notin \mathcal{L}(\psi')$.
- If e'_{k_1} is a star expression. Since k_1 is the minimum natural number satisfying $e_1 \subseteq e'_1 \bullet \cdots \bullet e'_{k_1}$, $w_1 \notin \mathcal{L}\left(e'_1 \bullet \cdots \bullet e'_{k_1-1}\right)$. Furthermore, since e'_{k_1} is a star expression, we know that $e_2 \bullet \cdots \bullet e_m \nsubseteq e'_{k_1} \bullet \cdots \bullet e'_k$ (otherwise, $\psi \subseteq \psi'$, contradiction). By induction hypothesis, it follows that $w_2 \bullet \cdots \bullet w_m \notin \mathcal{L}\left(e'_{k_1} \bullet \cdots \bullet e'_k\right)$. We infer that $w_1 \bullet \cdots \bullet w_m \notin \mathcal{L}(\psi')$.

Recent technical reports from the Department of Information Technology

2003-040 Wendy Kress: *Error Estimates for Deferred Correction Methods in Time*

2003-041 Wendy Kress: *A Compact Fourth Order Time Discretization Method for the Wave Equation*

2003-042 Gerardo Schneider: *Invariance Kernels of Polygonal Differential Inclusions*

2003-043 Kajsa Ljungberg, Sverker Holmgren, and Örjan Carlborg: *Simultaneous Search for Multiple QTL Using the Global Optimization Algorithm DIRECT*

2003-044 Dan Wallin, Henrik Johansson, and Sverker Holmgren: *Cache Memory Behavior of Advanced PDE Solvers*

2003-045 Sven-Olof Nyström: *A Polyvariant Type Analysis for Erlang*

2003-046 Martin Karlsson: *A Power-Efficient Alternative to Highly Associative Caches*

2003-047 Jimmy Flink: *Simuleringsmotor för tågtrafik med stöd för experimentell konfiguration*

2003-048 Timour Katchaounov and Tore Risch: *Interface Capabilities for Query Processing in Peer Mediator Systems*

2003-049 Martin Nilsson: *A Parallel Shared Memory Implementation of the Fast Multipole Method for Electromagnetics*

2003-050 Alexandre David: *Hierarchical Modeling and Analysis of Timed Systems*

2003-051 Pavel Krcal and Wang Yi: *Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata*

2003-052 Magnus Svärd and Jan Nordström: *Well Posed Boundary Conditions for the Navier-Stokes Equations*

2003-053 Erik Bängtsson and Maya Neytcheva: *Approaches to Reduce the Computational Cost when Solving Linear Systems of Equations Arising in Boundary Element Method Discretizations*

2003-054 Martin Nilsson: *Stability of the Fast Multipole Method for Helmholtz Equation in Three Dimensions*

2003-055 Martin Nilsson: *Rapid Solution of Parameter-Dependent Linear Systems for Electromagnetic Problems in the Frequency Domain*

2003-056 Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Forward Reachability Analysis of Timed Petri Nets*

2003-057 Erik Berg: *Low-Overhead Spatial and Temporal Data Locality Analysis*

2003-058 Erik Berg: *StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis*

2003-059 Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*

UPPSALA
UNIVERSITET

December 2003
ISSN 1404-3203
http://www.it.uu.se/