Edinburgh Research Explorer

Dense-Timed Petri Nets: Checking Zenoness, Token liveness and Boundedness

DENSE-TIMED PETRI NETS: CHECKING ZENONESS, TOKEN LIVENESS AND BOUNDEDNESS

PAROSH AZIZ ABDULLA [a], PRITHA MAHATA [b], AND RICHARD MAYR [c]

[a] Uppsala University, Department of Information Technology, Box 337, SE-751 05 Uppsala, Sweden
e-mail address: parosh@it.uu.se

[b] University of Newcastle, School of Electrical Engineering and Computer Science, University Drive, Callaghan NSW 2308, Australia
e-mail address: pritha.mahata@newcastle.edu.au

[c] North Carolina State University, Department of Computer Science, Campus Box 8206, Raleigh, NC 27695, USA
URL: http://www4.ncsu.edu/~rmayr

ABSTRACT. We consider *Dense-Timed Petri Nets (TPN)*, an extension of Petri nets in which each token is equipped with a real-valued clock and where the semantics is lazy (i.e., enabled transitions need not fire; time can pass and disable transitions). We consider the following verification problems for TPNs.

(i) *Zenoness:* whether there exists a zeno-computation from a given marking, i.e., an infinite computation which takes only a finite amount of time. We show decidability of zenoness for TPNs, thus solving an open problem from [dFERA00]. Furthermore, the related question if there exist arbitrarily fast computations from a given marking is also decidable.

On the other hand, *universal zenoness*, i.e., the question if *all* infinite computations from a given marking are zeno, is undecidable.

(ii) *Token liveness:* whether a token is *alive* in a marking, i.e., whether there is a computation from the marking which eventually consumes the token. We show decidability of the problem by reducing it to the *coverability problem*, which is decidable for TPNs.

(iii) *Boundedness:* whether the size of the reachable markings is bounded. We consider two versions of the problem; namely *semantic boundedness* where only live tokens are taken into consideration in the markings, and *syntactic boundedness* where also dead tokens are considered. We show undecidability of semantic boundedness, while we prove that syntactic boundedness is decidable through an extension of the Karp-Miller algorithm.

1. Introduction

Petri nets [Pet62, Pet77, Mur89] are one of the most widely used models for analysis and verification of concurrent systems. Many different formalisms have been proposed which extend Petri nets with clocks and real-time constraints, leading to various definitions of *Timed Petri nets (TPNs)*. A complete discussion of all these formalisms is beyond the scope of this paper and the interested reader is referred to the survey by Bowden [Bow96] and a more recent overview in [BCH+05].

In this paper we consider the TPN model used in [AN01] where each token has an age which is represented by a real-valued clock, and the firing-semantics is lazy (like in standard untimed Petri nets). This dense time TPN model of [AN01] is an adaption of the discrete time model of Escrig et al. [RGdFE99, dFERA00].

The main difference between dense time TPN and discrete time TPN is the following. In discrete time nets, time is interpreted as being incremented in discrete steps and thus the ages of tokens are in a countable domain, commonly the natural numbers. Such discrete time nets have been studied in, e.g., [RGdFE99, dFERA00]. In dense time nets, time is interpreted as continuous, and the ages of tokens are real numbers. Some problems for dense time nets have been studied in [AN01, AN02, ADMN04].

In this paper we mainly consider the dense time case. However, we also solve some open questions for discrete time nets, since they follow as corollaries from our more general results on the dense time case.

The main characteristics of our TPN model (i.e., the model of [AN01]) are the following.

- Our TPNs are *not* bounded. The number of tokens present in the net may grow beyond any finite bound.
- Each token has an age which is represented by a real-valued clock, i.e., time is continuous.
- A transition is enabled iff there are enough tokens of the right ages on its input places. The right ages are specified by labeling the input arcs of transitions with time intervals.
- The semantics is *lazy*, just like in standard untimed Petri nets. This means that an enabled transition need not fire immediately. It is possible that more time will pass and disable the transition again. (This is in contrast to many other classes of Petri nets with time, which have an eager semantics where transitions must fire when they are enabled; see [BCH+05] for an overview.)
- When a transition fires, the clocks of the consumed tokens are not preserved. Tokens which are newly created by a transition have their own new clocks.

The formal definition of this TPN model is given in Section 2.

TPN can, among other things, model parameterized timed systems (systems consisting of an unbounded number of timed processes) [AN01].

Our TPN model is computationally more powerful than timed automata [AD90, AD94], since it operates on a potentially unbounded number of clocks. In particular, TPN subsume normal untimed Petri nets w.r.t. the semantics of fired transition sequences, while finite timed automata do not subsume Petri nets. Furthermore, both the reachability problem [RGdFE99] and several liveness problems [dFERA00, AN02] are undecidable for TPNs (even in the discrete time case).

Most verification problems for TPNs are extensions of both classical problems previously studied for standard (untimed) Petri nets, and problems for finite-state timed models like timed automata. We consider several verification problems for TPNs.

Zenoness. A fundamental progress property for timed systems is that it should be possible for time to *diverge* [Tri99]. This requirement is justified by the fact that timed processes cannot be infinitely fast. Computations violating this property are called *zeno*. Given a TPN and a marking M, we check whether M is a *zeno-marking*, i.e., whether there is an infinite computation from M with a finite duration. The zenoness problem is solved in [AD90, Alu91] for timed automata using the region graph construction. Since region graphs only deal with a finite number of clocks, the algorithm of [AD90, Alu91] cannot be extended to check zenoness for TPNs. In Section 3, we solve the zenoness problem for TPNs. To do this, we consider a subclass of transfer nets [FS98] which we call *simultaneous-disjoint transfer net (SD-TN)*. This class is an extension of standard Petri nets, in which we also have *transfer* transitions which may move all tokens in one place to another with the restriction that (a) all such transfers take place simultaneously and (b) the sources and targets of all transfers are disjoint.

Given a TPN N, we perform the following three steps:

- Derive a corresponding SD-TN N'.
- Characterize the set of markings in N' from which there are infinite computations[1].
- Re-interpret the set computed above as a characterization of the set of zeno-markings in N.

In fact, the above procedure solves a more general problem than that of checking whether a given marking is zeno; namely it gives a symbolic characterization of the set of zeno-markings.

The zenoness problem was left open in [dFERA00] both for dense TPNs (the model we consider in this paper) and for discrete TPNs (where behavior is interpreted over the discrete time domain). The construction given in this paper considers the more general dense time case. The construction can easily be modified (in fact simplified) to deal with the discrete time case. (In the discrete time case, unlike for dense time, every zeno computation must have an infinite suffix that takes zero time.)

Arbitrarily Fast Computations. In Section 5 we consider a question related to zenoness: 'Given a marking M, is it the case that for every $\epsilon > 0$ there is an M-computation which takes at most ϵ time?' This is a stronger requirement than zenoness, and we call markings which satisfy it *allzeno*-markings. Like for zeno-markings, one can compute a symbolic characterization of the set of allzeno-markings, and thus the problem is decidable.

Markings from which there are computations which take no time at all are called *zerotime*-markings. For discrete time nets, allzeno-markings and zerotime-markings coincide, but for general dense time nets zerotime-markings are (in general) a strict subset. Again one can compute a symbolic characterization of the set of zerotime-markings.

Universal Zenoness. In the zenoness problem, the question was whether there existed at least one zeno run, i.e., an infinite computation which takes finite time. The universal zenoness problem is the question whether *all* infinite runs are zeno. The negation of this question is the following: Given some marking M, does there exist some non-zero M-computation, i.e., an infinite computation from M which takes an infinite amount of time? In Section 6 we show that this question (and thus universal zenoness) is undecidable, by a reduction from lossy counter machines [May03].

---

[1] In contrast to SD-TN, such a characterization is not computable for general transfer nets [May03].

Token Liveness. Markings in TPNs may contain tokens which cannot be used by any future computations of the TPN. Such tokens do not affect the behavior of the TPN and are therefore called *dead tokens*. We give an algorithm to check, given a token and a marking, whether the token is dead (or alive). We do this by reducing the problem to the problem of *coverability* in TPNs. An algorithm to solve the coverability problem is given in [AN01].

Token liveness for dense TPNs was left open in [dFERA00].

Boundedness. We consider the *boundedness problem* for TPNs: given a TPN and an initial marking, check whether the size of reachable markings is bounded. The decidability of this problem depends on whether we take dead tokens into consideration. In *syntactic boundedness* one considers dead tokens as part of the (size of the) marking, while in *semantic boundedness* we disregard dead tokens; that is we check whether we can reach markings with unboundedly many live tokens. Using techniques similar to [RGdFE99] it can be shown that semantic boundedness is undecidable. On the other hand we show decidability of syntactic boundedness. This is achieved through an extension of the Karp-Miller algorithm where each node represents a region (rather than a single marking). The underlying ordering on the nodes (regions) inside the Karp-Miller tree is a *well quasi-ordering* [Hig52]. This guarantees termination of the procedure.

Decidability of syntactic boundedness was shown for the simpler discrete time case in [dFERA00], while the problem was left open for the dense case.

2. TIMED PETRI NETS AND REGIONS

Timed Petri Nets. 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 labeled with a time-interval whose bounds are natural numbers (or possibly ∞ as upper bound). These intervals can be open, closed or half open. When firing a transition, tokens which are removed (added) from (to) places must have ages lying in the intervals of the corresponding transition arcs.

We use $\mathbb{N}, \mathbb{R}^{\geq 0}, \mathbb{R}^{>0}$ to denote the sets of natural numbers (including 0), nonnegative reals, and strictly positive reals, respectively. For a natural number k, we use $\mathbb{N}^k$ and $\mathbb{N}_\omega^k$ to denote the set of vectors of size k over $\mathbb{N}$ and $\mathbb{N} \cup \{\omega\}$, respectively ($\omega$ represents the first limit ordinal).

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 in both directions and $[w : z)$ is closed to the left and open to the right.

Definition 2.1. For a set A, we use A^* and $A^\odot$ to denote the set of finite words and finite multisets over A, respectively. We view a multiset b over A as a mapping $b : A \mapsto \mathbb{N}$. Sometimes, we write finite multisets as lists with multiple occurrences, so $[2.4^3, 5.1^2]$ represents a multiset b over $\mathbb{R}^{\geq 0}$ where $b(2.4) = 3$, $b(5.1) = 2$ and $b(x) = 0$ for $x \neq 2.4, 5.1$. For multisets b_1 and b_2 over $\mathbb{N}$, we say that $b_1 \leq b_2$ if $b_1(a) \leq b_2(a)$ for each $a \in A$. The multiset union $b = b_1 \cup b_2$ is defined by $b(a) = \max(b_1(a), b_2(a))$ for each $a \in A$ and the multiset intersection $b = b_1 \cap b_2$ is defined by $b(a) = \min(b_1(a), b_2(a))$ for each $a \in A$.

We define $b_1 + b_2$ to be the multiset b where $b(a) = b_1(a) + b_2(a)$, and (assuming $b_1 \leq b_2$) we define $b_2 - b_1$ to be the multiset b where $b(a) = b_2(a) - b_1(a)$, for each $a \in A$.

For a multiset $b : A \mapsto \mathbb{N}$, we write $|b| := \sum_{a \in A} b(a)$ for the number of elements in b. We use $\emptyset$ to denote the empty multiset and ϵ to denote the empty word.

Given a set A with partial order $\leq$, we define a partial order $\leq^w$ on A^* as follows. We have $a_1 \ldots a_n \leq^w b_1 \ldots b_m$ iff there is a subsequence $b_{j_1} \ldots b_{j_n}$ of $b_1 \ldots b_m$ s.t. $\forall k \in \{1, \ldots, n\}. a_k \leq b_{j_k}$.

Given a set A with an ordering $\preceq$ and a subset $B \subseteq A$, B is said to be *upward closed* in A if $a_1 \in B, a_2 \in A$ and $a_1 \preceq a_2$ implies $a_2 \in B$. Given a set $B \subseteq A$, we define the *upward closure* $B \uparrow$ to be the set $\{a \in A | \exists a' \in B : a' \preceq a\}$. A *downward closed* set B and the *downward closure* $B \downarrow$ are defined in a similar manner. We also use $a \uparrow$, $a \downarrow$, a instead of $\{a\} \uparrow$, $\{a\} \downarrow$, $\{a\}$, respectively.

Definition 2.2. [AN01] 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 $In(t, p)$ (respectively $Out(t, p)$) is defined, we say that p is an *input (respectively output) place* of t.

We let *max* denote the maximum integer appearing on the arcs of a given TPN.

A *marking* M of N is a finite multiset over $P \times \mathbb{R}^{\geq 0}$. The marking M defines the numbers and ages of tokens in each place in the net. We identify a token in a marking M by the pair (p, x) representing its place and age in M. Then, $M((p, x))$ defines the number of tokens with age x in place p. Abusing notation again, we define, for each place p, a multiset $M(p)$ over $\mathbb{R}^{\geq 0}$, where $M(p)(x) = M((p, x))$.

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

Transitions: We define two transition relations on the set of markings: timed transition and discrete transition. A *timed transition* increases the age of each token by the same real number. Formally, for $x \in \mathbb{R}^{>0}$, $M_1 \longrightarrow_x M_2$ if $M_2 = M_1^{+x}$. We use $M_1 \longrightarrow_\delta M_2$ to denote that $M_1 \longrightarrow_x M_2$ for some $x \in \mathbb{R}^{>0}$.

We define the set of *discrete transitions* $\longrightarrow_D$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of *firing* the discrete transition t. More precisely, $M_1 \longrightarrow_t M_2$ if the set of input arcs $\{(p, \mathcal{I}) | In(t, p) = \mathcal{I}\}$ is of the form $\{(p_1, \mathcal{I}_1), \ldots, (p_k, \mathcal{I}_k)\}$, the set of output arcs $\{(p, \mathcal{I}) | Out(t, p) = \mathcal{I}\}$ is of the form $\{(q_1, \mathcal{J}_1), \ldots, (q_\ell, \mathcal{J}_\ell)\}$, and there are multisets $b_1 = [(p_1, x_1), \ldots, (p_k, x_k)]$ and $b_2 = [(q_1, y_1), \ldots, (q_\ell, y_\ell)]$ over $P \times \mathbb{R}^{\geq 0}$ such that the following holds:
- $b_1 \leq M_1$
- $x_i \in \mathcal{I}_i$, for $i : 1 \leq i \leq k$.
- $y_i \in \mathcal{J}_i$, for $i : 1 \leq i \leq \ell$.
- $M_2 = (M_1 - b_1) + b_2$.

We say that t is *enabled* in M if there is a b_1 such that the first two conditions are satisfied. A transition t may be fired only if for each incoming arc, there is a token with the right age in the corresponding input place. These tokens will be removed when the transition is fired. The newly produced tokens have ages which are chosen nondeterministically from the relevant intervals on the transitions' output arcs.

We write $\longrightarrow = \longrightarrow_\delta \cup \longrightarrow_D$ to denote all transitions, $\stackrel{*}{\longrightarrow}$ to denote the reflexive-transitive closure of $\longrightarrow$ and $\longrightarrow_D^+$ to denote the transitive closure of $\longrightarrow_D$. It is easy to extend $\stackrel{*}{\longrightarrow}$ for sets of markings. We define $Reach(M) := \{M' | M \stackrel{*}{\longrightarrow} M'\}$ as the set of markings reachable from M.

Computations: Generally, a computation from a given marking is just a (finite or infinite) sequence of enabled transitions. For technical reasons, we need to distinguish two types of computation: disc-computations where the first transition is a discrete transition and time-computations where the first transition is a timed transition.

A M_0-*disc-computation* π from a marking M_0 is a computation that starts with a discrete transition. It is a (finite or infinite) sequence

$$M_0 \longrightarrow^+_D M'_0 \longrightarrow_{x_0} M_1 \longrightarrow^+_D M'_1 \longrightarrow_{x_1} M_2 \longrightarrow^+_D M'_2 \longrightarrow_{x_2} M_3 \longrightarrow^+_D \ldots$$

of markings and transitions where $x_i \in \mathbb{R}^{>0}$. (If the sequence is infinite but contains only finitely many timed transitions then the infinite suffix has the form $\longrightarrow^\omega_D$.) It follows that

- The first transition is a discrete transition. Thus $M_0 \longrightarrow^+_D M'_0$.
- Every timed transition has a non-zero delay, i.e., $x_i \in \mathbb{R}^{>0}$.
- Without restriction, timed transitions cannot directly follow each other. We can require this, since $\longrightarrow_{x_1}\longrightarrow_{x_2}$ has the same effect as $\longrightarrow_{(x_1+x_2)}$. Therefore, timed transitions must be separated by at least one discrete transition. Thus we require $M_i \longrightarrow^+_D M'_i$ for $i \geq 0$.
- This implies that every infinite computation π must contain infinitely many discrete transitions $\longrightarrow_D$. An infinite computation may contain either finitely many or infinitely many timed transitions.

The *delay* of the disc-computation π is defined as

$$\Delta(\pi) := \sum_{i=0}^{\infty} x_i$$

A M_0-*time-computation* π from a marking M has the form

$$M \longrightarrow_x M_0 \xrightarrow{\pi'} \ldots$$

where $x \in \mathbb{R}^{>0}$ and π' is a M_0-disc-computation. In this case the delay $\Delta(\pi) := x + \Delta(\pi')$.

Intuitively, the delay is the total amount of time passed in all timed transitions in the sequence. For infinite computations π, the delay $\Delta(\pi)$ can be either infinite or finite. In the latter case the computation π is called a *zeno computation* (see Section 3). By $M \xrightarrow{\pi}$ we denote the fact that π is an M-computation.

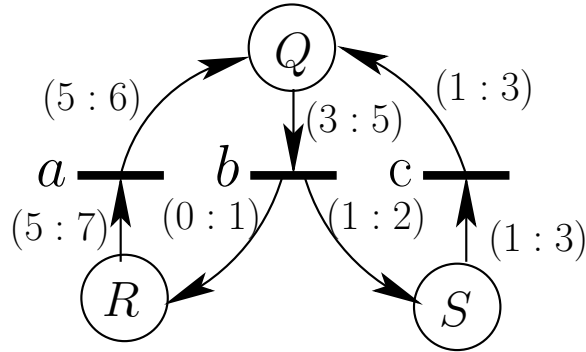

Figure 1: A small timed Petri net.

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 $M_2 = [(R, 0.2), (S, 1.6), (R, 5.8), (R, 5.0)]$.

Our model subsumes untimed Petri nets in the following sense. If all intervals are of the form $[0 : \infty)$ then the age of the tokens does not matter for the transitions, and thus the possible behavior (i.e., sequences of fired transitions) is the same as that of an untimed Petri net with the same structure. However, there cannot be any bijection between the sets of markings of a timed- and the corresponding untimed net, since the former is (in general) uncountable.

Next, we recall a constraint system called *regions* defined for *Timed automata* [AD90].

Regions: 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. 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 in the following manner.

Definition 2.3. A *region* is a triple (b_0, w, b_{max}) where

- $b_0 \in (P \times \{0, \ldots, max\})^{\odot}$. b_0 is a multiset of pairs. A pair of the form (p, n) represents a token with age exactly n in place p.
- $w \in \left((P \times \{0, \ldots, max - 1\})^{\odot} - \{\emptyset\} \right)^*$. This means that w is a word over the set $(P \times \{0, \ldots, max - 1\})^{\odot} - \{\emptyset\}$, i.e., w is a word where each element in the word is a non-empty multiset over $P \times \{0, \ldots, max - 1\}$. The pair (p, n) represents a token in place p with age x such that $x \in (n : n + 1)$. Pairs in the same multiset represent tokens whose ages have equal fractional parts. The order of the multisets in w corresponds to the order of the fractional parts (i.e., smaller fractional parts come first in the word w).
- $b_{max} \in P^{\odot}$. 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. (This is because the transitions in the net cannot distinguish between different ages of tokens if these are strictly bigger than max. Note that tokens with age exactly max are represented in b_0.)

The semantic of a region (b_0, w, b_{max}) would not change if we allowed empty multisets to appear in w. Therefore we forbid this in order to obtain a unique representation. However, the multisets b_0 and b_{max} can be empty.

Formally, each region R characterizes an infinite set of markings $[\![R]\!]$ 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 each multiset b_j be of the form $\left[\left(q_{(j,1)}, y_{(j,1)}\right), \ldots, \left(q_{(j,\ell_j)}, y_{(j,\ell_j)}\right) \right]$ for $j : 0 \leq j \leq m$ and b_{m+1} is of the form $[q_{(m+1,1)}, \ldots, q_{(m+1,l_{m+1})}]$. We say that M *satisfies* R, i.e., $M \in [\![R]\!]$, iff there is a bijection h from the set $\{1, \ldots, n\}$ to the set of pairs $\{(j, k) \mid (0 \leq j \leq m + 1) \wedge (1 \leq k \leq \ell_j)\}$ such that the following conditions are satisfied.

- $p_i = q_{h(i)}$. Each token should have the same place as that required by the corresponding element in R.
- If $h(i) = (j, k)$ then $j = m + 1$ iff $x_i > max$. Tokens older than max should correspond to elements in multiset b_{m+1}. The actual ages of these tokens are not relevant.
- If $x_i \leq max$ 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 $frac(x_i) = 0$ 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 $frac(x_{i_1}) < frac(x_{i_2})$ iff $j_1 < j_2$. This condition implies $frac(x_{i_1}) = frac(x_{i_2})$ iff $j_1 = j_2$. Thus, tokens with equal fractional parts correspond to elements in the same multiset (unless they belong to b_{m+1}). Furthermore, the ordering among the multisets inside R reflects the ordering among the fractional parts of the clock values (increasing from left to right).

We sometimes identify a region R with the set of markings $[\![R]\!]$ it represents (i.e., we write R instead of $[\![R]\!]$).
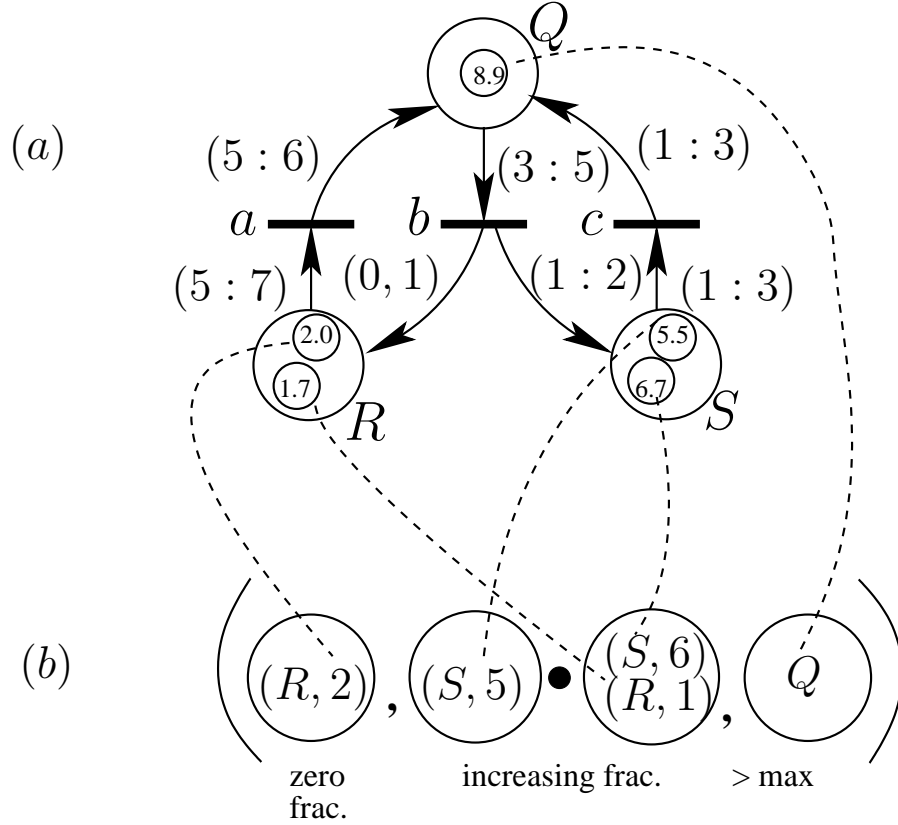
Figure 2: Marking M in (a) satisfies region R in (b).

Example 2.4. Consider the TPN N in Figure 1 with $max = 7$. Figure 2(a) shows a marking $M = [(R, 2.0), (S, 5.5), (R, 1.7), (S, 6.7), (Q, 8.9)]$. Figure 2(b) shows the unique region $R = ([(R, 2)], [(S, 5)] \bullet [(R, 1), (S, 6)], [Q])$ such that $M \in [\![R]\!]$. (The symbol $\bullet$ stands for concatenation.) In Figure 2(b), each circle corresponds to a multiset of tokens of N with same fractional parts. Dotted lines show how the tokens of M in TPN correspond to elements in the region R.

Equivalence and orders. 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 time-abstract bisimulation 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 $M_3 \longrightarrow M_4$.

Notice that given a marking M, it is easy to compute the unique region R_M satisfied by M.

Next we define an order and a preorder on markings of TPN. First, there is the usual order $\leq$ on multisets (markings are multisets of timed tokens). We have $M_1 \leq M_2$ iff $\forall p. M_1(p) \leq M_2(p)$, i.e., M_1 can be obtained from M_2 by removing some tokens.

The preorder $\preceq$ abstracts from the precise values of the ages of the tokens and considers only their relation to each other. We define $M_1 \preceq M_2$ if there is an M_2' with $M_1 \equiv M_2'$ and $M_2' \leq M_2$. 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 $\equiv$ equivalent (but not necessarily $=$ equivalent) to M_1. The relation $\preceq$ is only a preorder on the set of markings, because it is not antisymmetric. However, it is an order on the equivalence classes w.r.t. $\equiv$.

We let $M_1 \prec M_2$ denote that $M_1 \preceq M_2$ and $M_1 \not\equiv M_2$. Notice that $\longrightarrow$ is *monotonic* with respect to the preorder $\preceq$, i.e, 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 $M_3 \longrightarrow M_4$.

Next we define a partial order $\preceq$ on the set of regions.

Definition 2.5. Let $R = (b_0, b_1 \ldots b_m, b_{m+1})$ and $R' = (c_0, c_1 \ldots c_l, c_{l+1})$ be regions. Then, $R \preceq R'$ iff there is a strict monotone injection $g : \{0, \ldots, m+1\} \rightarrow \{0, \ldots, l+1\}$ with $g(0) = 0$ and $g(m+1) = l+1$ and $b_i \leq c_{g(i)}$ for each $i : 0 \leq i \leq m+1$. We let $R \prec R'$ denote that $R \preceq R'$ and $R \neq R'$.

The order $\preceq$ on regions agrees with the order $\preceq$ on markings.

Lemma 2.6. *For regions R and R', if $R \preceq R'$ then for each $M \in [\![R]\!], M' \in [\![R']\!]$, we have $M \preceq M'$.*

Proof. Directly from Def. 2.3 and Def. 2.5. □

Lemma 2.7. *Given a TPN and a region R, the upward closure $[\![R]\!]^\uparrow$ w.r.t. $\leq$ is the same as the upward-closure w.r.t. $\preceq$. Formally, $[\![R]\!]^\uparrow := \{M \mid \exists M' \in [\![R]\!].M' \leq M\} = \{M \mid \exists M' \in [\![R]\!].M' \preceq M\}$*

Proof. The $\subseteq$ inclusion is trivial, since $M' \leq M$ implies $M' \preceq M$. To prove the $\supseteq$ inclusion let $M' \in [\![R]\!]$ and $M' \preceq M$. Then, by definition of $\preceq$ there exists some marking M'' s.t. $M'' \leq M$ and $M'' \equiv M'$. It follows from $M' \in [\![R]\!]$ and the definition of $\equiv$ that $M'' \in [\![R]\!]$. Thus M is also in the first set. □

The following Lemma shows that the $\preceq$ preorder on regions of Def. 2.5 is compatible with the $\preceq$ preorder on markings. Thus (sets of) regions can be used as a canonical representation of upward-closed sets of markings, provided that they are closed under $\equiv$. We define the upward closure of a region w.r.t. $\preceq$ by $R{\uparrow} := \{R' \mid R \preceq R'\}$ and generalize the definition of the denotation from regions to sets of regions in the standard manner. So we define $[\![R{\uparrow}]\!] := \bigcup_{R \preceq R'} [\![R']\!]$.

Lemma 2.8. *Consider a region R of a TPN and the preorder $\preceq$ on markings and regions as defined in Def. 2.5. Then $[\![R]\!]^\uparrow = [\![R{\uparrow}]\!]$.*

Proof. If R is the empty region then the equivalence holds trivially. For the rest assume that R is not empty. If $M \in [\![R]\!]^\uparrow$ then there exists a marking $M' \leq M$ s.t. $M' \in [\![R]\!]$, by Lemma 2.7. It follows that $R = R_{M'} \preceq R_M =: R'$ and thus $M \in [\![R']\!] \subseteq [\![R{\uparrow}]\!]$.

If $M \in [\![R \uparrow]\!]$ then there exists some region R' with $R \preceq R'$ and $M \in [\![R']\!]$. Pick some marking $M' \in [\![R]\!]$. By Lemma 2.6 we get $M' \preceq M$. Thus we obtain $M \in [\![R]\!]^{\uparrow}$ by Lemma 2.7. $\qquad\square$

One can symbolically represent certain upward-closed sets of markings as the upward closures of finite sets of regions.

Definition 2.9. A *Multi-region upward closure* (MRUC) α is represented as a finite set of regions $\alpha := \{R_1, \ldots, R_n\}$ where each R_i is a region. This represents an upward closed set of markings $[\![\alpha]\!]$ defined as follows.

$$[\![\alpha]\!] := \bigcup_{i=1,\ldots,n} [\![R_i]\!]^{\uparrow}$$

Note that, by Lemma 2.8, $[\![\alpha]\!] = \bigcup_{i=1,\ldots,n} [\![R_i \uparrow]\!]$.

Lemma 2.10. *Multi-region upward closures (MRUCs) are effectively closed under union and intersection.*

Proof. The union operation is trivial, since for MRUC α, β we have $[\![\alpha]\!] \cup [\![\beta]\!] = [\![\alpha \cup \beta]\!]$.

For the intersection operation consider two MRUCs $\alpha := \{A_1, \ldots, A_n\}$ and $\beta := \{B_1, \ldots, B_m\}$. Then

$$[\![\alpha]\!] \cap [\![\beta]\!] = \bigcup_{1 \leq i \leq n,\, 1 \leq j \leq m} [\![A_i]\!]^{\uparrow} \cap [\![B_j]\!]^{\uparrow}$$

Thus it suffices to show that for any two regions A, B one can construct a MRUC $inter(A, B)$ s.t. $[\![inter(A, B)]\!] = [\![A]\!]^{\uparrow} \cap [\![B]\!]^{\uparrow}$. Given this, one can express the intersection as a new MRUC $\cup_{1 \leq i \leq n, 1 \leq j \leq m} inter(A_i, B_j)$, since

$$[\![\alpha]\!] \cap [\![\beta]\!] = \left[\!\!\left[\bigcup_{1 \leq i \leq n,\, 1 \leq j \leq m} inter(A_i, B_j) \right]\!\!\right]$$

We construct the MRUC $inter(A, B)$ for given regions A, B. Let $A = (a_0, a_1 a_2 \ldots a_n, a_{max})$ and $B = (b_0, b_1 b_2 \ldots b_m, b_{max})$.

Intuition: For the multisets a_0, b_0 and a_{max}, b_{max} constructing the minimal requirements for the intersection of their upward-closures is simple. It is just the maximum, i.e., the multiset union (see Def. 2.1 for multisets), and we have $a_0^{\uparrow} \cap b_0^{\uparrow} = (a_0 \cup b_0)^{\uparrow}$ (similarly for a_{max}, b_{max}).

The sequences of multisets $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_m$ represent orderings of the fractional parts of the ages of tokens in those multisets. However, the fractional part of a_1 could be smaller, equal to, or larger than the fractional part of b_1, b_2, etc. All of these cases must be considered. If two multisets a_i, b_j represent the same fractional part, then the minimal requirement for markings in the upward-closure of the intersection is the maximum, i.e., the multiset union of a_i and b_j. Otherwise they must appear individually in the proper order of the fractional parts.

Construction: Formally, let F be the set of all injective, strictly monotone increasing functions $f : \{1, \ldots, n\} \to \{1, \ldots, n + m\}$ and G the set of all injective, strictly monotone increasing functions $g : \{1, \ldots, m\} \to \{1, \ldots, n+m\}$. (Note that F and G are finite.) These functions are normally not surjective and we define $R(f) := f(\{1, \ldots, n\})$ and $R(g) := g(\{1, \ldots, m\})$. For any $f \in F$ and $g \in G$ we define a sequence of multisets

$$s(f, g) := c_1 c_2 \ldots c_{n+m}$$

such that for any $i \in \{1, \ldots, n + m\}$

- If $i \in R(f) \cap R(g)$ then $\exists j, k. \, i = f(j) = g(k)$. Let $c_i := a_j \cup b_k$.
- If $i \in R(f)$ and $i \notin R(g)$ then $\exists j = f^{-1}(i)$. Let $c_i := a_j$.
- If $i \notin R(f)$ and $j \in R(g)$ then $\exists k = g^{-1}(i)$. Let $c_i := b_k$.
- Else $c_i := \emptyset$.

For each f, g, the sequence of multisets $s(f, g)$ describes a possible interleaving/combination of the sequences $a_1 \ldots a_n$ and $b_1 \ldots b_m$. However, $s(f, g)$ might contain some empty multisets, which must be removed in order to satisfy the requirements for regions (see Def. 2.3). Given a sequence of multisets $x_1 \ldots x_k$, let $e(x_1 \ldots x_k)$ be the subsequence where all the empty multisets have been removed.

We can now define the MRUC

$$inter(A, B) := \bigcup_{f \in F, \, g \in G} \{(a_0 \cup b_0, e(s(f, g)), a_{max} \cup b_{max})\}$$

Proof of correctness: We show that this construction satisfies the required property $[\![inter(A, B)]\!] = [\![A]\!]^\uparrow \cap [\![B]\!]^\uparrow$.

Let $M \in [\![inter(A, B)]\!]$. Then there exist $f \in F, g \in G$ s.t. $M \in [\![(a_0 \cup b_0, e(s(f, g)), a_{max} \cup b_{max})]\!]^\uparrow$. Since $a_1, \ldots, a_n$ is a subsequence of $e(s(f, g))$ and $a_0 \subseteq a_0 \cup b_0$ and $a_{max} \subseteq a_{max} \cup b_{max}$ we get $[\![A]\!]^\uparrow = [\![(a_0, a_1 a_2 \ldots a_n, a_{max})]\!]^\uparrow \supseteq [\![(a_0 \cup b_0, e(s(f, g)), a_{max} \cup b_{max})]\!]^\uparrow$. Therefore, $M \in [\![A]\!]^\uparrow$. By a symmetric argument (with a and b interchanged) we obtain $M \in [\![B]\!]^\uparrow$. So finally we get $M \in [\![A]\!]^\uparrow \cap [\![B]\!]^\uparrow$.

Now we show the other inclusion. Let $M \in [\![A]\!]^\uparrow \cap [\![B]\!]^\uparrow$. There exist markings $M_1 \leq M$ and $M_2 \leq M$ with $M_1 \in [\![A]\!]$ and $M_2 \in [\![B]\!]$. Since M_1, M_2 are markings, they are multisets of (timed) tokens and we can define a new marking M' as their multiset union (see Def. 2.1) by $M' := M_1 \cup M_2$ and obtain $M' \leq M$. Now there exist functions $f \in F$ and $g \in G$, expressing the relative orders of the fractional parts in M_1 and M_2, s.t. $M' \in [\![(a_0 \cup b_0, e(s(f, g)), a_{max} \cup b_{max})]\!]$. It follows that $M \in [\![(a_0 \cup b_0, e(s(f, g)), a_{max} \cup b_{max})]\!]^\uparrow$ and thus $M \in [\![inter(A, B)]\!]$. $\qquad\square$

We define functions Pre and $Post$ on sets of markings S such that $Pre(S)$ and $Post(S)$ are the one-step predecessors and successors of markings in S, respectively. Formally, $Pre(S) := \{M \mid \exists M' \in S. \, M \longrightarrow M'\}$ and $Post(S) := \{M \mid \exists M' \in S. \, M' \longrightarrow M\}$. By replacing the transition relation with its reflexive-transitive closure we obtain the sets of all predecessors and successors, respectively. Formally, $Pre^*(S) := \{M \mid \exists M' \in S. \, M \stackrel{*}{\longrightarrow} M'\}$ and $Post^*(S) := \{M \mid \exists M' \in S. \, M' \stackrel{*}{\longrightarrow} M\}$.

The following lemmas show that for TPN and multi-region upward closures (MRUC) S, one can effectively construct the sets $Post(S)$, $Pre(S)$ and $Pre^*(S)$ as MRUC.

Lemma 2.11. *([ADMN04]) Let S be a set of markings which is represented as the upward-closure of a finite set of regions, i.e., a MRUC. Then the set $Post(S)$ is effectively constructible as a MRUC.*

The construction for $Pre^*(S)$ is done by the classic technique of successive construction of $Pre^{\leq n}(S)$ for larger and larger n (all of which are upward closed and representable by MRUC) which eventually converges to $Pre^*(S)$ by Higman's Lemma [Hig52], because $\preceq$ is a well-founded preordering on regions. (The correctness is implied by the compatibility of the preorder $\preceq$ on regions with the order $\leq$ on markings, i.e., Lemma 2.7 and Lemma 2.8.) A

proof can be found in [AJ98] and a more general result (for the more expressive formalism of 'existential zones') has been shown in [AN01].

Lemma 2.12. *Let S be a set of markings which is represented as the upward-closure of a finite set of regions, i.e., a MRUC. Then the sets $Pre(S)$ and $Pre^*(S)$ are effectively constructible as MRUC.*

Finally, it is known that, for TPN, the set $Post^*(S)$ cannot be effectively constructed in any symbolic representation with a decidable membership problem, since the reachability problem is undecidable [RGdFE99].

3. Zenoness

A zeno-computation of a timed Petri net is an infinite computation that has a finite delay.

Zenoness-Problem
Instance: A timed Petri net N, and a marking M of N.
Question: Is there an infinite M-computation π and a finite number m s.t. $\Delta(\pi) \leq m$?

We consider a timed Petri net N. A marking M is called a *zeno-marking* of N iff the answer to the above problem is 'yes'.
Note that the zeno-computation π can be either a disc-computation or a time-computation, depending on whether the first transition is discrete or timed.
We let $ZENO$ denote the set of all zeno-markings of N. More generally, we define

$$ZENO^m := \{M \mid \exists \text{ an infinite computation } \pi.\, M \xrightarrow{\pi} \ \wedge \Delta(\pi) \leq m\}$$

Thus $ZENO = \bigcup_{m \geq 0} ZENO^m$.

The decidability of the zenoness-problem for timed Petri nets (i.e., the problem if $M \in ZENO$ for a given marking M, or, more generally, constructing $ZENO$) was mentioned in [dFERA00] by Escrig, et.al. as an open problem for both discrete and dense-timed Petri nets. In this section, we show that for any TPN, a characterization of the set $ZENO$ can be effectively computed. We also show that this implies the computability of $ZENO$ for discrete-timed Petri nets.

The following outline explains the main steps of our proof.

Step 1: We translate the original timed Petri net N into an untimed *simultaneous-disjoint-transfer net* N'. Simultaneous-disjoint-transfer nets are a subclass of *transfer Petri nets* [Hei82, FS01] where all transfers happen at the same time and do not affect each other (i.e., all sources and targets of all transfers are disjoint). The computations of N' represent, in a symbolic way, the computations of N that can be performed in time less than $1 - \delta$ for some predefined $0 < \delta < 1$.

Step 2: We consider the set INF of markings of N', from which an infinite computation is possible. INF is upward-closed and can therefore be characterized by the finite set INF_{min} of its minimal elements. While INF_{min} is not computable for general transfer nets [DJS99, May03], it is computable for simultaneous-disjoint-transfer nets, as shown in Lemma 3.41.

Step 3: We re-interpret the set INF (resp. INF_{min}) of N' markings in the context of the timed Petri net N and construct from it a characterization of the set $ZENO$, described by a multi-region upward closure (MRUC) (see Def. 2.9).

To simplify the presentation, we first show Step 1 and Step 3. Then, we show how to perform Step 2.

3.1. **Step 1: Translating TPNs to Simultaneous-Disjoint-Transfer Nets.**
First we define *simultaneous-disjoint-transfer nets*.

Definition 3.1. Simultaneous-disjoint-transfer nets (short SD-TN) are a subclass of transfer nets. A SD-TN N is described by a tuple $(P, T, Input, Output, Trans)$ where
- P is a set of places,
- T is a set of ordinary transitions,
- $Input, Output : T \to 2^P$ are functions that describe the input and output places of every transition, respectively (as in ordinary Petri nets), and
- $Trans$ describes the simultaneous and disjoint transfer transition. In order to emphasize the simultaneous operation of the transfers, we define $Trans$ as a single transition with many effects, rather than as a set of transitions. We have $Trans = (I, O, ST)$ where $I \subseteq P$, $O \subseteq P$, and $ST \subseteq P \times P$. $Trans$ consists of two parts: (a) I and O describe the input and output places of the Petri net transition part; (b) the pairs in ST describe the source and target places of the transfer part. Furthermore, the following restrictions on $Trans$ must be satisfied:
  - If $(sr, tg), (sr', tg') \in ST$ then sr, sr', tg, tg' are all different and $\{sr, tg\} \cap (I \cup O) = \emptyset$.

Let $M : P \to \mathbb{N}$ be a marking of N. We use $\leq$ as the ordering on the set of markings (Section 2). The firing of normal transitions $t \in T$ is defined just as for ordinary Petri nets. A transition $t \in T$ is enabled at marking M iff $\forall p \in Input(t).\, M(p) \geq 1$. Firing t yields the new marking M' where

$$
\begin{aligned}
M'(p) &= M(p)   && \text{if } p \in Input(t) \cap Output(t) \\
M'(p) &= M(p) - 1 && \text{if } p \in Input(t) - Output(t) \\
M'(p) &= M(p) + 1 && \text{if } p \in Output(t) - Input(t) \\
M'(p) &= M(p)   && \text{otherwise}
\end{aligned}
$$

The transfer transition $Trans$ is enabled at M iff $\forall p \in I.\, M(p) \geq 1$. Firing $Trans$ yields the new marking M' where

$$
\begin{aligned}
M'(p) &= M(p)   && \text{if } p \in I \cap O \\
M'(p) &= M(p) - 1   && \text{if } p \in I - O \\
M'(p) &= M(p) + 1   && \text{if } p \in O - I \\
M'(p) &= 0   && \text{if } \exists p'.\, (p, p') \in ST \\
M'(p) &= M(p) + M(p') && \text{if } (p', p) \in ST \\
M'(p) &= M(p)   && \text{otherwise}
\end{aligned}
$$

The restrictions above ensure that these cases are disjoint. Note that after firing $Trans$ all source places of transfers are empty, since, by the restrictions defined above, no place is both source and target of a transfer.

We use $M \longrightarrow M'$ to denote that M' is reached from M either by executing an ordinary Petri net transition $t \in T'$ or the transfer transition $Trans$.

In the following, sometimes we use *transfer* transition to mean simultaneous-disjoint transfer transitions.

3.1.1. *Construction of SD-TN from a TPN.* For a given TPN $N = (P, T, In, Out)$ we construct a SD-TN $N' = (P', T', Input, Output, Trans)$. The intuition is that N' simulates symbolically all computations of N which can happen in time $< 1 - \delta$ for some predefined $1 > \delta > 0$. First we show how to construct the places of SD-TN. Then we show how to simulate a discrete transition of N by a set of transitions of N'. Finally, we show how to simulate timed transitions of N by simultaneous-disjoint-transfers and a set of normal discrete transitions as in ordinary PNs.

We let max be the maximal finite constant that appears in the arcs of the TPN. We define a finite set of symbols $Sym := \{k \mid k \in \mathbb{N}, 0 \le k \le max\} \cup \{k+ \mid k \in \mathbb{N}, 0 \le k \le max\} \cup \{k- \mid k \in \mathbb{N}, 1 \le k \le max\}$ and a total order on Sym by $k < k+ < (k+1)- < (k+1)$ for every k.

3.1.2. *Constructing places of SD-TN.* We let $P' = \{p(sym) \mid p \in P, sym \in Sym\}$, i.e., for every place $p \in P$ of N we have a set containing places of the form $p(sym)$ such that $sym \in Sym$. The set P' is finite, since both P and sym are finite.

A token in place $p(k)$ encodes a token of age exactly k on place p. A token in $p(k+)$ encodes a token in place p of an age x which satisfies $k < x \le k + \delta$ for some a-priori defined $0 < \delta < 1$. This means that the age of this token cannot reach $k + 1$ in any computation taking time $< 1 - \delta$. A token in $p(k-)$ encodes a token in p whose age x satisfies $k - 1 + \delta < x < k$ and which may or may not reach age k during a computation taking time $1 - \delta$. For instance, given $\delta = 0.6$, a TPN token $(p, 1.5)$ is encoded as $p(1+)$ while another TPN token $(p, 2.7)$ is encoded as $p(3-)$. The SD-TN tokens $p(k), p(k+)$ and $p(k-)$ are called *symbolic encodings* of the corresponding TPN token (p, a).

In particular, the age of a $p(k-)$ token could be chosen arbitrarily close to k, such that its age could reach (or even exceed) k in computations taking an arbitrarily small time.

3.1.3. *Translating Discrete Transitions.* First we define a function $enc : Intrv \to 2^{Sym}$ as follows.

$$
\begin{aligned}
enc([x : y]) &:= \{sym \in Sym \mid x \le sym \le y\} \\
enc((x : y]) &:= \{sym \in Sym \mid x < sym \le y\} \\
enc([x : y)) &:= \{sym \in Sym \mid x \le sym < y\} \\
enc((x : y)) &:= \{sym \in Sym \mid x < sym < y\}
\end{aligned}
$$

For instance, $enc([1 : 2]) = \{1, 1+, 2-, 2\}$ and $enc([1 : 2)) = \{1, 1+, 2-\}$. We say that $enc(\mathcal{I})$ is the *encoding* of interval $\mathcal{I}$. By the definition above, the bound ∞ is encoded as $max+$, i.e., $enc([1 : \infty)) = \{1, 1+, 2-, 2, \ldots, max, max+\}$.

For every transition $t \in T$ in the TPN N, we have a set $T'(t)$ of new transitions in N'. The intuition is that the transitions in $T'(t)$ encode all possibilities of the age intervals of input and output tokens.

Example 3.2. Consider the TPN in Figure 3, part 1. The only (discrete) transition t has an input arc from place p labeled $[0 : 1]$ and two output arcs both labeled $[0 : 0]$ to places p and q, respectively. The translation of this transition into its corresponding SD-TN would yield 4 different transitions in $T'(t)$ with output arcs to both places $p(0)$ and $q(0)$, and input arcs from places $p(0), p(0+), p(1-)$ or $p(1)$, respectively, as shown in Figure 3, parts 2.(a), 2.(b), 2.(c), and 2.(d).
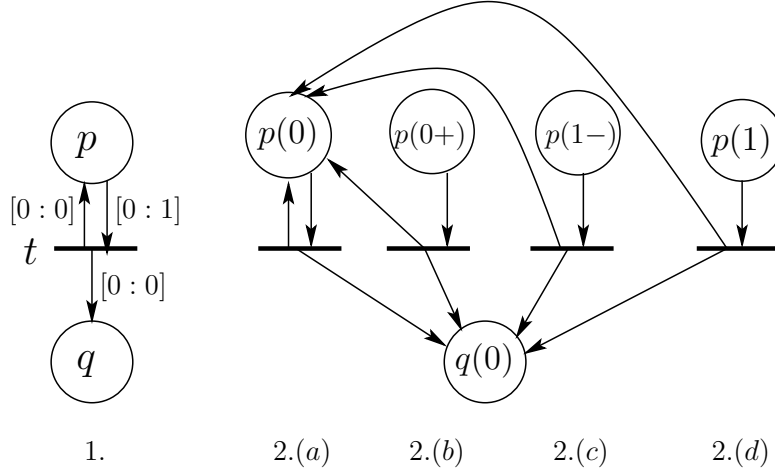
Figure 3: Simulating (1) t in TPN by (2) a set $T'(t)$ consisting of 4 transitions in 2.(a), 2.(b), 2.(c) and 2.(d).

Example 3.3. Consider the TPN in Figure 4, part 1. The only (discrete) transition t has an input arc from place p as in Figure 3, part 1., but the output arc to place q is labeled by the interval $[0:1]$. This will yield the 16 different transitions in $T'(t)$, shown in Figure 4, part 2., since $enc([0:1]) = \{0, 0+, 1-, 1\}$.

Each transition t of TPN N yields a set $T'(t)$ of transitions in the corresponding SD-TN N'. Each transition in the set $T'(t)$ is of the form $t'(A, B)$ where A and B are the set of input and output places of $t'(A, B)$ respectively, i.e., $Input(t'(A, B)) = A$ and $Output(t'(A, B)) = B$. In the following, for each transition t in TPN, we compute a set $\mathcal{P}_{in}(t)$ ($\mathcal{P}_{out}(t)$) which contains the set of input (output) places for each transition in $T'(t)$.

For every $t \in T$, consider the set of input arcs $A_{in}(t) = \{p_1(\mathcal{I}_1), \ldots, p_m(\mathcal{I}_m)\}$ and the set of output arcs $A_{out}(t) = \{p'_1(\mathcal{J}_1), \ldots, p'_\ell(\mathcal{J}_\ell)\}$. Now, we define $\mathcal{P}_{in}(t) \subseteq 2^{P'}$ where each element in $\mathcal{P}_{in}(t)$ is a set A of places and is given by

$$A = \{p_1(sym_1), \ldots, p_m(sym_m)\}$$

where $sym_i \in enc(\mathcal{I}_i)$ for $i : 1 \leq i \leq m$. Intuitively, each set A in $\mathcal{P}_{in}(t)$ corresponds to a unique combination of encodings of input tokens of t in N.

For every $t \in T$ we define $\mathcal{P}_{out}(t) \subseteq 2^{P'}$ in a similar manner. We define $\mathcal{P}_{out}(t)$ where each element in $\mathcal{P}_{out}(t)$ is a set B of places and is given by

$$B = \{p'_1(sym'_1), \ldots, p'_\ell(sym'_\ell)\}$$

where $sym'_i \in enc(\mathcal{J}_i)$ for $i : 1 \leq i \leq \ell$. Similarly, each set B in $\mathcal{P}_{out}(t)$ corresponds to a unique combination of encodings of output tokens of t in N.

We define $T'(t) := \{t'(A, B) \mid A \in \mathcal{P}_{in}(t), B \in \mathcal{P}_{out}(t)\}$ and finally $T' := \bigcup_{t \in T} T'(t)$.

Example 3.4. Consider the example in Figure 3. Here, $In(t, p) = [0 : 1]$, $Out(t, p) = [0 : 0]$, $In(t, q) = \emptyset$ and $Out(t, q) = [0 : 0]$. We have $enc([0 : 1]) = \{0, 0+, 1-, 1\}$ and $enc([0 : 0]) = \{0\}$. Then $\mathcal{P}_{in}(t) = \{\{p(0)\}, \{p(0+)\}, \{p(1-)\}, \{p(1)\}\}$ and $\mathcal{P}_{out}(t) = \{\{q(0)\}\}$. The four transitions in Figure 3.2 are given by $t'(\{p(0)\}, \{q(0)\})$, $t'(\{p(0+)\}, \{q(0)\})$,
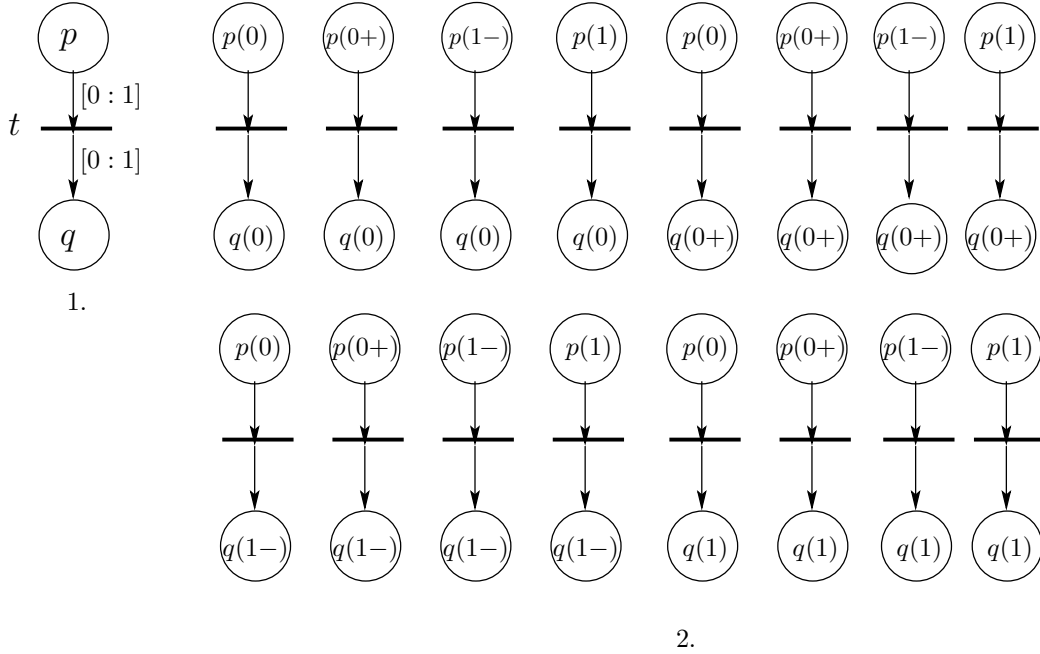
Figure 4: Simulating (1) t in TPN by (2) a set $T'(t)$ consisting of 16 transitions. (For readability, these 16 transitions are listed individually, rather than in a combined net.)

$t'(\{p(1-)\}, \{q(0)\})$ and $t'(\{p(1)\}, \{q(0)\})$, respectively. $T'(t)$ consists of the above four transitions.

3.1.4. *Translating Timed Transitions.* So far, the transitions in T' only encode the discrete transitions of N. The passing of time will be encoded by a sequence of transitions, including one use of the transfer transition. Our construction must ensure the following properties.

- We need to keep discrete transitions and time-passing separate. Therefore, we must first modify the net to obtain alternating discrete phases and time-passing phases.
- Time-passing phases must not directly follow each other. They must be separated by at least one discrete transition.

Our SD-TN is extended and modified in several steps.

(1) First we add three extra places p_{disc}, p_{time1} and p_{time2} to P' which act as control-states for the different phases. (The time-passing phase has two sub-phases). The construction will ensure that at any time there is exactly one token on exactly one of these places.
(2) Normal transitions can fire if and only if p_{disc} is marked. Thus we modify all transitions $t \in T'$ by adding p_{disc} to $Input(t)$ and $Output(t)$.
(3) We add an extra place p_{count} to P' which counts the number of fired discrete transitions since the last time-passing phase. Thus we modify all transitions $t \in T'$ by adding p_{count} to $Output(t)$. This is needed to ensure that time-passing phases are separated by at least one discrete transition. A new time-passing phase can only start if p_{count} is non-empty, and p_{count} will be cleared of tokens during the time-passing phase.

(4) Now we add a new transition $t_{switch-time}$ which starts the time-passing phase. We define $Input(t_{switch-time}) = \{p_{disc}, p_{count}\}$ and $Output(t_{switch-time}) = \{p_{time1}\}$. It can only fire if p_{count} is marked (thus time-passing phases cannot directly follow each other) and moves the control-token from p_{disc} to p_{time1}. (Note that p_{count} is not necessarily empty after this operation, since it might have contained more than one token. The place p_{count} will be cleared later by the transfer transition.)

(5) If the control-token is on p_{time1} then the transfer transition *Trans* is the only enabled transition. It encodes (in an abstract way) the effect of the passing time on the ages of tokens. After an arbitrarily small amount of time < 1 passes, all tokens of age k have an age $> k$. This is encoded by the simultaneous-disjoint transfer arc, which moves all tokens from places $p(k)$ to places $p(k+)$. Furthermore, it will move the control-token from place p_{time1} to place p_{time2}. Finally, it needs to clear the place p_{count} of tokens. To do this, we add a new special place p_{dump} (which is not an input place of any transition; the number of tokens on p_{dump} is semantically irrelevant) and transfer all tokens from p_{count} to p_{dump}. Formally, *Trans* $:= (I, O, ST)$ where $I := \{p_{time1}\}$, $O := \{p_{time2}\}$, and $ST := \{(p(k), p(k+)) \mid 0 \le k \le max\} \cup \{(p_{count}, p_{dump})\}$. Note that the transfer transition *Trans* is enabled even if no tokens are present on the places $p(k)$.

(6) Now the control-token is on place p_{time2}. Next we add two new sets of transitions to T', which encode what happens to tokens of age $k-$ when (a small amount < 1 of) time passes. Their age might either stay below k, reach k or exceed k. Notice that we do not need to do anything in the first case.
   - For every $k \in \{1, \ldots, max\}$ we have a transition with input places p_{time2} and $p(k-)$ and output places p_{time2} and $p(k)$. This encodes the second scenario.
   - Furthermore, for every $k \in \{1, \ldots, max\}$ we have a transition with input places p_{time2} and $p(k-)$ and output places p_{time2} and $p(k+)$. This encodes the third scenario.

(7) Finally, we add an extra transition $t_{switch-disc}$ with input place p_{time2} and output place p_{disc}, which switches the net back to normal discrete mode.

Note that after a time-passing phase the only tokens on places $p(k)$ are those which came from $p(k-)$, because all tokens on $p(k)$ were first transferred to $p(k+)$ by the transfer transition. Furthermore, the place p_{count} is empty after a time-passing phase, and thus $t_{switch-time}$ is not immediately enabled. At least one discrete transition must fire before the next time-passing phase. Therefore, every infinite computation of the SD-TN N' must contain infinitely many discrete transitions.

Convention: Since the number of tokens on place p_{dump} is semantically irrelevant, we will ignore this place in the rest of our proof. It was only introduced for technical reasons to empty p_{count} by the transfer, since we do not have reset-arcs, but only a transfer arc.

Example 3.5. In Figure 5, we simulate the timed transitions of a TPN with a single place p and $max = 1$. The transition $t_{switch-time}$ starts the time-passing phase by moving the token from p_{disc} to p_{time1} and consumes one token from p_{count} (thus it cannot fire if p_{count} is empty). The transfer transition is described by the dotted line and the transfer arcs are shown as thick arrows from the source of the transfer to the target of the transfer, namely from $p(0)$ to $p(0+)$ and from $p(1)$ to $p(1+)$. The place p_{count} is cleared by moving all its tokens to the (otherwise unused) place p_{dump}. The Petri net part of a transfer (input from p_{time1} and output to p_{time2}) is shown as ordinary arcs. The transitions t_1 and t_2 move a token from $p(1-)$ to $p(1)$ and to $p(1+)$, respectively, if there is a token in p_{time2}. Finally, $t_{switch-disc}$ moves the token from p_{time2} back to p_{disc} and ends the time-passing phase.
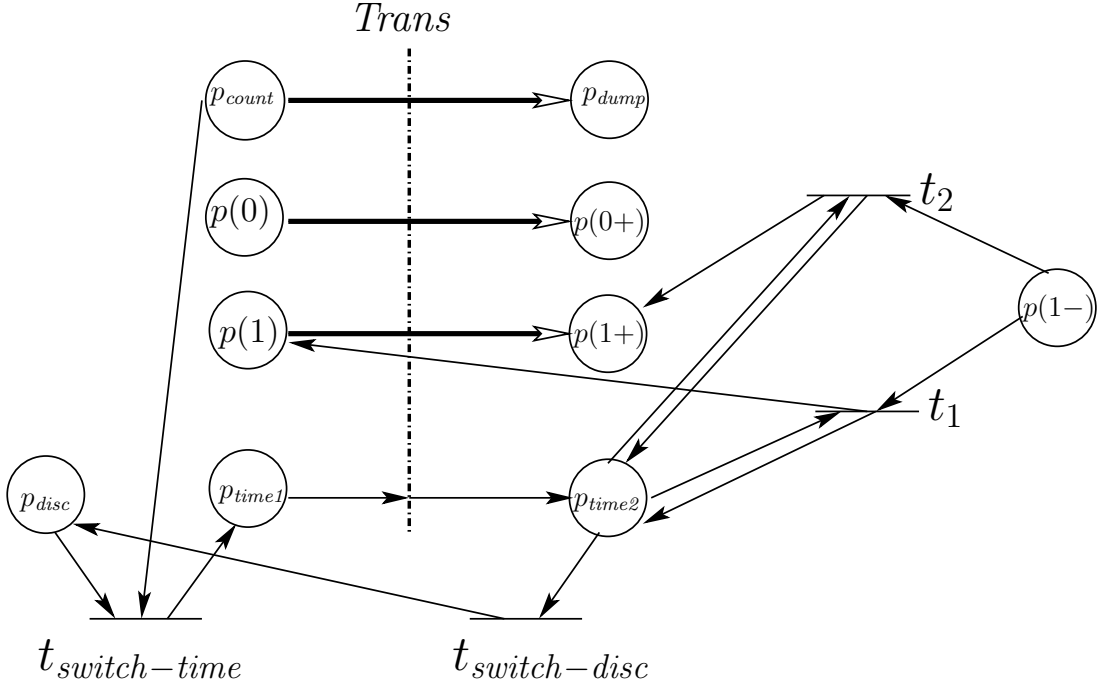
Figure 5: Simulating a time-passing transition in a TPN for time $< 1 - \delta$, by the corresponding SD-TN.

3.2. **Step 3: Constructing ZENO.** In this section, we show how to compute the set ZENO as a MRUC.

Definition 3.6. Let N be a TPN and $N' = (P', T', Input, Output, Trans)$ the corresponding SD-TN, defined as in Subsection 3.1.

- We say that a marking M' of N' is a *standard marking* if $M'(p_{disc}) = 1$ and $M'(p_{time1}) = M'(p_{time2}) = 0$ and $M'(p_{count}) = 0$. (It follows that a computation from a standard marking cannot start directly with a time-passing phase.) Let Ω be the set of all markings of N' and Ω' the set of all standard markings of N'.
- We denote by INF the set of all markings of N' from which infinite computations start. Since INF is upward-closed in Ω with respect to $\leq$ and $\leq$ is a well-quasi-ordering, INF can be characterized by its finitely many minimal elements (see also Lemma 3.18). Let INF_{min} be the set of minimal elements (markings).
- Let INF' and INF'_{min} be the restriction to standard markings of INF and INF_{min}, respectively. I.e., $INF' := INF \cap \Omega'$ and $INF'_{min} := INF_{min} \cap \Omega'$. The set INF' is not upward-closed in Ω. However, by the following Lemma 3.7, INF' is the upward-closure of INF'_{min} in Ω'. Thus INF' can be characterized by the finite set INF'_{min} of its minimal elements.

Lemma 3.7. INF' *is the upward-closure of* INF'_{min} *in* Ω'.

Proof. Let $X := \{M' \in \Omega' \mid \exists M \in INF'_{min}. M' \geq M\}$ be the upward-closure of INF'_{min} in Ω'. We need to show that $INF' = X$.

The inclusion $X \subseteq INF'$ holds trivially, by monotonicity of SD-TN and the fact that all markings in X are standard markings.

Now we show the other inclusion $INF' \subseteq X$. Let $M' \in INF' = INF \cap \Omega'$. Since $M' \in INF$, there exists some marking $M \in INF_{min}$ such that $M \leq M'$. Since $M \in INF$, it follows from the definition of INF and the construction of the SD-TN N' that $M(p_{disc}) + M(p_{time1}) + M(p_{time2}) \geq 1$, i.e., at least one of these places must be marked or there cannot be an infinite run. Since $M' \in \Omega'$ we have $M'(p_{disc}) = 1$ and $M'(p_{time1}) = M'(p_{time2}) = M'(p_{count}) = 0$. Therefore, by $M \leq M'$, we have that $M(p_{disc}) = 1$ and $M(p_{time1}) = M(p_{time2}) = M(p_{count}) = 0$ and thus $M \in \Omega'$. So we obtain $M \in INF_{min} \cap \Omega' = INF'_{min}$. Since $M' \in \Omega'$ is a standard marking and $M' \geq M$, we finally obtain $M' \in X$ as required. $\square$

The following definitions establish the connection between the markings of the timed Petri net N and the markings of the SD-TN N'.

Definition 3.8. For every δ with $0 < \delta < 1$ we define a function $int_\delta : (P \times \mathbb{R}^{\geq 0})^{\odot} \to (P' \to \mathbb{N})$ that maps a marking M of N to its corresponding marking M' in N'. $M' := int_\delta(M)$ is defined as follows. Let

$$
\begin{aligned}
M'(p(k)) &:= M((p,k)) & &\text{for } k \in \mathbb{N}, 0 \leq k \leq max. \\
M'(p(k+)) &:= \textstyle\sum_{k < x \leq k+\delta} M((p,x)) & &\text{for } k \in \mathbb{N}, 0 \leq k \leq max - 1. \\
M'(p(max+)) &:= \textstyle\sum_{max < x} M((p,x)) \\
M'(p((k+1)-)) &:= \textstyle\sum_{k+\delta < x < k+1} M((p,x)) & &\text{for } k \in \mathbb{N}, 0 \leq k \leq max - 1. \\
M'(p_{disc}) &:= 1 \\
M'(p_{time1}) &:= 0 \\
M'(p_{time2}) &:= 0 \\
M'(p_{count}) &:= 0
\end{aligned}
$$

Note that $M' = int_\delta(M)$ is a standard marking according to Def. 3.6.

For instance, for a TPN marking $M = [(p,1),(p,0.5),(p,0.95),(p,1.9),(p,2.1),(p,3.9)]$ and $max = 2, \delta = 0.8$ we obtain $int_\delta(M) = [p(1),p(0+),p(1-),p(2-),p(max+),p(max+),p_{disc}]$.

The intuition is as follows. In an infinite computation π starting at M with $\Delta(\pi) < 1 - \delta$, no TPN token (p,x) with $k < x \leq k+\delta$ can reach age $k+1$ by aging. This is reflected in N' by the fact that $p(k+)$ tokens are not affected during the time-passing phase. On the other hand, TPN tokens (p,x) with $k + \delta < x < k + 1$ can reach an age $\geq k + 1$ by aging. This is reflected in N' by the fact that $p((k+1)-)$ tokens can become $p(k+1)$ or $p((k+1)+)$ tokens during the time-passing phase.

The following lemma establishes a correspondence between fast disc-computations of the TPN (i.e., starting with a discrete transition; see Section 2) and computations of the SD-TN.

Lemma 3.9. *Consider a TPN N with marking M_0, the corresponding SD-TN N' constructed as above, and $0 < \delta < 1$. If there exists an infinite M_0-disc-computation π such that $\Delta(\pi) < 1 - \delta$ then there exists an infinite $int_\delta(M_0)$-computation π' in N', i.e., $int_\delta(M_0) \in INF'$.*

Proof. We show that for every infinite M_0-disc-computation π of the form

$$M_0 \longrightarrow_D M_0^1 \longrightarrow_D M_0^2 \longrightarrow_D \ldots M_0^{n_0}$$
$$\longrightarrow_{x_0}$$
$$M_1 \longrightarrow_D M_1^1 \longrightarrow_D M_1^2 \longrightarrow_D \ldots M_1^{n_1}$$
$$\longrightarrow_{x_1}$$
$$M_2 \ldots$$

with $n_i \geq 1$ and $\Delta(\pi) < 1 - \delta$, there is a corresponding infinite computation in N' of the form

$$int_{\delta_0}(M_0) \longrightarrow int_{\delta_0}(M_0^1) + \{p_{count}\} \longrightarrow int_{\delta_0}(M_0^2) + \{p_{count}^2\} \longrightarrow \ldots int_{\delta_0}(M_0^{n_0}) + \{p_{count}^{n_0}\}$$
$$\xrightarrow{*}$$
$$int_{\delta_1}(M_1) \longrightarrow int_{\delta_1}(M_1^1) + \{p_{count}\} \longrightarrow int_{\delta_1}(M_1^2) + \{p_{count}^2\} \longrightarrow \ldots int_{\delta_1}(M_1^{n_1}) + \{p_{count}^{n_1}\}$$
$$\xrightarrow{*}$$
$$int_{\delta_2}(M_2) \ldots$$

with $\delta_0 = \delta$ and for all i, $1 > \delta_{i+1} > \delta_i$. Let π_i be the infinite suffix of π starting at M_i. The values of δ_i will be defined such that $\Delta(\pi_i) < 1 - \delta_i$. (The condition $\delta_{i+1} > \delta_i$ is required, because $\Delta(\pi_{i+1}) < \Delta(\pi_i)$.)

For every discrete transition step $M_i^j \longrightarrow_D M_i^{j+1}$ there exists a transition step in N' of the form $int_{\delta_i}(M_i^j) + \{p_{count}^j\} \rightarrow_{dt} int_{\delta_i}(M_i^{j+1}) + \{p_{count}^{j+1}\}$, where $dt \in T'(t)$ by the construction in Section 3.1.1 and Def. 3.8. Note that the functions int_{δ_i} always return standard markings (with no tokens on place p_{count}). However, in the computation of the SD-TN, the number of tokens on p_{count} represents the number of steps since the last time-passing phase.

For every timed transition step $M_i^{n_i} \longrightarrow_{x_i} M_{i+1}$ we have $\delta_{i+1} = \delta_i + x_i \leq 1$. By the construction in Section 3.1.1 and Def. 3.8 there is a sequence of transitions in N' (the encoding of the time-passing phase) of the form $int_{\delta_i}(M_i^{n_i}) + \{p_{count}^{n_i}\} \xrightarrow{*} int_{\delta_{i+1}}(M_{i+1})$. The time-passing phase can start at $int_{\delta_i}(M_i^{n_i}) + \{p_{count}^{n_i}\}$, because $n_i \geq 1$, i.e., there is at least one token on place p_{count}. Note in particular that if some token (p, x) with $k + \delta_i < x < k + 1$ reaches an age equal to (or greater than) $k + 1$ in the transition from $M_i^{n_i}$ to M_{i+1} then its encoding $p((k+1)-)$ can be transformed into a token $p(k+1)$ or $p((k+1)+)$ in the time-passing phase of N'. Furthermore, all tokens in $M_i^{n_i}$ with fractional part 0 are transformed into tokens with a strictly positive fractional part in M_{i+1}, since $x_i > 0$. In N' this is encoded by the fact that all $p(k)$ tokens become $p(k+)$ tokens in the time-passing phase. Finally, all tokens are removed from p_{count} in the time-passing phase. Thus the resulting marking $int_{\delta_{i+1}}(M_{i+1})$ is a standard marking again. ∎

The reverse implication of Lemma 3.9 does not generally hold. The fact that $int_\delta(M) \in INF'$ for some marking M of a TPN N does not imply that there is an infinite M-computation in the corresponding TPN. The infinite $int_\delta(M)$-computation in N' depends on the fact that the $p(k-)$ tokens do (or don't) become $p(k)$ or $p(k+)$ tokens at the right step in the computation. For example, in an infinite computation taking time 0.5, two different TPN tokens $(p, 0.8)$ and $(p, 0.9)$ are both interpreted as $p(1-)$ in N'. However, $(p, 0.8)$ cannot become $(p, 1)$ by aging unless $(p, 0.9)$ becomes $(p, 1.1)$, while their symbolic encodings $p(1-)$ can become $p(1)$ or $p(1+)$ in any order.

To establish a reverse correspondence between markings of N' and markings of N we need the following definitions.

Definition 3.10. Consider a TPN $N = (P, T, In, Out)$. Let N' be the corresponding SD-TN with places $P' = \{p(sym) \mid p \in P, sym \in Sym\} \cup \{p_{disc}, p_{time1}, p_{time2}, p_{count}\}$ and a standard marking $M' : P' \to \mathbb{N}$. Let M'^-, M'^+ be the sub-markings of M' defined as follows.

- $M'^-(p(k-)) = M'(p(k-))$ for each place of the form $p(k-)$ in P'; $M'^-(p(k+)) = 0$ and $M'^-(p(k)) = 0$ for each place of the form $p(k+)$ and $p(k)$ in P', respectively. $M'^-(p_x) = 0$ for any $p_x \in \{p_{disc}, p_{time1}, p_{time2}, p_{count}\}$.
- $M'^+(p(k+)) = M'(p(k+))$ for each place of the form $p(k+)$ in P'. But $M'^+(p(k-)) = 0$ and $M'^+(p(k)) = 0$ for each place of the form $p(k-)$ and $p(k)$ in P', respectively. $M'^+(p_x) = 0$ for any $p_x \in \{p_{disc}, p_{time1}, p_{time2}, p_{count}\}$.

Let $perm(M'^-)$ be the set of all words

$$w_- = b_1 \bullet \ldots \bullet b_n \in \left((P \times \{0, \ldots, max - 1\})^{\odot} - \{\emptyset\} \right)^*$$

such that for all p and $k < max$ we have that $M'^-(p((k+1)-)) = b_1((p, k)) + \ldots + b_n((p, k))$.

Similarly, let $perm(M'^+)$ be the set of all words

$$w_+ = b_1 \bullet \ldots \bullet b_n \in \left((P \times \{0, \ldots, max - 1\})^{\odot} - \{\emptyset\} \right)^*$$

such that for all p and $k < max$, we have $M'^+(p((k)+)) = b_1((p, k)) + \ldots + b_n((p, k))$.

Intuitively, $perm(M'^-)$ describes all possible permutations of the fractional parts of (the ages of) tokens in a TPN marking M which are symbolically encoded as $p(k-)$ tokens in the corresponding SD-TN standard marking M'. Note that several different tokens can have the same fractional part. Similarly, the set $perm(M'^+)$ describes all possible permutations of the fractional parts of (the ages of) tokens in a TPN marking M which are symbolically encoded as $p(k+)$ tokens in the corresponding SD-TN standard marking M'.

Example 3.11. Let $max = 1$. Consider $M' = [p_{disc}, p(1), q(1+), p(0+), q(1-), q(1-)]$. Then $perm(M'^-) = \{[(q, 0)] \bullet [(q, 0)], [(q, 0), (q, 0)]\}$ and $perm(M'^+) = \{[(p, 0)]\}$. Notice that $q(1+)$ does not belong to $perm(M'^+)$, since $max = 1$.

Every standard marking M' of the SD-TN defines a set of TPN markings, depending on which permutation of the fractional parts of the ages of the $p(k-)$-encoded tokens and $p(k+)$-encoded tokens is chosen.

Definition 3.12. Let N' be a SD-TN. For every standard marking $M' : P' \to \mathbb{N}$ we define a multi-region upward closure (MRUC) $Reg(M')$ as follows. The MRUC $Reg(M')$ contains all regions $Reg(M', w_+, w_-)$ of the form $(b_0, w_+ \bullet w_-, b_{max})$, where $b_0((p, k)) = M'(p(k))$ for all p and all $k \le max$, $w_+ \in perm(M'^+)$, $w_- \in perm(M'^-)$ and $b_{max}(p) = M'(p(max+))$ for all p.

Example 3.13. Consider $M' = [p_{disc}, p(1), q(1+), p(0+), q(1-), q(1-)]$ and sets $perm(M'^+)$, $perm(M'^-)$ of Example 3.11. $Reg(M')$ consists of the 2 regions shown in Figure 6.

Next we show how an infinite disc-computation of the SD-TN corresponds to a zeno computation in the TPN which starts with a discrete transition.

Lemma 3.14. *Let N be a TPN with corresponding SD-TN N' and $M' \in INF'$. Then*

$$\exists w_- \in perm(M'^-). \forall w_+ \in perm(M'^+). [\![Reg(M', w_+, w_-)]\!]^{\uparrow} \subseteq \bigcup_{\delta > 0} ZENO^{1-\delta} \subseteq ZENO$$
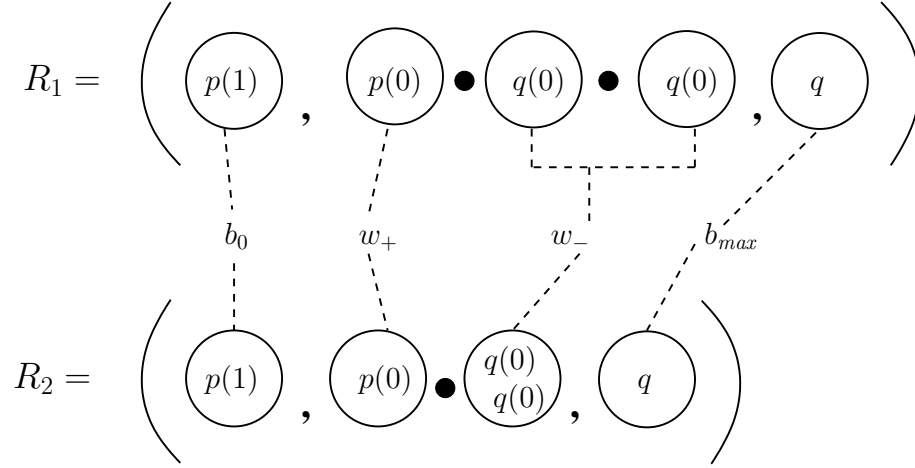
Figure 6: $Reg(M') = \{R_1, R_2\}$

Proof. Since $M' \in INF'$, there is an infinite M'-computation $\pi' = M' \to M'_1 \to M'_2 \to \dots$. The first transition in π' is a discrete transition, since M' is a standard marking. The computation π' contains a (possibly infinite) number of time-passing phases (where the control-token shifts to the place p_{time1} and then p_{time2}) $tpp_1, tpp_2, \dots$. Now consider the *original* $p(k-)$ tokens in M' which become $p(k)$ tokens or $p(k+)$ tokens in the i-th time-passing phase tpp_i. Other tokens which were *newly created* during the computation π' are not considered here. (They will be treated differently; see below). Let α_i be the multiset of $p(k-)$ tokens in M' which become $p(k+)$ tokens in tpp_i and β_i the multiset of $p(k-)$ tokens in M' which become $p(k)$ tokens in tpp_i. (Note that this does not happen by the transfer transition, but by normal transitions in second part of the time-passing phase, where the control-token is on place p_{time2}.) We have $\alpha_i, \beta_i \leq M'^-$, but not necessarily $\Sigma_{i \in \mathbb{N}}(\alpha_i + \beta_i) = M'^-$, because $p(k-)$ tokens can also be used by normal transitions in the discrete phase or never become $p(k)$ or $p(k+)$ tokens at all. Let $\gamma := M'^- - \Sigma_{i \in \mathbb{N}}(\alpha_i + \beta_i)$. Since M'^- is finite, there exists a smallest number m such that $\alpha_i + \beta_i = \emptyset$ for all $i > m$. It follows that there exists an infinite suffix π'' of π' such that in π'' no *original* $p(k-)$ token of M' becomes a $p(k)$ or $p(k+)$ token.

We define $w_- \in perm(M'^-)$ by $w_- := \gamma \bullet \beta_m \bullet \alpha_m \bullet \cdots \bullet \beta_1 \bullet \alpha_1$.

We need to prove that

$$\forall w_+ \in perm(M'^+). [\![Reg(M', w_+, w_-)]\!]^\uparrow \subseteq \bigcup_{\delta > 0} ZENO^{1-\delta}$$

For this it suffices to show that $[\![Reg(M', w_+, w_-)]\!] \subseteq \bigcup_{\delta > 0} ZENO^{1-\delta}$, because $ZENO^{1-\delta}$ is upward-closed. Now let $w_+ \in perm(M'^+)$ and let $M \in [\![Reg(M', w_+, w_-)]\!]$. We need to show that $M \in ZENO^{1-\delta}$ for some $\delta > 0$, i.e., that there exists an infinite M-computation π with $\Delta(\pi) < 1 - \delta$.

Since $M \in [\![Reg(M', w_+, w_-)]\!]$ there exists a δ with $0 < \delta < 1$ and $int_\delta(M) = M'$. By our assumption above, $M' \in INF'$ is a standard marking where an infinite computation π' starts. The computation π' begins with a normal transition (not a time-passing phase),

since M' is a standard marking. Based on this π', we now construct an infinite M-disc-computation π with $\Delta(\pi) < 1 - \delta$.

A crucial feature of the construction of this particular M-disc-computation π is the order of the fractional parts of the ages of tokens. While this order is given for the tokens already present in M, it can be chosen conveniently (i.e., as needed) for those tokens which are newly created during π. The main ideas for this construction are the following:

- Since $\Delta(\pi) < 1$, for any token it can happen at most once during π that it reaches the next higher integer age by aging. In particular, initially present tokens which are interpreted as $p(k-)$ may age to $p(k)$ or $p(k+)$, but not to $p((k+1)-)$ or higher during π.
- All time intervals on transition arcs in the timed Petri net have integer bounds (see Section 2). Thus one can have intervals like $(1 : 4]$ or $[2 : 7)$, but not $[1.3 : 2.1]$. This means that if a token is newly created during π then the fractional part of its age can be chosen nondeterministically arbitrarily closely to the next higher integer. For example, if a token is created by an output arc labeled $[1 : 2)$ then its age could be 1.7, 1.9, 1.99, or 1.99999, etc. Consider an already existing token with an age whose fractional part is a nonzero value x. Now another token is newly created, and let y be the fractional part of its age. Then all cases $y < x$, $y > x$ and $y = x$ are possible, e.g., $y = x/2$ or $y = x + (1-x)/2$, or $y = x$. This means that the newly created token could reach the next higher integer age before, after, or at the same time as the old token, depending on which value y is chosen. For each of these scenarios there is a computation in with the fractional part y is chosen to implement it. In general, for any permutation of the orders of the fractional parts of the ages of newly created tokens (w.r.t. already existing tokens and each other), there is some computation in which their ages are chosen to create this order. Of course, this only applies to tokens which exist at the same time in the net during the computation π, not those who are created (directly or indirectly) by each other.

The computation π has the form $M \longrightarrow_D M_{j_1} \to M_{j_2} \to \dots$ where the sequence $\{j_i\}_{i \in \mathbb{N}}$ is a subsequence of $1, 2, \dots$ (it skips the intermediate steps in the time-passing phases of π') and $M'_{j_i} = int_{\delta_{j_i}}(M_{j_i}) + \{p^n_{count}\}$ (for some $n \geq 0$) and $\delta_{j_i} = \delta + \Delta(M \longrightarrow M_{j_1} \longrightarrow M_{j_2} \longrightarrow \dots \longrightarrow M_{j_i})$. (The first transition in π is a discrete transition, since also the first transition in π' is one.)

For every simulation of a discrete transition of N in π' (i.e., not in the time-passing phase) of the form $M'_i \to M'_{i+1}$ where $M'_i = int_{\delta_i}(M_i) + \{p^n_{count}\}$ (for some $n \geq 0$) there is a corresponding discrete transition in π of the form $M_i \longrightarrow_D M_{i+1}$ where $\delta_{i+1} = \delta_i$ and $M'_{i+1} = int_{\delta_{i+1}}(M_{i+1}) + \{p^{n+1}_{count}\}$. This follows directly from Def. 3.1. (Note that the extra parts with $\{p^n_{count}\}$ and $\{p^{n+1}_{count}\}$ are necessary. For technical reasons, the SD-TN counts the number of discrete transitions since the last time-passing phase, while the functions int_{δ_i} always return standard markings without tokens on p_{count}.)

Now we consider the i'-th time-passing phase for $1 \leq i' \leq m$. (Recall the definition above that m is the index number of the last time-passing phase where *original* $p(k-)$ tokens of M' change into $p(k)$ or $p(k+)$ tokens. The remaining case of $i' > m$ will be considered later.) For every sequence of transitions $M'_i \overset{*}{\longrightarrow} M'_l$ in π' representing the i'-th time-passing phase there is a corresponding single time-transition in π of the form $M_i \longrightarrow_{\varepsilon_{i'}} M_l$, where $M'_i = int_{\delta_i}(M_i) + \{p^n_{count}\}$ (for some $n \geq 1$), $\delta_l = \delta_i + \varepsilon_{i'}$ and $M'_l = int_{\delta_l}(M_l)$. (Note that M'_i must contain at least one token on p_{count} for the time-passing phase to start there and thus $n \geq 1$. On the other hand, M'_l is a standard marking, since it is reached at the end of a time-passing phase and thus does not contain any tokens on p_{count}.) The

delay $\varepsilon_{i'}$ is chosen as $\varepsilon_{i'} := 1 - f_{i'}$ where $f_{i'}$ is the fractional part of the age of those tokens in M_i which are mapped to $\beta_{i'}$ by int_{δ_i}. This ensures that in this timed transition the right tokens (of those originally present in M) reach (those mapped to $\beta_{i'}$) or exceed (those mapped to $\alpha_{i'}$) the next higher integer age. For the other tokens of M_i, which were newly created during π we can arbitrarily choose the values of their fractional parts, i.e., for every combination of these values there is a possible computation which implements it. Thus one can assume that these fractional parts are conveniently chosen such that they do (or don't) reach (or exceed) the next higher integer age, just as required by the condition $int_{\delta_l}(M_l) = M_l'$. Since $int_\delta(M) = M'$, only those tokens in M with a fractional part $> \delta$ were mapped to $p(k-)$ tokens in M' and only those tokens can reach (or exceed) age k in π. Therefore it follows from our choice of the $\varepsilon_{i'}$ for $i' \leq m$ that $\sum_{i'=1}^m \varepsilon_{i'} < 1 - \delta$. Thus we get $\lambda := (1 - \delta) - \sum_{i'=1}^m \varepsilon_{i'} > 0$. (The quantity λ will be used to determine the $\varepsilon_{i'}$ for $i' > m$.)

Now we consider the i'-th time-passing phase for $i' > m$. These are the time-passing phases in the infinite suffix π'' of π' mentioned above. For them, it works like the case above, except that the delays $\varepsilon_{i'}$ do no longer depend on the initial marking M, because $\alpha_{i'} + \beta_{i'} = \emptyset$ for $i' > m$. As shown above, none of the original tokens of M are involved in these i'-th time-passing phases for $i' > m$. The only tokens involved in this (reaching or exceeding the next higher integer age in this phase) are tokens newly generated in π (which have an age greater than δ and are mapped to $p(k-)$). As explained above, the fractional parts of their ages can be chosen conveniently (i.e., as needed) such that they reach or exceed the next higher integer age exactly as required for the correspondence with the computation π'. In particular, their ages can be chosen arbitrarily close to the next higher integer age such that the required delays $\varepsilon_{i'}$ (for $i' > m$) can be made arbitrarily small. We choose $\varepsilon_{i'} := (\lambda/2) * 2^{-i'}$ for $i' > m$.

So we obtain $\Delta(\pi) = \sum_{i' \in \mathbb{N}} \varepsilon_{i'} = \sum_{1 \leq i' \leq m} \varepsilon_{i'} + \sum_{i' > m} \varepsilon_{i'} \leq \sum_{1 \leq i' \leq m} \varepsilon_{i'} + \lambda/2 < \sum_{1 \leq i' \leq m} \varepsilon_{i'} + \lambda = 1 - \delta$. Thus $\Delta(\pi) < 1 - \delta$ and $M \in ZENO^{1-\delta}$, as required. $\square$

Now we describe the algorithm to compute the set $ZENO$ as a multi-region upward closure. The algorithm computes a MRUC Z, given by Definition 3.15, and we prove in Lemma 3.16 and Lemma 3.17 that $[\![Z]\!] = ZENO$.

Definition 3.15. Let N be a TPN with corresponding SD-TN N'.

$$Z := \bigcup_{M' \in INF'_{min}} \bigcup_{w_+ \in perm(M'^+)} \bigcap_{w_- \in perm(M'^-)} Pre^*(\{Reg(M', w_+, w_-)\})$$

3.3. **Proof of Correctness.** We need to show that Z is effectively constructible and that $[\![Z]\!] = ZENO$.

The constructibility of Z requires the following steps.

- The set INF'_{min} is finite and effectively constructible. This will be shown in Subsection 3.4.
- For any $M' \in INF'_{min}$ the sets $perm(M'^+)$ and $perm(M'^-)$ are finite and effectively constructible. This follows directly from Definition 3.10 and the finiteness of M'.
- Since $Reg(M', w_+, w_-)$ is a region, we can interpret $\{Reg(M', w_+, w_-)\}$ as a MRUC. Then $Pre^*(\{Reg(M', w_+, w_-)\})$ can be effectively constructed as a MRUC by Lemma 2.12. (Note that Pre^* is computed w.r.t. the relation $\longrightarrow = \longrightarrow_\delta \cup \longrightarrow_D$ which includes both timed- and discrete transitions. Thus the zeno-computations starting from markings in $[\![Z]\!]$ may also start with a timed transition.)

- By Lemma 2.10, the finite union and intersection operations on MRUC are computable and yield a MRUC Z.

  Now we show that $[\![Z]\!] = ZENO$.

Lemma 3.16. $[\![Z]\!] \subseteq ZENO$.

Proof. Let $M \in [\![Z]\!]$. Then there is an $M' \in INF'_{min}$ and a sequence $w_+ \in perm(M'^+)$ such that $M \in [\![\bigcap_{w_- \in perm(M'^-)} Pre^*(\{Reg(M', w_+, w_-)\})]\!]$.

We choose the sequence $w_- \in perm(M'^-)$ according to Lemma 3.14 and so obtain $M \in [\![Pre^*(\{Reg(M', w_+, w_-)\})]\!]$ and $[\![Reg(M', w_+, w_-)]\!]^\uparrow \subseteq ZENO$. Thus $M \in ZENO$, since $Pre^*(ZENO) = ZENO$. $\qquad\square$

Lemma 3.17. $ZENO \subseteq [\![Z]\!]$.

Proof. Let $M \in ZENO$. By the definition of *zeno-marking*, there exists an infinite M-computation π and a finite number m such that $\Delta(\pi) \le m$. It follows that there exists an infinite suffix of π that takes only $< 1/2$ time. Thus there exists a marking M_1 such that $M \xrightarrow{*} M_1$ and an infinite M_1-computation π_1 with $\Delta(\pi_1) < 1/2$. Since M_1 contains finitely many tokens and π_1 is infinite, there exists an infinite suffix of π_1 such that none of the original tokens of M_1 is used in this infinite suffix (although some might still be present; these are represented by M_4, see below). Since every infinite computation must contain infinitely many discrete transitions (see Section 2), there exists an infinite suffix of this infinite suffix of π_1 which starts with a discrete transition.

Thus there exist markings M_2, M_3 and M_4 and a finite computation π_2 such that

- $M_1 \xrightarrow{\pi_2} M_2 = M_3 + M_4$
- All tokens in M_3 were created during π_2.
- There is an infinite M_3-disc-computation π_3 with $\Delta(\pi_2\pi_3) < 1/2$, and thus $\Delta(\pi_3) < 1/2$.

Let $M'_3 := int_{1/2}(M_3)$. Then we have $M'_3 \in INF'$ by Lemma 3.9, since π_3 is an infinite disc-computation. From Definition 3.12, we have that there are permutations $w_+ \in perm(M_3'^+)$ and $w_- \in perm(M_3'^-)$ such that $M_3 \in [\![Reg(M'_3, w_+, w_-)]\!]$.

Since $M'_3 \in INF'$ and INF' is upward-closed (in Ω'; see Def. 3.6), there exists a marking $M''_3 \in INF'_{min}$ such that $M''_3 \le M'_3$. Therefore $M_3''^+ \le M_3'^+$, $M_3''^- \le M_3'^-$ and $perm(M_3''^+) \subseteq perm(M_3'^+)$ and $perm(M_3''^-) \subseteq perm(M_3'^-)$.

This means that there also exist permutations $w'_+ \in perm(M_3''^+)$ with $w'_+ \le^w w_+$ and $w'_- \in perm(M_3''^-)$ with $w'_- \le^w w_-$ (see Def. 2.1) and thus $[\![Reg(M''_3, w'_+, w'_-)]\!]^\uparrow \supseteq [\![Reg(M'_3, w_+, w_-)]\!]^\uparrow$. It follows that $M_3 \in [\![Reg(M'_3, w_+, w_-)]\!] \subseteq [\![Reg(M'_3, w_+, w_-)]\!]^\uparrow \subseteq [\![Reg(M''_3, w'_+, w'_-)]\!]^\uparrow$.

Now consider all those tokens in M_3 which are mapped to $p(k-)$ tokens in M'_3, i.e., those with a fractional part of their age which is $> 1/2$. These tokens (like all others in M_3) were all created during π_2 and none of them had an integer age during π_2, because $\Delta(\pi_2) < 1/2$. Thus, the fractional parts of their ages are totally independent and any permutation is possible, i.e., for any permutation there is a computation which implements it (for the reasons explained in the proof of Lemma 3.14).

Therefore, for *every* $w_- \in perm(M_3'^-)$ there is a marking $M_3^{w_-}$ in N such that

- $M_1 \xrightarrow{*} M_3^{w_-} + M_4$
- $M_3^{w_-} \in [\![Reg(M'_3, w_+, w_-)]\!]$.

Since $M_3'' \leq M_3'$ we have that for every $w_- \in perm(M_3''^-)$ there is a corresponding $w_- \in perm(M_3'^-)$ with $w_-' \leq^w w_-$, i.e., w_-' is the restriction of w_- to M_3''. It then follows from the property above that for *every* $w_-' \in perm(M_3''^-)$ there is a marking $M_3^{w_-'} := M_3^{w_-}$ in N s.t.

- $M_1 \xrightarrow{*} M_3^{w_-'} + M_4$
- $M_3^{w_-'} \in [\![Reg(M_3'', w_+', w_-')]\!]^\uparrow$.

It follows that for *every* $w_-' \in perm(M_3''^-)$ we have $M_3^{w_-} + M_4 \in [\![Reg(M_3'', w_+', w_-')]\!]^\uparrow$ and thus $M_1 \in Pre^*(\{Reg(M_3'', w_+', w_-')\})$. Since $M \in Pre^*(M_1)$ we finally obtain

$$M \in \left[\!\!\left[\bigcap_{w_-' \in perm(M_3''^-)} Pre^*(\{Reg(M_3'', w_+', w_-')\}) \right]\!\!\right]$$

with $M_3'' \in INF_{min}'$ and $w_+' \in perm(M_3''^+)$, and thus $M \in [\![Z]\!]$. $\square$

By Lemma 3.16 and Lemma 3.17 we have that $ZENO = [\![Z]\!]$. It remains to show that INF_{min}' is effectively constructible.

3.4. **Step 2: Computing INF_{min}'.** Computability of the set $ZENO$ (in the last section) requires that the minimal elements of any upward closed set is effectively constructible. In this section, we show for any SD-TN, how to construct the set of minimal elements INF_{min} of INF. Then INF_{min}' is obtained by just restricting INF_{min} to standard markings (see Def. 3.6).

For constructing INF_{min}, we use a result by Valk and Jantzen [VJ85]. Our algorithm depends on the concepts of semi-linear languages, Presburger Arithmetic, Parikh's Theorem and Dickson's Lemma, described in the following. Recall that we use $(v_1, \ldots, v_n)$ or $\vec{v}$ interchangeably to denote a vector of size n.

Lemma 3.18. (Dickson's Lemma [Dic13])
For every infinite sequence of vectors $\vec{x_1}, \vec{x_2}, \vec{x_3}, \ldots$ in $\mathbb{N}^n$ there exists an infinite non-decreasing subsequence. In particular, there exist indices i, j with $i < j$ s.t. $\vec{x_i} \leq \vec{x_j}$ ($\leq$ taken component-wise).

3.4.1. *Semilinear Sets.* First we define *linear* sets.

Definition 3.19. A set $L \subseteq \mathbb{N}^n$ is called *linear*, if there exist vectors $\vec{v_0}, \vec{v_1}, \ldots, \vec{v_m} \in \mathbb{N}^n$ such that

$$L = \left\{ \vec{v_0} + \sum_{i=1}^{m} k_i \vec{v_i} \quad | \quad k_1, \ldots, k_m \in \mathbb{N} \right\}$$

We denote this linear set by $L = L(\vec{v_0}; \vec{v_1}, \ldots, \vec{v_m})$.

Example 3.20. $L((0,0); (0,2), (2,0)) = \{(0,0) + k_1(0,2) + k_2(2,0) | k_1, k_2 \in \mathbb{N}\}$ is linear.

Definition 3.21. A subset of $\mathbb{N}^n$ is called *semilinear* if it is a finite union of linear sets.

Theorem 3.22. [Gin66] *Semilinear sets are closed under union, intersection, complementation and first-order quantification.*

Next we define the Parikh mapping φ. Given a finite alphabet $\Sigma = \{a_1, \ldots, a_n\}$, φ is a function from Σ^* to $\mathbb{N}^n$, defined by $\varphi(w) = (\#_{a_1}(w), \ldots, \#_{a_n}(w))$, where $\#_{a_i}(w)$ is the number of occurrences of a_i in w. Thus $\varphi(\epsilon) = (0, \ldots, 0)$ and $\varphi(w_1 \bullet \ldots \bullet w_m) = \sum_{i=1}^{m} \varphi(w_i)$. Finally, given a language $L \subseteq \Sigma^*$, $\varphi(L) = \{\varphi(w) | \ w \in L\}$. If $\varphi(L)$ is semilinear for a language L, then L is called a semilinear language.

Theorem 3.23. *(Parikh's Theorem)* [Par66] $\varphi(L)$ *is effectively semilinear for each context-free language L.*

As a special case, Theorem 3.23 holds for regular languages, since every regular language is a context-free language [Par66].

Example 3.24. Let $\Sigma = \{a_1, a_2, a_3\}$.
Then $\varphi(a_1 a_2 a_1 a_3 a_2 a_3 a_3) = (2, 2, 3) \in L((2, 0, 1); (0, 1, 1))$.
Also, $\varphi(ab^*ca) = \{(2, 0, 1) + n * (0, 1, 0) | \ n \in \mathbb{N}\}$.

3.4.2. *Presburger Arithmetic.* Presburger arithmetic is the first-order theory of the integers with addition and the ordering relation over $\mathbb{Z}$, also denoted as $(\mathbb{Z}, \leq, +)$. Formally, Presburger arithmetic is the first-order theory over *atomic formulae* of the form

$$\sum_{1 \leq i \leq n} a_i x_i \quad \sim c$$

where a_i, c are integer constants, x_i-s are variables ranging over integers and $\sim$ is a comparison operator, where $\sim \in \{=, \neq, <, \leq, >, \geq\}$. This means that a *Presburger formula* ρ is either an *atomic formula*, or it is constructed from the Presburger formulae ρ_1, ρ_2 recursively as follows:

$$\rho := \neg \rho_1 \ | \ \rho_1 \wedge \rho_2 \ | \ \rho_1 \vee \rho_2 \ | \ \exists x_i. \rho_1(x_1, \ldots, x_n)$$

where $\rho_1(x_1, \ldots, x_n)$ is a Presburger formula over free variables $x_1, \ldots, x_n$ and $1 \leq i \leq n$.

Theorem 3.25. (Presburger) [BA93] *Presburger arithmetic is decidable.*

As a shorthand notation, we work with $\mathbb{Z}_\omega = \mathbb{Z} \cup \{\omega\}$ instead of the usual $\mathbb{Z}$, where ω is the first limit ordinal. This is not a problem, since Presburger-arithmetic on $\mathbb{Z}_\omega$ can easily be reduced to Presburger-arithmetic on $\mathbb{Z}$ as follows. For every variable x one adds an extra variable x' which is used in such a way that the original state $x = k < \omega$ is represented by $(x, x') = (k, 0)$ and the original state $x = \omega$ is represented by $(x, x') = (0, 1)$. It is easy to encode the usual properties like $\omega + k = \omega - k = \omega + \omega = \omega$.

Theorem 3.26. [GS66] *A subset of $\mathbb{N}^n$ is semilinear iff it is definable in Presburger Arithmetic.*

3.4.3. *Result from Valk and Jantzen.*

We recall a result from [VJ85].

Theorem 3.27. *(Valk & Jantzen [VJ85])   Given an upward-closed set $V \subseteq \mathbb{N}^k$, the finite set V_{min} of minimal elements of V is effectively computable iff for any vector $\vec{u} \in \mathbb{N}_{\omega}^k$ the predicate $\vec{u} \downarrow \cap V \neq \emptyset$ is decidable.*

Proof. Assume that the minimal elements of V, denoted by V_{min} can be computed. Then $V = V_{min} + \mathbb{N}^k$ gives a semilinear representation of V. Since $\vec{u} \downarrow$ is also a semilinear set, a representation of which can be found effectively, the predicate $\vec{u} \downarrow \cap V \neq \emptyset$ is decidable.

On the other hand, assume that the predicate is decidable for any vector $\vec{u} \in \mathbb{N}_{\omega}^k$. The following method then effectively constructs V_{min}. First start with a singleton set of vectors $W_0 := \{(\omega, \ldots, \omega)\}$ with k ω-s. Let W_i be the set of vectors that we need to consider in the i-th iteration and $\mathcal{V}_i$ the set of minimal elements found for V_{min} in the i-th iteration. Initially $\mathcal{V}_0 := \emptyset$. We let $pred_V(\vec{u})$ denote $\vec{u} \downarrow \cap V \neq \emptyset$. We repeat the following.

  Stage 1: In this stage, we perform the following two loops sequentially.
  Loop 1: We choose some vector $\vec{u}$ from W_i and compute $pred_V(\vec{u})$. If the value is false, then we remove u from W_i. We get out of this loop if $pred_V(\vec{u})$ is true or $W_i = \emptyset$.
      After exiting from the above loop if $W_i = \emptyset$, then $V_{min} = \mathcal{V}_i$ and we stop the algorithm. Otherwise, $pred_V(\vec{u})$ is true; $\vec{u} \downarrow$ contains at least one element of V_{min} and one such element will be found in the next loop.
  Loop 2: We repeat the following until all coordinates of $\vec{u}$ are considered. Choose some coordinate $\vec{u}(i)$ of $\vec{u}$ which has not yet been considered and replace $\vec{u}(i)$ in $\vec{u}$ by the smallest natural number such that $pred_V(\vec{u})$ for this new vector is still true.
      The above computed new vector will then be an element of V_{min}. So, we update $\mathcal{V}_{i+1} = \mathcal{V}_i \cup \{\vec{u}\}$.

  Stage 2: Let the new found vector be $\vec{u} = (z_1, \ldots, z_k)$. In this stage, we try to find other vectors in V_{min}. We let

$$W_i' = \left\{ (z_1', \ldots, z_k') \in \mathbb{N}_{\omega}^k \mid \exists j : 1 \leq j \leq k : z_j' = z_j - 1 \ \wedge \ \forall m \neq j.\, z_m' = \omega \right\}.$$

We update $W_{i+1} := min(W_i, W_i')$ where $min(W, W') = \left\{ min(\vec{u}, \vec{u'}) \mid \vec{u} \in W, \vec{u'} \in W' \right\}$ and *min* of two vectors are evaluated component-wise. Then we increment the iterator by $i := i + 1$ and go back to Loop 1. □

3.4.4. *Computing INF_{min} for a Petri net.* While a marking of a normal untimed Petri net (or a SD-TN) is a mapping $M : P \to \mathbb{N}$ (see Def. 3.1), an ω-marking is defined as a mapping $M : P \to \mathbb{N}_{\omega}$, where $\mathbb{N}_{\omega} = \mathbb{N} \cup \{\omega\}$. In the following we work with ω-markings, i.e., when we speak of markings these may be ω-markings.

For any Petri net N let *INF* be the set of markings where infinite runs start, and INF_{min} the finite set of minimal elements of *INF*, similarly as for SD-TN in Def. 3.6. We use the result of Valk and Jantzen to compute INF_{min} for a Petri net. To apply this algorithm, we require the computability of the predicate $M \downarrow \cap INF \neq \emptyset$ ($pred_{INF}(\vec{M})$) for any ω-marking M. The decidability of this predicate was first shown in [BM99]. We include a description of this construction here (adapted to our notation), because the more

general construction for SD-TN in the next section is based on it and would be hard to understand without it.

Definition 3.28. (Coverability graph) [KM69]
Given a Petri net N (with k places) with initial ω-marking M_0, the Karp-Miller coverability graph is a finite directed graph $\mathcal{C} = (G, \rightarrow)$ with $G \subseteq \mathbb{N}_\omega^k$ whose vertices are labeled with ω-markings of N. It is constructed as follows.

Starting from M_0, one begins to construct the (generally infinite) computation graph of N, i.e., the graph of reachable markings, connected by arcs representing fired transitions. However, if one encounters a marking M_2 which is strictly bigger than a previously encountered marking M_1 (i.e., $M_2 \geq M_1$ and $M_2 \neq M_1$) then one replaces M_2 by $M_2 + \omega(M_2 - M_1)$. This describes the effect that by repeating the sequence of transitions between M_1 and M_2 one could reach markings with arbitrarily many tokens on those places p where $M_2(p) > M_1(p)$. (Note that such sequences can be repeated because Petri nets are monotonic.) If one encounters the same ω-marking as previously, then one creates a loop.

It follows from Dickson's Lemma (see Lemma 3.18) that the generated graph is finite and the construction terminates.

The following properties of the coverability graph follow directly from the construction (see [KM69]).

Lemma 3.29.
(1) *For every marking M, reachable from the initial marking M_0, there is an ω-marking $M_{\mathcal{C}}$ in the coverability graph such that $M \leq M_{\mathcal{C}}$.*
(2) *For every ω-marking $M_{\mathcal{C}}$ in $\mathcal{C}$, there are markings M reachable from M which contain arbitrarily large numbers of tokens in the places with ω in $M_{\mathcal{C}}$.*
(3) *The arcs in the coverabiliy graph are induced by the transitions in the Petri net. If it is possible to fire some sequence of transitions from a marking $M_{\mathcal{C}}$ in the coverability graph, leading to a marking $M_{\mathcal{C}}'$, then there is a reachable marking $M \leq M_{\mathcal{C}}$ in the Petri net which can fire the same sequence of transitions, leading to a marking $M' \leq M_{\mathcal{C}}'$.*

Definition 3.30. (Effect Vector) To every transition t in a normal untimed Petri net with k places one can associate a vector $\vec{v}_t \in \mathbb{Z}^k$ which describes the effect of the transition on the markings of the net, i.e., the change in the marking caused by firing the transition. This means that if $M_1 \xrightarrow{t} M_2$, then $M_2 = M_1 + \vec{v}_t$. We call $\vec{v}_t$ the *effect-vector* of transition t.

Lemma 3.31. [BM99] *Given a Petri net N with k places and an ω-marking $M_0 \in \mathbb{N}_\omega^k$ where $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ and ω denotes the first limit ordinal (satisfying $z + \omega = z - \omega = \omega$ for $x \in \mathbb{N}$), it is decidable if $M_0 \downarrow \cap INF \neq \emptyset$.*

Proof. We show that if $M_0 \downarrow \cap INF \neq \emptyset$ then this condition will be detected by the following construction. Furthermore, we prove that the construction does not yield any false positives.
Construction:
Let $\mathcal{C} = (G, \rightarrow)$ with $G \subseteq \mathbb{N}_\omega^k$ be the coverability graph of N from the initial marking M_0, which is computable (see Def. 3.28 and [KM69]).

The main idea is to analyze the coverability graph $\mathcal{C}$ and look for a cycle s.t. the transitions fired in this cycle have a combined positive effect on the marking (and will thus be repeatable). It will be shown that such a cycle in $\mathcal{C}$ exists if and only if $M_0 \downarrow \cap INF \neq \emptyset$.

First, for every ω-marking M in the coverability graph $\mathcal{C}$, we compute a finite-state automaton $\mathcal{A}_M$ as follows.

- The transition graph of $\mathcal{A}_M$ is the largest strongly connected subgraph of $\mathcal{C}$ containing M.
- The initial state of $\mathcal{A}_M$ is M.
- $\mathcal{A}_M$ has only one final state, which is also M.
- Let l be the number of edges in $\mathcal{A}_M$. We label every arc in $\mathcal{A}_M$ with a unique symbol Λ_i for $i : 1 \leq i \leq l$. To every symbol Λ_i, we associate the *effect-vector* (see Def. 3.30) $\vec{\zeta_i} \in \mathbb{Z}^k$ that describes the effect of the transition that was fired in the step from one node to the other.

Let $L(\mathcal{A}_M)$ be the regular language (over alphabet $\{\Lambda_i \,|\, 1 \leq i \leq l\}$) recognized by $\mathcal{A}_M$. The aim is to find a cyclic path in $\mathcal{A}_M$ from a marking M back to M where the sum of all the effect-vectors of all traversed arcs is $\geq \vec{0}$. This cyclic path is not necessarily a simple cycle. The effect-vector of an arc that is traversed j times is counted j times. Such a cyclic path with positive overall effect is repeatable infinitely often and thus corresponds to a possible infinite computation of the system N.

Given the automaton $\mathcal{A}_M$ with M as its initial and the only final state, every word in $L(\mathcal{A}_M)$ corresponds to a cyclic path from M to M. For any word w, let $|w|_{\Lambda_i}$ be the number of occurrences of Λ_i in w. The question now is if there is a word $w \in L(\mathcal{A}_M)$ such that

$$\sum_{1 \leq i \leq l} |w|_{\Lambda_i} \vec{\zeta_i} \geq \vec{0}$$

Such words characterize loops starting and ending in the same node of the coverability graph. We show how to answer the above question in the following.

- First we compute the Parikh image of $L(\mathcal{A}_M)$, i.e., the set $\{(|w|_{\Lambda_1}, \ldots, |w|_{\Lambda_l}) \,|\, w \in L(\mathcal{A}_M)\}$. This set is effectively semilinear by Parikh's Theorem.
- By Theorem 3.26, we compute a Presburger formula $\rho(x_1, \ldots, x_l)$ from the semilinear set computed above. The variables $x_1, \ldots, x_l$ count the number of times each edge Λ_i appears in a word $w \in L(\mathcal{A}_M)$.
- Finally, to decide if $\sum_{1 \leq i \leq l} |w|_{\Lambda_i} \vec{\zeta_i} \geq \vec{0}$, we check the satisfiability of $\rho_{\mathcal{A}} = \rho(x_1, \ldots, x_l) \wedge \sum_{1 \leq i \leq l} x_i \vec{\zeta_i} \geq \vec{0}$, which is again a Presburger formula. By Theorem 3.25, we can decide whether this formula is satisfiable.

For every marking M in the coverability graph $\mathcal{C}$ (these are finitely many) we check this condition for the automaton $\mathcal{A}_M$ and we say that $M_0 {\downarrow} \cap INF \neq \emptyset$ is true if and only if the condition holds for at least one automaton $\mathcal{A}_M$.

Correctness: Now we show the correctness of the above construction. If $M_0 {\downarrow} \cap INF \neq \emptyset$ then there exists a marking $M \in \mathbb{N}^k$ with $M \leq M_0$ and $M \in INF$. Thus there exists an infinite M-computation π. By Dickson's lemma, there are markings M', M'' and a sequence of transitions Seq such that $M \xrightarrow{*} M' \longrightarrow_{Seq} M''$ and $M' \leq M''$. Thus the total effect of Seq is non-negative.

Now, from Lemma 3.29, we know that there is a ω-marking $M_{\mathcal{C}}$ in the coverability graph such that $M'' \leq M_{\mathcal{C}}$. Due to monotonicity of the transition relation, there is a path labeled with transitions in Seq and which leads us from $M_{\mathcal{C}}$ to a ω-marking larger than $M_{\mathcal{C}}$. Repeating this process from the larger node will finally lead us to a node which is largest of all ω-markings larger than $M_{\mathcal{C}}$. We will reach such a node $M_{\mathcal{C}}^{max}$, since the graph is finite.

This means that we can fire transitions in Seq from $M_{\mathcal{C}}^{max}$ and we get back to $M_{\mathcal{C}}^{max}$ itself (since there are no ω-marking larger than $M_{\mathcal{C}}^{max}$ in $\mathcal{C}$ and by monotonicity Seq leads to a larger or equal node in $\mathcal{C}$). So, $M_{\mathcal{C}}^{max} \longrightarrow_{Seq} M_{\mathcal{C}}^{max}$, i.e., there are ω-markings $M_1, \ldots, M_n$ such that $M_{\mathcal{C}}^{max} \longrightarrow M_1 \longrightarrow \ldots \longrightarrow M_n = M_{\mathcal{C}}^{max}$ with effect-vectors $\vec{\zeta_1}, \ldots, \vec{\zeta_n}$ such that $\sum_{1 \leq i \leq n} \vec{\zeta_i} \geq \vec{0}$. This is the condition checked in our construction.

To prove the other direction, suppose that there is a word $w \in L(\mathcal{A}_{M_{\mathcal{C}}})$ for some ω-marking $M_{\mathcal{C}}$ in the coverability graph such that $\sum_{1 \leq i \leq l} |w|_{\Lambda_i} \vec{\zeta_i} \geq \vec{0}$. This means that there is a ω-marking $M_{\mathcal{C}}$ from which there is a path (through a sequence Seq of transitions) back to itself with non-negative effect. From Lemma 3.29 we know that there are markings M' reachable from M_0 which agree with $M_{\mathcal{C}}$ in its finite coordinates, and can be made arbitrarily large in the coordinates where $M_{\mathcal{C}}$ is ω. We can choose one such marking M' such that it contains enough tokens in those coordinates where $M_{\mathcal{C}}$ is ω to be able to perform one iteration of Seq. Now, Seq has a non-negative effect. This means that one can repeatedly execute Seq starting from M'. The reachability of such an M' from M_0 and a non-negative loop from M' implies the existence of an infinite M_0-computation. This means that $M_0 \downarrow \cap INF \neq \emptyset$. $\square$
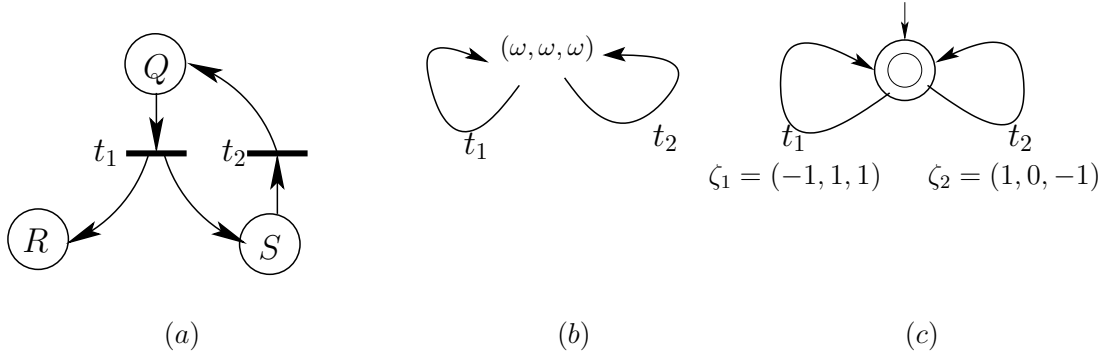

Figure 7: (a). A small Petri net, (b). Coverability graph for this net from (ω, ω, ω). (c) Automaton $A_{(\omega,\omega,\omega)}$.

Example 3.32. Consider the Petri net in Figure 7(a) and the coverability graph (Figure 7(b)) of the above Petri net from a ω-marking $M = (\omega, \omega, \omega)$ [2] where $M(Q) = M(R) = M(S) = \omega$. We show that $M \downarrow \cap INF \neq \emptyset$. The automaton produced for the single node in the coverability graph is shown in Figure 7(c). Notice that $\Lambda_1 = t_1$ and $\Lambda_2 = t_2$. Also, the effect-vectors $\vec{\zeta_1}$ and $\vec{\zeta_2}$ show the effect of firing t_1 and t_2 respectively. Notice that $L(A_{(\omega,\omega,\omega)}) = \{w| \ w \in \{t_1, t_2\}^*\}$. This means that $\varphi(L(A_{(\omega,\omega,\omega)})) = L((0,0); (1,0),(0,1))$. Finally, we compute a Presburger formula $\rho(x_1, x_2)$ for the above linear set and from it, construct the formula $\rho(x_1, x_2) \wedge x_1 \vec{\zeta_1} + x_2 \vec{\zeta_2} \geq \vec{0}$. One of the solutions of this formula is given by $x_1 = x_2 = k$ for any natural number k. This means $M \downarrow \cap INF \neq \emptyset$.

---

[2]Markings of a Petri net are written as multisets over places and vectors over the set of natural numbers interchangeably.
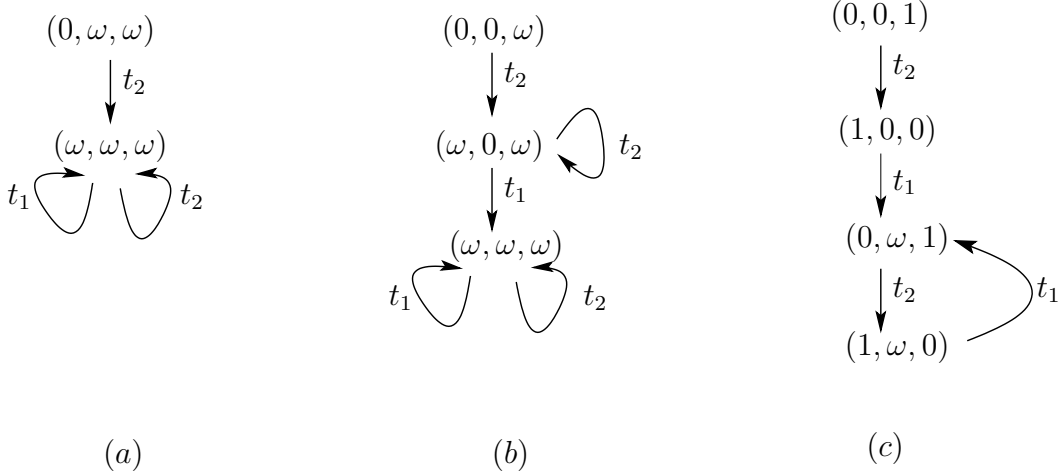
$(0, \omega, \omega)$

$\downarrow t_2$

(ω, ω, ω)

$t_1 \quad \quad t_2$

(a)

$(0, 0, \omega)$

$\downarrow t_2$

$(\omega, 0, \omega) \quad t_2$

$\downarrow t_1$

(ω, ω, ω)

$t_1 \quad \quad t_2$

(b)

$(0, 0, 1)$

$\downarrow t_2$

$(1, 0, 0)$

$\downarrow t_1$

$(0, \omega, 1)$

$\downarrow t_2 \quad t_1$

$(1, \omega, 0)$

(c)

Figure 8: (a). Coverability graph from $(0, \omega, \omega)$. (b) Coverability graph from $(0, 0, \omega)$. (c) Coverability graph from $(0, 0, 1)$.

Example 3.33. In the above, we show an example for computing $pred_{INF}(M)$ for an ω-marking M. Now we show how to compute INF_{min} for the same Petri net using Valk and Jantzen's algorithm. We start with a single marking (ω, ω, ω). Immediately, we get out of Loop 1, since $pred_{INF}((\omega, \omega, \omega))$ is true (as shown in Example 3.32). In Loop 2, one finds a minimal element in INF_{min}. This is done by first reducing the first coordinate for Q in (ω, ω, ω) to 0. In Figure 8(a), we show the coverability graph from $(0, \omega, \omega)$. $pred_{INF}((0, \omega, \omega))$ is true, since we reach a node (ω, ω, ω) in the coverability graph from $(0, \omega, \omega)$ and $pred_{INF}((\omega, \omega, \omega))$ is already shown to be true in the previous example. Then we replace the ω in place R to 0 and compute the coverability graph for $(0, 0, \omega)$ in Figure 8(b). $pred_{INF}((0, 0, \omega))$ is true again by the same reasoning. Notice that $pred_{INF}((0, 0, 0))$ is false. So, finally we show the coverability graph from marking $(0, 0, 1)$ in Figure 8(c) and $pred_{INF}((0, 0, 1))$ is true. Thus $(0, 0, 1)$ is included in INF_{min}.

In Stage 2, we have $W_0' = \{(\omega, \omega, 0)\}$ and $W_1 = min((\omega, \omega, \omega), (\omega, \omega, 0)) = \{(\omega, \omega, 0)\}$.

Now we go to Loop 1 again. From Figure 9(a), it is evident that $pred_{INF}((\omega, \omega, 0))$ is true. Now, we again perform Loop 2. We find that $pred_{INF}((0, \omega, 0))$ is false, but $pred_{INF}((1, \omega, 0))$ is true (the coverability graph from $(1, \omega, 0)$ is shown in Figure 9(b)). We show the coverability from $(1, 0, 0)$ in Figure 9(c) and it follows that $pred_{INF}((1, 0, 0))$ is true. Thus $(1, 0, 0)$ is another member of INF_{min}.

In Stage 2, we have $W_1' = (0, \omega, \omega)$ and $W_2 = min((0, \omega, \omega), (\omega, \omega, 0)) = (0, \omega, 0)$. Now $pred_{INF}((0, \omega, 0))$ is false and $W_2 = \emptyset$ and the construction terminates. Thus $INF_{min} = \{(0, 0, 1), (1, 0, 0)\}$.

3.4.5. *Computing INF_{min} for SD-TNs.*

To compute INF_{min} for SD-TNs, we will use Valk and Jantzen's Theorem 3.27 again. This algorithm requires a decision procedure for the predicate $M_0 \downarrow \cap INF \neq \emptyset$ for any given ω-marking $M_0 \in \mathbb{N}_\omega^k$ for an SD-TN. First we construct a coverability graph for a given SD-TN. We need the following definitions and notational conventions.
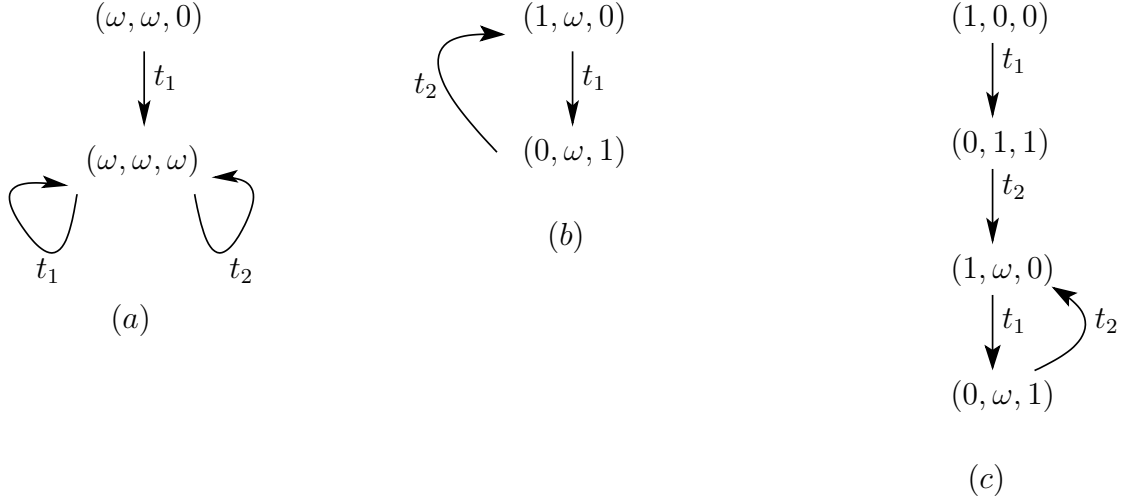
$(\omega, \omega, 0)$

$\downarrow t_1$

(ω, ω, ω)

$t_1 \quad\quad t_2$

(a)

$(1, \omega, 0)$

t_2

$\downarrow t_1$

$(0, \omega, 1)$

(b)

$(1, 0, 0)$

$\downarrow t_1$

$(0, 1, 1)$

$\downarrow t_2$

$(1, \omega, 0)$

$\downarrow t_1 \quad \big) t_2$

$(0, \omega, 1)$

(c)

Figure 9: (a). Coverability graph from $(\omega, \omega, 0)$ (b). Coverability graph from $(1, \omega, 0)$. (c) Coverability graph from $(1, 0, 0)$.

Definition 3.34. By Def. 3.1 of SD-TN, the source places and target places of transfers are disjoint and thus after a simultaneous transfer all source places are empty. We call a marking an 'after transfer marking' (AT-marking) if it is reached just after firing *Trans*. We represent markings as vectors in $\mathbb{N}^k$ of the form (transfer source places, other places). So AT-markings have the form $(\overrightarrow{0}, \overrightarrow{v})$ with $\overrightarrow{0} \in \mathbb{N}^{k'}$ and $\overrightarrow{v} \in \mathbb{N}^{k''}$ with $k = k' + k''$ where k' is the number of transfer source places. The corresponding markings in the coverability graph $\mathcal{C}$ are called ω-AT-markings and have the form $(\overrightarrow{0}, \overrightarrow{v})$ with $\overrightarrow{v} \in \mathbb{N}_\omega^{k''}$.

First we show that the coverability graph for SD-TN can be effectively constructed (Lemma 3.35), then we prove that this graph satisfies the required properties (Lemma 3.36) and finally we give an example.

Lemma 3.35. *For any SD-TN N with initial marking M_0, the coverability graph can be effectively constructed.*

Proof. We use ω-markings from $\mathbb{N}_\omega^k$ (where k is the number of places). One proceeds from M_0 similarly as in the Karp-Miller construction ([KM69]; see also Def. 3.28) except for the transfer arc. The detection of loops is done slightly differently in the two cases (with and without the transfer arc).

(1) *Loop without transfer arc:* If one encounters the case $M_1 \longrightarrow_{Seq} M_2$ with
  - $M_1 < M_2$,
  - *Seq* is a sequence of transitions of N such that the transfer arc was not used in *Seq*,
  then we replace M_2 by $M_2 + \omega(M_2 - M_1)$ as in the case of Petri nets. Notice that $\omega M = M'$ such that $M'(p) = \omega$ for all place p with $M(p) > 0$. Obviously, *Seq* can be repeated arbitrarily often to yield an arbitrarily high number of tokens on the places where M_2 is strictly larger than M_1.

(2) *Loop containing transfer arc:* Let M_1 and M_2 be two markings *reached just after transfers*, i.e., $\longrightarrow_{Trans} M_1 \longrightarrow_{Seq} M_1' \longrightarrow_{Trans} M_2$ (where *Seq* may contain other transfers). We call such markings ω-AT-markings (AT for 'after transfer'). If $M_1 < M_2$ then we

replace M_2 by $M_2 + \omega(M_2 - M_1)$. The sequence of transitions $\longrightarrow_{Seq} \longrightarrow_{Trans}$ can be repeated arbitrarily often to yield arbitrarily high numbers of tokens on the places where M_2 is strictly bigger than M_1. This is possible, because in SD-TN the set of places which are sources of transfers and the set of places which are targets of transfers are disjoint by Def. 3.1. Thus the transfers in $\longrightarrow_{Seq} \longrightarrow_{Trans}$ do not negatively affect those places p where $M_1(p) < M_2(p)$. This point does not carry over to general transfer nets. In particular, all transfer-target places, once marked by ω in this construction, will stay ω in the future. Furthermore, all transfer source places are empty after the transfer, since all transfers are simultaneous.

(3) If one reaches an ω-marking encountered before, then one creates a loop.

It is easy to show that the so-generated coverability graph is finite. Assume the contrary, i.e., that there is an infinite sequence $M_0, M_1, \ldots$ of *different* nodes in the coverability graph. Now, there are two cases.

- In this infinite sequence, there is only a finite number of occurrences of the transfer transition *Trans*. Suppose M_r was the last marking produced by transfer transition. Consider the sequence $M_{r+1}, M_{r+2}, \ldots$. This sequence is still infinite. By Dickson's lemma (Lemma 3.18), any such infinite sequence of markings of the SD-TN contains an infinite non-decreasing subsequence. Since, by our assumption above, all markings M_i are different, this subsequence must be strictly increasing. Thus, in our construction above, it would happen infinitely often that a place is marked by ω which previously had only held a finite number. However, since the infinite suffix $M_{r+1}, M_{r+2}, \ldots$ does not contain any transfer, all places marked ω stay at ω. This yields a contradiction, since there are only finitely many places in the net.

- There is an infinite number of markings produced by the transfer transition *Trans*, which appear in the sequence $M_0, M_1, \ldots$. We take the subsequence $M'_0, M'_1, \ldots$ of $M_0, M_1, \ldots$ such that each marking M'_i for $i \geq 0$ is a marking produced by the transfer transition (i.e., an ω-AT-marking). Since there are infinitely many transfer transitions in the sequence $M_0, M_1, \ldots$, the sequence $M'_0, M'_1, \ldots$ is also infinite. Now, like the previous case, we will always find a strictly increasing subsequence of $M'_0, M'_1, \ldots$. Thus, by the construction above, we would infinitely often introduce the number ω into some places of the net. However, this could only happen to places which are *not* sources of transfers, since all source-places of transfers are marked zero in ω-AT-markings. Since those places marked by ω are not sources of any transfers, they will always remain marked ω. (Here we require the specific property from SD-TN. This does not hold for general transfer nets, where a target place of one transfer could be the source place of another.) This yields a contradiction, because there are only finitely many places in the net and ω could not be introduced infinitely often as required above.

Since our assumption above led to a contradiction in both cases, the opposite must be true, i.e., the generated coverability graph is finite. $\square$

Remark: Notice that if a place p is a source of a transfer transition, then $M_1(p) < M_2(p)$ does not in general imply that p may eventually contain an arbitrarily high number of tokens. This is due to the fact that the loop may contain a transfer transition which will remove all tokens from p.

Lemma 3.36.

(1) *For every reachable marking M from the initial marking M_0 in an SD-TN, there is an ω-marking $M_{\mathcal{C}}$ in the coverability graph such that $M \leq M_{\mathcal{C}}$.*
(2) *For every ω-marking $M_{\mathcal{C}}$ in $\mathcal{C}$, there are markings M reachable from M_0 which contain arbitrarily large numbers of tokens in the places with ω in $M_{\mathcal{C}}$.*
(3) *The arcs in the coverability graph are induced by the transitions in the SD-TN. If some sequence of transitions if possible to fire from a marking $M_{\mathcal{C}}$ in the coverability graph, leading to a marking $M'_{\mathcal{C}}$, then there is a reachable marking $M \leq M_{\mathcal{C}}$ in the SD-TN which can fire the same sequence of transitions, leading to marking $M' \leq M'_{\mathcal{C}}$.*

Proof. The proof is similar to the correctness proof of the Karp-Miller algorithm for ordinary Petri nets [KM69].

(1) First, for every computation path staring at M_0 in the SD-TN there is a corresponding path in the coverability graph constructed in Lemma 3.35. Furthermore, markings are only replaced by larger ω-markings in the coverability graph. By the monotonicity of SD-TN, the first result follows.
(2) By the construction of the coverability graph for SD-TN in Lemma 3.35, values ω can be introduced in two ways: by encountering an increasing loop without transfer arcs or an increasing loop with transfer arcs.

 In the first case, the loop can simply be repeated arbitrarily often to yield arbitrarily high numbers of tokens on the increasing places (marked by ω in the coverability graph), because of the monotonicity of the net, just as for ordinary Petri nets.

 In the second case, new ω are only introduced for increasing loops between ω-AT-markings, i.e., loops of the form $\longrightarrow_{Trans} (\vec{0}, \vec{v}) \longrightarrow_{Seq} M_1 \longrightarrow_{Trans} (\vec{0}, \vec{v}')$ where $\vec{v}' > \vec{v}$. Since the source places of transfers are all marked 0 in these markings, no ωs are introduced to them here. (However, source places of transfers may aquire ω (either permanently or just temporarily until the next transfer) by ordinary Petri nets loops in the first case described above.) By the special restrictions on transfers in SD-TN (unlike in general transfer nets) the places marked by vectors $\vec{v}, \vec{v}'$ which may aquire ω here are never the source of *any* transfer. Thus the loop $\longrightarrow_{Seq} \longrightarrow_{Trans}$ can be repeated arbitrarily often to yield markings with arbitrarily high numbers of tokens on those places where $\vec{v}'$ is strictly larger than $\vec{v}$.
(3) The third property follows directly from the definition of the coverability graph. $\qquad \square$

Remark 3.37. It follows directly from Lemma 3.35 and Lemma 3.36 that place-boundedness is decidable for simultaneous-disjoint transfer nets, while it is undecidable for general transfer nets [DJS99, May03].

Example 3.38. Consider a small SD-TN shown in Figure 10(a). In Figure 10(b), we show the coverability graph $\mathcal{C}$ from a marking $M = (2, 0, 0)$ of SD-TN where $M(p_1) = 2, M(p_2) = 0$ and $M(p_3) = 0$. We omit the transfer arcs in the coverability graph if the source place of transfer does not contain a token. Notice that $Trans = (\emptyset, \emptyset, (p_1, p_3))$ and $(0, 0, 2)$ and $(0, \omega, \omega)$ are the only ω-AT-markings in $\mathcal{C}$.

3.4.6. *Computing $pred_{INF}$ for SD-TNs.*

 Now that we can compute the coverability graph for SD-TN, we continue to develop the algorithm for deciding the predicate $pred_{INF}$, i.e., deciding if $M_0 \downarrow \cap INF \neq \emptyset$ for any given ω-marking $M_0 \in \mathbb{N}_\omega^k$ for an SD-TN.

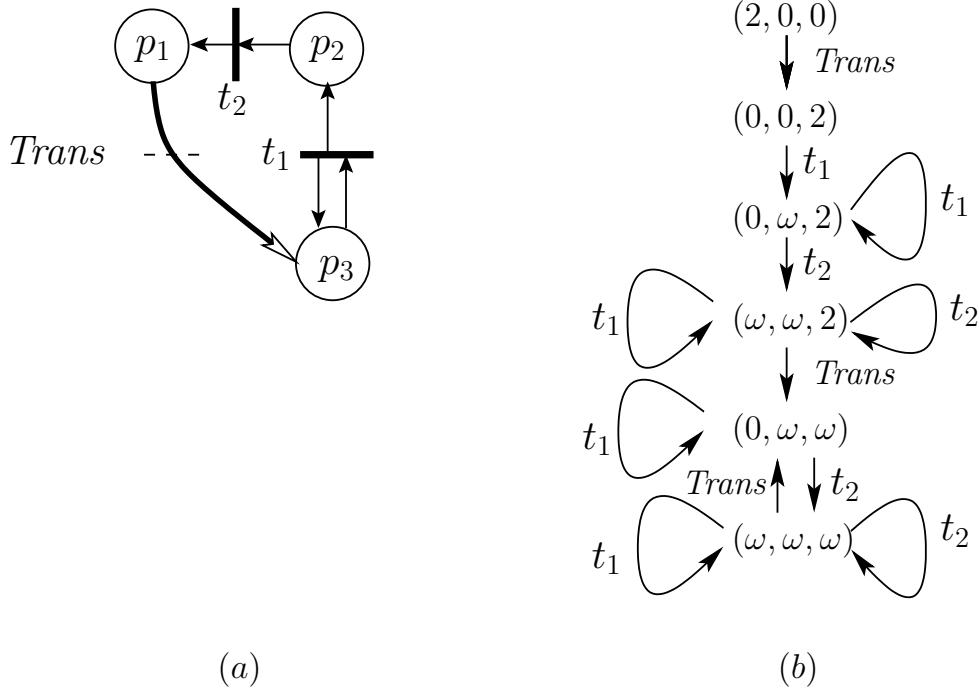(a) (b)

Figure 10: (a) A small SD-TN. (b) Coverability graph $\mathcal{C}$ for this net.

Lemma 3.39. *Given an SD-TN N with k places and an ω-marking $M_0 \in \mathbb{N}_\omega^k$, it is decidable if $M_0 \!\downarrow \cap\, INF \neq \emptyset$.*

Proof. First we give an algorithm to detect the non-emptiness of the intersection $M_0\!\downarrow \cap INF$. Let $\mathcal{C} = (G, \rightarrow)$ with $G \subseteq \mathbb{N}_\omega^k$ be the coverability graph of N from initial marking M_0. An infinite computation π from a marking M in $M_0\!\downarrow$ is detected as follows. There are two cases. Either there are finitely many or infinitely many transfers in such an infinite computation.

- In the first case, the transfer transition *Trans* is used only finitely often and π has an infinite suffix π' which starts at some marking M' and only normal Petri net transitions are used in π'. Since $M \xrightarrow{*} M'$, there is a node $M_\mathcal{C}$ in $\mathcal{C}$ such that $M' \leq M_\mathcal{C}$. To find out whether there is a positive effect of such cycles consisting of ordinary Petri net transitions, we let N' be the ordinary Petri net obtained from N by removing the transfer transition *Trans*. So π' is an infinite M'-computation of N'. Let $INF_{N'} \subseteq \mathbb{N}^k$ be the (upward-closed) set of markings from which infinite computations of N' start. So we have $M_\mathcal{C}\!\downarrow \cap INF_{N'} \neq \emptyset$. In fact, we consider each ω-marking $M_\mathcal{C} \in G$ and detect the presence of an infinite computation with just ordinary Petri net transitions if the following condition (Cond1) holds.

$$(\text{Cond1}) \qquad \exists M_\mathcal{C} \in G.\, M_\mathcal{C}\!\downarrow \cap INF_{N'} \neq \emptyset$$

This is a problem about ordinary Petri nets and it has already been shown to be decidable (Lemma 3.31). Deciding (Cond1) requires only finitely many calls to the decision procedure in Lemma 3.31, because G is finite.

- In the second case, the transfer transition *Trans* is used infinitely often in π. Recall that in Lemma 3.31, we construct automata from the coverability graph, for each of its nodes

and associate an effect-vector with each edge of such an automaton. In this case, the presence of transfer transitions in the cycles of SD-TNs does not let us follow such a procedure directly. This is due to the fact that the effect of the transfer depends on the amount of tokens in the source places of the transfer and that is not a constant number.

In this case, first we compute the effect-vectors between two ω-AT-markings $\mathcal{M}, \mathcal{M}'$ in the coverability graph such that $\mathcal{M}'$ is reachable from $\mathcal{M}$. For any pair of ω-AT-markings $\mathcal{M}, \mathcal{M}' \in G$ we can effectively construct a semilinear set $\textit{Effect}(\mathcal{M}, \mathcal{M}') \subseteq \mathbb{Z}^k$ which represents all possible effects of sequence of transitions of the form $\textit{Seq}.\textit{Trans}$ with $\mathcal{M} \longrightarrow_{Seq} \longrightarrow_{Trans} \mathcal{M}'$ where $\textit{Seq}$ is a sequence of transitions which does not contain $\textit{Trans}$. This is done as follows. First, we compute the semilinear sets $\textit{Effect}'(\mathcal{M}, X) \subseteq \mathbb{Z}^k$ for all $X \in G$ such that $X \longrightarrow_{Trans} \mathcal{M}'$ in the coverability graph $\mathcal{C}$ and $\mathcal{M} \xrightarrow{*} X$ without using $\textit{Trans}$. The sets $\textit{Effect}'(\mathcal{M}, X)$ are semilinear and effectively constructible, by computability of Presburger-arithmetic and its equivalence with semilinear languages (Theorem 3.26). This is due to the fact that $\mathcal{C}$ is a finite graph whose arcs are labelled with constant vectors in $\mathbb{Z}^k$ and the Parikh-image of regular languages is effectively semilinear. This means that one can consider $\mathcal{M}$ as the initial- and X as the final state of a finite automaton $\mathcal{A}$. Each edge in $\mathcal{A}$ is labelled by a unique symbol Λ and there is an associated effect-vector ζ for the effect of the transition by that edge. Let $\rho(x_1, \ldots, x_l)$ be the Presburger formula for the Parikh-image of $L(\mathcal{A})$ where l is the number of edges in the coverability graph. A valuation of the variable x_i for $i : 1 \le i \le l$ gives how many times the symbol Λ_i appears in a word in $L(\mathcal{A})$. Given k as the number of places in SD-TN, we have $\textit{Effect}'(\mathcal{M}, X)$ given by a Presburger formula

$$\rho_X(y_1, \ldots, y_k) = \exists x_1 \ldots, x_l. \, \rho(x_1, \ldots, x_l) \wedge \bigwedge_{1 \le i \le k} y_i = \sum_{1 \le j \le l} x_j \zeta_j(i)$$

Secondly, we obtain $\textit{Effect}(\mathcal{M}, \mathcal{M}')$ as a Presburger formula by introducing the effect of transfers ($\textit{Trans} = (I, O, ST)$) as follows. Consider the set $\mathbf{X}$ containing ω-markings X such that $\mathcal{M} \xrightarrow{*} X \longrightarrow_{Trans} \mathcal{M}'$. For each $X \in \mathbf{X}$, we compute a Presburger formula

$$\rho''_X(z_1, \ldots, z_k) = \exists y_1, \ldots, y_k. \, (\rho_X(y_1, \ldots, y_k) \wedge \rho'_X(y_1, \ldots, y_k, z_1, \ldots, z_k))$$

where $\rho'_X(y_1, \ldots, y_k, z_1, \ldots, z_k)$ is a conjunction of the following formulae.
- $\forall j, j' : (p_{j'}, p_j) \in ST. \, z_j = y_j + y_{j'} \ \wedge \ z_{j'} = 0$. Here, ST is from Def. 3.1. This corresponds to a transfer from place $p_{j'}$ to place p_j whenever $(p_{j'}, p_j) \in ST$.
- $\forall p_j \in I. \, z_j = y_j - 1 \ \wedge \ \forall p_j \in O. \, z_j = y_j + 1$. This corresponds to Petri net part of transfers, since I contains places from which there is an input arc to the transfer transition and O contains places from which there is an output arc to the transfer transitions.
- $\forall j.(p_j \notin ST \wedge p_j \notin I \cup O) \Rightarrow z_j = y_j$. Here $p_j \notin ST$ is used to mean that there are no pairs $(p, q) \in ST$, such that $p_j = p$ or $p_j = q$. This means that there is no change in the number of tokens at the other places.

Finally the effect $\textit{Effect}(\mathcal{M}, \mathcal{M}') = \bigvee_{X \in \mathbf{X}} \rho''_X(z_1, \ldots, z_k)$. By Theorem 3.25, we can compute a semilinear set from the Presburger formula given above for $\textit{Effect}(\mathcal{M}, \mathcal{M}')$.

Now we construct a new finite graph $\mathcal{C}' = (G', \rightarrow)$ as follows. $G' \subseteq G$ is the set of ω-AT-markings in G. For $\mathcal{M}, \mathcal{M}' \in G'$ we have $\mathcal{M} \rightarrow \mathcal{M}'$ in $\mathcal{C}'$ iff $\mathcal{M} \longrightarrow_{Seq''} \longrightarrow_{Trans} \mathcal{M}'$ in $\mathcal{C}$ where $\textit{Seq}''$ does not contain $\textit{Trans}$. The arc between $\mathcal{M}$ and $\mathcal{M}'$ is labeled with (a symbolic Presburger-arithmetic representation of the semilinear set) $\textit{Effect}(\mathcal{M}, \mathcal{M}')$.

We check the following condition (Cond2).

$$(\text{Cond2}) \quad \exists n \in \mathbb{N}. \, \mathcal{M}_0, \dots, \mathcal{M}_n \in G'. \, \mathcal{M}_0 \to \mathcal{M}_1 \to \cdots \to \mathcal{M}_n = \mathcal{M}_0.$$

$$\exists \vec{v_i} \in \textit{Effect}(\mathcal{M}_i, \mathcal{M}_{i+1}). \sum_{i=0}^{n-1} \vec{v_i} \geq \vec{0}$$

Note that the $\mathcal{M}_i$ above do not need to be disjoint.

Now we show how to check the condition (Cond2). We transform the graph $\mathcal{C}'$, whose arcs are labeled with semilinear sets $\textit{Effect}(\mathcal{M}, \mathcal{M}')$ into a new equivalent graph $\mathcal{C}''$ whose arcs are labeled with constant vectors. Since $\textit{Effect}(\mathcal{M}, \mathcal{M}')$ is effectively semilinear, it can be represented as a finite union of linear sets of the form $L(\vec{u_i}; \vec{w_i^1}, \dots, \vec{w_i^{n_i}})$ where $i : 1 \leq i \leq m$ and $m \geq 1$. $\mathcal{C}''$ contains the nodes of $\mathcal{C}'$ and some additional nodes:
- if there is an edge between two nodes $\mathcal{M}, \mathcal{M}'$ labeled by $\textit{Effect}(\mathcal{M}, \mathcal{M}')$ (of the above form) in $\mathcal{C}'$, we add new nodes $\mathcal{M}'_i$ for $i : 1 \leq i \leq m$ in $\mathcal{C}''$.

Also, for any pair of nodes $\mathcal{M}, \mathcal{M}'$ in C', labeled by $\bigcup_{1 \leq i \leq m} L(\vec{u_i}; \vec{w_i^1}, \dots, \vec{w_i^{n_i}})$, we have the following arcs in $\mathcal{C}''$. For each $i : 1 \leq i \leq m$, we have
- an edge from $\mathcal{M}$ to $\mathcal{M}'_i$, labeled by $\vec{u_i}$.
- edges from $\mathcal{M}'_i$ to $\mathcal{M}'_i$, labeled by $\vec{w_i^j}$ for $j : 1 \leq j \leq n_i$.
- an edge from $\mathcal{M}'_i$ to $\mathcal{M}'$, labeled by $\vec{0}$.

Let $\mathcal{C}'' = (G'', \to)$ be the graph obtained in this way. We get immediately that the following condition (Cond3) holds for C'' iff (Cond2) holds for C'.

$$(\text{Cond3}) \quad \exists n \in \mathbb{N}. \, \mathcal{M}_0, \dots, \mathcal{M}_n \in G''.$$

$$(\mathcal{M}_0 \xrightarrow{v_0} \mathcal{M}_1 \xrightarrow{v_1} \dots \xrightarrow{v_{n-1}} \mathcal{M}_n = \mathcal{M}_0) \; \wedge \; \sum_{i=0}^{n-1} \vec{v_i} \geq \vec{0}$$

The condition (Cond3) is decidable, since C'' is a finite graph and by Parikh's theorem [Par66] the Parikh-image of regular languages is effectively semilinear. (Just interpret C'' as a finite automaton and try out any $\mathcal{M}_0 \in G''$ as initial and final state.) Then we proceed as in Lemma 3.31. Thus (Cond2) is decidable.

Example 3.40. In Figure 11(a) we show $\mathcal{C}'$ obtained from $\mathcal{C}$ of Figure 10(b) with edges labeled by their Presburger-arithmetic representation. We have $\textit{Effect}((0, 0, 2), (0, \omega, \omega)) = \{(0, 1, 0) + k_1(0, 1, 0) + k_2(0, -1, 1) \mid k_1, k_2 \in \mathbb{N}\}$ and $\textit{Effect}((0, \omega, \omega), (0, \omega, \omega)) = \{(0, -1, 1) + k_1(0, 1, 0) + k_2(0, -1, 1) \mid k_1, k_2 \in \mathbb{N}\}$. (Note that the transfer moves all tokens from the first component to the third component.) In Figure 11(b), finally we show the graph $\mathcal{C}''$ obtained from $\mathcal{C}'$ in Figure 11(a).

Correctness of the above constructions: Now we show the correctness of the above two constructions (by using Lemma 3.36).
- Firstly, we show that (Cond1) is sufficient and necessary for the existence of an infinite M-computation π with finitely many transfers for some $M \leq M_0$.

Suppose there is an infinite M-computation π with finitely many transfers. Then π has an infinite suffix π', starting at some marking M' which uses only ordinary Petri net transitions. Since N' is obtained by removing transfer transitions, π' is an infinite M'-computation of N'. This implies that Cond1 holds for N' (Lemma 3.31). Since the coverability graph for N' is a subgraph of that for N, Cond1 also holds for N. On the
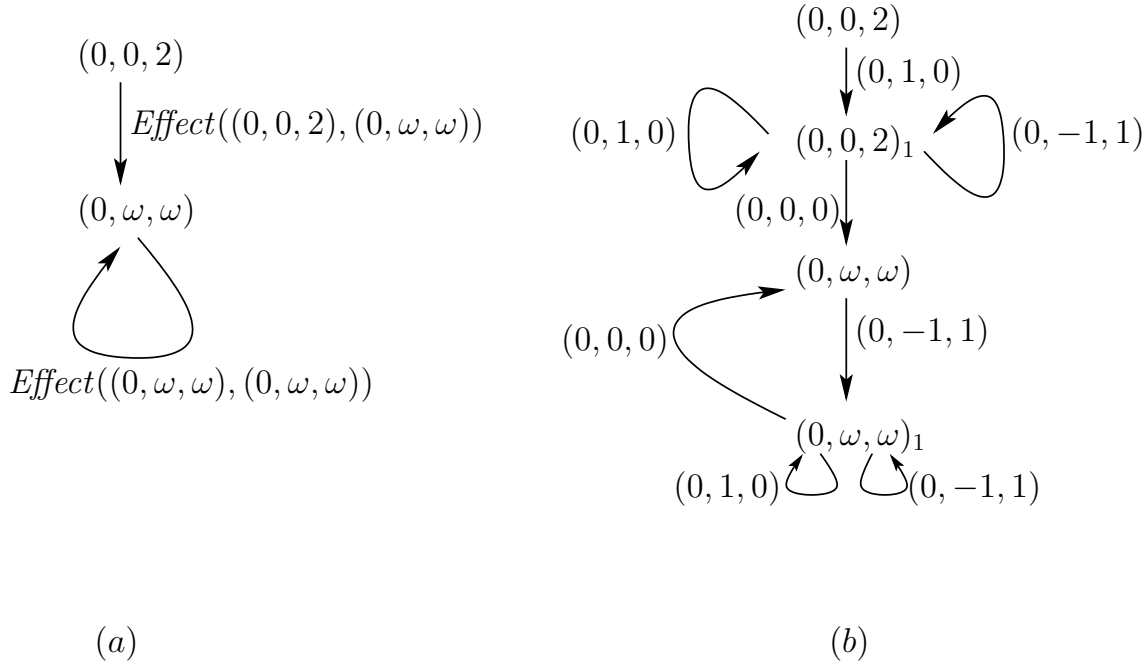
$$(a) \qquad\qquad\qquad\qquad (b)$$

Figure 11: (a). Graph $\mathcal{C}'$ derived from $\mathcal{C}$ in Figure 10(b). (b) Graph $\mathcal{C}''$ derived from $\mathcal{C}'$.

other hand, from Lemma 3.31, we have that if Cond1 holds for N', then there is an infinite M'-computation. Since $M \xrightarrow{*} M'$, we have an infinite M-computation in N.

- Secondly, we show that (Cond2) is sufficient and necessary for the existence of an infinite M-computation with infinitely many transfers for some $M \leq M_0$.

  If Cond2 is satisfied (i.e., there is a sequence Seq of transitions with non-negative effect), then there exist markings $M \leq M_0$ where $M_0 \in \mathcal{C}$ and $M' \leq M_0$ such that $M \xrightarrow{*} M'$ (by definition of $\mathcal{C}, \mathcal{C}', \mathcal{C}''$ and Lemma 3.36) such that M' is large enough to perform Seq once from M'. Now, Seq has a non-negative effect, therefore one can keep on repeating Seq resulting into an infinite M'-computation. This implies that there is an infinite M-computation.

  Now we show the other direction. Assume that there is some $M \in \mathbb{N}^k$ with $M \leq M_0$ and $M \in INF$ and some infinite M-computation π which uses $Trans$ infinitely often. Thus it contains infinitely many AT-markings. Thus, by Dickson's Lemma (Lemma 3.18, [Dic13]), there is a computation (possibly containing several transfers) where $M \xrightarrow{*} (\overrightarrow{0}, \overrightarrow{x_1}) \longrightarrow_{Seq} (\overrightarrow{0}, \overrightarrow{x_2})$ with $\overrightarrow{x_2} \geq \overrightarrow{x_1}$. Thus the total effect of the sequence Seq is non-negative. From Lemma 3.36, it follows that there exists an ω-AT-marking $\mathcal{M}_0 \in G$ with $\mathcal{M}_0 \geq (\overrightarrow{0}, \overrightarrow{x_2})$. In fact there exists a largest such $\mathcal{M}_0$ (as in case of Petri nets, see Lemma 3.31) such that we have $\mathcal{M}_0 \longrightarrow_{Seq} \mathcal{M}_0$ in C. So, $Effect(\mathcal{M}_0, \mathcal{M}_0) \geq \overrightarrow{0}$. The sequence Seq can be decomposed into $Seq = Seq_1 Seq_2 \ldots Seq_n$ with $\mathcal{M}_i \longrightarrow_{Seq_i} \mathcal{M}_{i+1}$ for $1 \leq i \leq n-1$ and $\mathcal{M}_n = \mathcal{M}_0$. Here $\{\mathcal{M}_0, \ldots, \mathcal{M}_n\}$ is the set of ω-AT markings visited in Seq. In other words, each Seq_i contains the transfer transition only once at the end. It follows that $\mathcal{M}_0 \to \mathcal{M}_1 \cdots \to \mathcal{M}_n = \mathcal{M}_0$ is a cyclic path in C' and $\overrightarrow{v_i} \in Effect(\mathcal{M}_i, \mathcal{M}_{i+1})$ and $\sum_{i=0}^{n-1} \overrightarrow{v_i} = Effect(\mathcal{M}_0, \mathcal{M}_n) \geq \overrightarrow{0}$. Therefore the condition (Cond2) is satisfied.

Altogether we obtain that $M_0 \downarrow \cap INF \neq \emptyset$ iff (Cond1) or (Cond2) is satisfied. (It is possible that both (Cond1) and (Cond2) are true.) Since both conditions are decidable, we obtain decidability of $M_0 \downarrow \cap INF \neq \emptyset$. $\qquad\square$

Lemma 3.41. *For any SD-TN N' the set INF'_{min} can be effectively constructed.*

Proof. Since *INF* is upward-closed, we can, by Lemma 3.39 and Theorem 3.27, construct the minimal elements of the set *INF*, i.e., the set INF_{min}. We obtain INF'_{min} by the restriction of INF_{min} to standard markings. $\qquad\square$

3.5. **Characterizing ZENO.**

Theorem 3.42. *Let N be a TPN. The set ZENO is effectively constructible as a MRUC.*

Proof. We first construct the SD-TN N' corresponding to N, according to Section 3.1.1. Then we consider the MRUC Z from Def. 3.15.

We have $ZENO = [\![Z]\!]$ by Lemma 3.16 and Lemma 3.17. The MRUC Z is effectively constructible by Lemma 3.41, Definition 3.15, Lemma 2.12 and Lemma 2.10. $\qquad\square$

4. The Zenoness-Problem for Discrete-timed Petri Nets

In this section, we discuss how to characterize the set *ZENO* for discrete-timed Petri nets, thus solving the open problem from [dFERA00]. First we describe how the semantics of a discrete-timed Petri net is different from that of a dense-timed Petri net.

- Firstly, the ages of the token are natural numbers rather than real numbers.
- Secondly, the timed transition takes only discrete steps.

A direct solution for discrete-timed nets is to simply modify the construction of the SD-TN N' in Section 3.1.1 by removing the time-passing phase in Subsubsection 3.1.4. The resulting net N' is then a normal Petri net, since it does not contain a transfer arc. This modified construction would yield *ZENO* for the discrete-time case, because (unlike in the dense-time case) every infinite zeno-computation in a discrete-time net has an infinite suffix taking no time at all.

In the special case where all time intervals on transitions are bounded (i.e., ∞ does not appear) there is another solution. Here one can encode discrete-timed nets into dense-timed nets, as shown in Figure 12. The trick is to split the intervals on the input (output) arcs to several point intervals on a number of transitions.

5. Arbitrarily Fast Computations

If $M_0 \in ZENO$ then, by definition, there exists an infinite M_0-computation that requires only finite time, i.e., $\exists m, \pi. \Delta(\pi) \leq m$. It follows that for any smaller number m' with $0 < m' \leq m$ there exists some marking M' with $M_0 \xrightarrow{*} M'$ and an infinite suffix π' of π s.t. π' is an infinite M'-computation with $\Delta(\pi') \leq m'$. Thus, there exist more and more markings with faster and faster computations. Formally,

$$\forall \epsilon > 0. \exists M_\epsilon \in Post^*(M_0), \text{ an infinite } \pi_\epsilon. \ M_\epsilon \xrightarrow{\pi_\epsilon} \wedge \Delta(\pi_\epsilon) \leq \epsilon \qquad (5.1)$$

However, this does *not* imply that there exists some fixed reachable marking M where arbitrarily fast computations start, because each M_ϵ could be different. The existence of
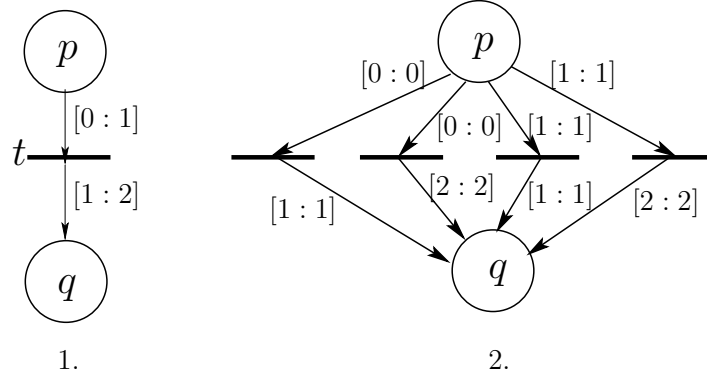
Figure 12: Simulating (1) t in TPN by (2) a set consisting of 4 transitions in 2.

arbitrarily fast computations from a fixed reachable marking is a stronger condition than zenoness, defined as follows.

$$\exists M \in Post^*(M_0). \forall \epsilon > 0. \exists \text{ an infinite } \pi_\epsilon. \ M \xrightarrow{\pi_\epsilon} \wedge \Delta(\pi_\epsilon) \leq \epsilon \qquad (5.2)$$

In general, condition (5.1) does *not* imply condition (5.2), as will be shown by Lemma 5.1.

ALL-ZENONESS-PROBLEM
Instance: A timed Petri net N, and a marking M of N.
Question: For all $\epsilon > 0$ does there exist an infinite M-computation π_ϵ s.t. $\Delta(\pi_\epsilon) \leq \epsilon$?

A marking M is called an *allzeno-marking* of N iff the answer to the above problem is 'yes'.
    We consider a timed Petri net N. We let *ALLZENO* denote the set of the allzeno-markings of N.

Lemma 5.1. *For all TPN we have* $Pre^*(ALLZENO) \subseteq ZENO$. *There exist TPN (e.g., the TPN in Figure 13) where the inclusion is strict.*

Proof. The inclusion $ALLZENO \subseteq ZENO$ follows directly from the definitions (let, e.g., $\epsilon := 1$). Since Pre^* is monotonous, we get $Pre^*(ALLZENO) \subseteq Pre^*(ZENO) = ZENO$.
    Now we consider the example TPN in Figure 13 with initial marking

$$M_0 := [(X, 1), (A, 1), (Y, 0.9)]$$

There is a zeno run π from M_0 of the following form: Transitions t_1 and t_2 alternate and the length of the delays between them drops exponentially.
    Formally, $\pi = (\rightarrow_{t_1} \rightarrow_{\delta_i} \rightarrow_{t_2} \rightarrow_{\delta_{i+1}})_{i=0,2,4,\ldots}$ with $\delta_i = (0.1) * 2^{-i}$ and thus $\Delta(\pi) \leq 0.2$. Therefore $M_0 \in ZENO$.
    Now we show that $M_0 \notin Pre^*(ALLZENO)$.
    In every reachable marking $M \in Post^*(M_0)$ there is one token on place X, one token on place Y and either one token on place A or one token on place B. Without restriction we consider the case where there is a token on place A; the other case is symmetric. So we have $M = [(X, \chi), (A, \alpha), (Y, \psi)]$. If $\chi > 1$, $\alpha > 1$ or $\psi > 1$ or $\chi \neq \alpha$ then there is no infinite run at all. Otherwise, if $\chi < 1$ then for $\epsilon := (1 - \chi)/2 > 0$ there is no run π_ϵ from M with $\Delta(\pi_\epsilon) \leq \epsilon$, and thus $M \notin ALLZENO$. There remains the case where $\chi = \alpha = 1$. Then
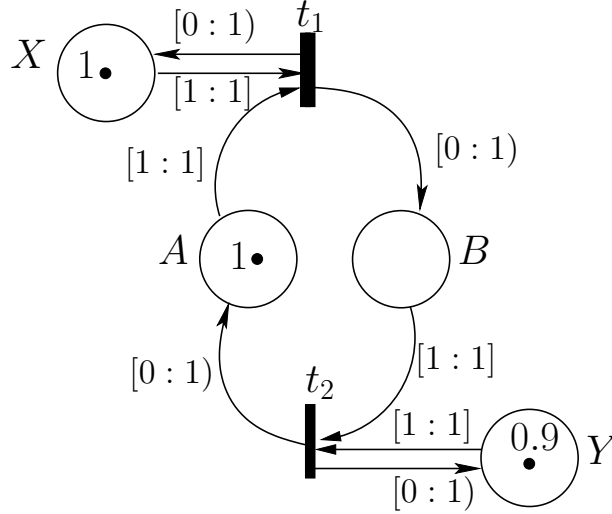
Figure 13: A TPN with initial marking $M_0 := [(X,1),(A,1),(Y,0.9)] \in ZENO$. No reachable marking is in $ALLZENO$, but allzeno markings exist, e.g., $[(X,1),(Y,1),(A,1),(B,1)]$. Note the half-open intervals $[0:1)$ which do not include 1.

transition t_1 must fire immediately, because otherwise the tokens become too old (i.e., > 1) and there is no infinite run. Let the resulting marking be $M' = [(X,\chi'),(Y,\psi),(B,\beta)]$. By construction of the net, we have $\beta < 1$. If $\psi \neq \beta$ then there is no infinite run. So we must have $\psi = \beta < 1$. Then, for $\epsilon := (1-\psi)/2 > 0$ there is no infinite run π_ϵ from either M' or M with $\Delta(\pi_\epsilon) \leq \epsilon$. Thus $M \notin ALLZENO$. So we have shown that no reachable $M \in Post^*(M_0)$ is in $ALLZENO$, i.e., $Post^*(M_0) \cap ALLZENO = \emptyset$. Therefore, $M_0 \notin Pre^*(ALLZENO)$. $\square$

Now we show that the All-Zenoness-Problem for TPN is decidable. In fact, the set $ALLZENO$ is effectively constructible as a MRUC.

Intuition: The construction of $ALLZENO$ is similar to the construction of $ZENO$ in Section 3. The main differences can be understood with the following observations.

- In arbitrarily fast runs (unlike in zeno-runs) no tokens of the *initial* marking can reach the next higher integer age by aging. For example, a token of age $1 - \epsilon$ for $\epsilon > 0$ cannot reach age 1 in a run π with $\Delta(\pi) \leq \epsilon/2$. On the other hand, tokens which are newly created during the run can reach the next higher integer age by aging, since their ages may be chosen (nondeterministically) arbitrarily close to the next higher integer. This is because all the bounds of the time intervals on transition arcs in the TPN are integers.

- If it were not for the initial marking, we would have the following situation: If there is a run π with $\Delta(\pi) = \epsilon$ where $0 < \epsilon < 1$ then there also exists a run π' with $\Delta(\pi') = \epsilon/2$. One just replaces any delay of length δ in π by a shorter delay $\delta/2$ in π' and any token of age x which is newly created in π is replaced in π' by a newly created token (on the same place) of age $x + (\lceil x \rceil - x)/2$. Furthermore, a token with an integer age i will always have a non-integer age $i + \delta$ after some delay δ for any $0 < \delta < 1$, i.e., regardless of how small δ is.

- How to treat the tokens of the initial marking? Since none of them can age to the next higher integer in arbitrarily fast computations, they cannot be encoded as $p(k-)$ tokens in the corresponding SD-TN. Instead they are all encoded as $p(k)$ tokens (if they have an integer age) or as $p(k+)$ tokens (if they have a non-integer age).
- Finally, there is the problem that arbitrarily fast computations can be either disc-computations or time-computations, depending on whether their first transition is discrete or timed. In the construction of the set *ZENO* this was elegantly solved, because this construction included the Pre^* operation which is taken w.r.t. *all* transitions (both discrete and timed). However, since of construction of *ALLZENO* does not include Pre^*, this difference must be addressed explicitly here.
- Given this, one can encode arbitrarily fast computations of TPN into computations of SD-TN, in a similar way as for zeno-computations (with delay < 1) in Section 3.

Construction of *ALLZENO***:** Given a TPN N, we first construct a SD-TN N' in the same way as in Subsection 3.1. Then we define a mapping *int* from markings of N to markings of N', similarly as in Definition 3.8.

Definition 5.2. We define a function $int : (P \times \mathbb{R}^{\geq 0})^{\odot} \to (P' \to \mathbb{N})$ that maps a marking M of N to its corresponding marking M' in N'. $M' := int(M)$ is defined as follows.

$$
\begin{array}{llll}
M'(p(k)) & := & M((p,k)) & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max. \\
M'(p(k+)) & := & \sum_{k<x<k+1} M((p,x)) & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max-1. \\
M'(p(max+)) & := & \sum_{max<x} M((p,x)) & \\
M'(p((k+1)-)) & := & 0 & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max-1. \\
M'(p_{disc}) & := & 1 & \\
M'(p_{time1}) & := & 0 & \\
M'(p_{time2}) & := & 0 & \\
M'(p_{count}) & := & 0 &
\end{array}
$$

Note that $M' = int(M)$ is a standard marking according to Def. 3.6, and M' does not contain any $p(k-)$ tokens.

Next we define an operation τ which encodes the effect of passing an arbitrarily small, but non-zero, amount of time. No tokens can age to the next higher integer age in arbitrarily short time, but all tokens of an integer age k will have an age $> k$ afterwards. Given a standard marking $M \in \Omega'$ (recall Def. 3.6) of the SD-TN N', we define $M' := \tau(M)$ as follows.

$$
\begin{array}{llll}
M'(p(k)) & := & 0 & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max. \\
M'(p(k+)) & := & M(p(k+)) + M(p(k)) & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max-1. \\
M'(p(max+)) & := & M(p(max+)) + M(p(max)) & \\
M'(p((k+1)-)) & := & M(p((k+1)-)) & \text{for } k \in \mathbb{N},\ 0 \leq k \leq max-1. \\
M'(p_{disc}) & := & M(p_{disc}) & \\
M'(p_{time1}) & := & M(p_{time1}) & \\
M'(p_{time2}) & := & M(p_{time2}) & \\
M'(p_{count}) & := & M(p_{count}) &
\end{array}
$$

Note that the operation τ is only defined on standard markings and its result is also a standard marking.

Unlike in Section 3, there is a more direct correspondence between the computations of a marking M and the computations of $int(M)$ and $\tau(int(M))$. (Recall the Def. 3.6 of INF'.)

Lemma 5.3. *Consider a TPN N with marking M_0 and the corresponding SD-TN N'.*
$M_0 \in ALLZENO \implies (int(M_0) \in INF' \ \vee \ \tau(int(M_0)) \in INF')$.

Proof. Let $M_0 \in ALLZENO$. Then there exist arbitrarily fast computations from M_0. It follows that there are either arbitrarily fast disc-computations from M_0, or arbitrarily fast time-computations from M_0 (or both). Let

$$D := \{(\lceil x \rceil - x) \mid \exists p.\, M_0((p, x)) > 0 \ \wedge \ (\lceil x \rceil - x) > 0\}$$

(1) First we consider the case that there are arbitrarily fast disc-computations from M_0. There are two cases.
  (a) If $D = \emptyset$ then all tokens in M_0 have integer ages. It follows that $int(M_0)$ does not contain any $p(k+)$ or $p(k-)$ tokens. We let $\delta := 1/2$ and obtain $int_\delta(M_0) = int_{1/2}(M_0) = int(M_0)$. By our assumption there are arbitrarily fast disc-computations from M_0 and thus there exists an infinite M_0-disc-computation π with $\Delta(\pi) < 1/2 = 1 - \delta$. Therefore, by Lemma 3.9, $int(M_0) = int_\delta(M_0) \in INF'$.
  (b) If $D \neq \emptyset$ then we define $\epsilon > 0$ as the minimal non-zero distance of the age of any token in M_0 from the next higher integer.

$$\epsilon := \min(D) > 0$$

  Let $\delta := 1 - \epsilon/2$. Then $int_\delta(M_0) = int(M_0)$. By our assumption there are arbitrarily fast disc-computations from M_0 and thus there exists an infinite M_0-disc-computation π with $\Delta(\pi) \leq \epsilon/3 < 1 - \delta$. Therefore, by Lemma 3.9, $int(M_0) = int_\delta(M_0) \in INF'$.

(2) Now we consider the case that there are arbitrarily fast time-computations from M_0. Again there are two cases.
  (a) Assume $D = \emptyset$, i.e., all tokens in M_0 have integer ages. Since there are arbitrarily fast time-computations from M_0, there exists a marking M_1 such that $M_0 \to_\lambda M_1$ with $0 < \lambda < 1/3$ and an infinite disc-computation π from M_1 with $\Delta(\pi) < 1/3$. It follows that $\tau(int(M_0)) = int(M_1)$. We let $\delta := 1/2$ and obtain $int_\delta(M_1) = int_{1/2}(M_1) = int(M_1) = \tau(int(M_0))$. Since π is an infinite M_1-disc-computation with $\Delta(\pi) < 1/3 < 1/2 = 1 - \delta$, Lemma 3.9 yields $int_\delta(M_1) \in INF'$. Therefore $\tau(int(M_0)) = int_\delta(M_1) \in INF'$.
  (b) Now assume $D \neq \emptyset$. As before, we define $\epsilon := \min(D) > 0$ and $\delta := 1 - \epsilon/2$. Since there are arbitrarily fast time-computations from M_0, there exists a marking M_1 such that $M_0 \to_\lambda M_1$ with $0 < \lambda < \epsilon/3$ and an infinite disc-computation π from M_1 with $\Delta(\pi) < \epsilon/3$. It follows that $\tau(int(M_0)) = int(M_1)$, because $\lambda < \epsilon$. Furthermore, $int_\delta(M_1) = int(M_1)$, because $\lambda < \epsilon/3 < \epsilon/2 = 1 - \delta$. Thus $\tau(int(M_0)) = int_\delta(M_1)$. Since π is an infinite M_1-disc-computation with $\Delta(\pi) < \epsilon/3 < \epsilon/2 = 1 - \delta$, Lemma 3.9 yields $int_\delta(M_1) \in INF'$. Therefore $\tau(int(M_0)) = int_\delta(M_1) \in INF'$. $\qquad\square$

Lemma 5.4. *Consider a TPN N with marking M_0 and the corresponding SD-TN N'.*
$int(M_0) \in INF' \implies M_0 \in ALLZENO$.

Proof. Let $M' := int(M_0) \in INF'$. Then, by Lemma 3.14, we have

$$\exists w_- \in perm(M'^-). \forall w_+ \in perm(M'^+). [\![Reg(M', w_+, w_-)]\!]^\uparrow \subseteq \bigcup_{\delta > 0} ZENO^{1-\delta}$$

From the definition of the function *int* we know that M'^- is empty and thus $w_- = \epsilon$, i.e., the empty sequence. Thus, $\forall w_+ \in perm(M'^+). [\![Reg(M', w_+, \epsilon)]\!]^\uparrow \subseteq \bigcup_{\delta > 0} ZENO^{1-\delta}$, and therefore $M_0 \in \bigcup_{\delta > 0} ZENO^{1-\delta}$. It follows that there exists some fixed $\delta > 0$ such that $M_0 \in ZENO^{1-\delta}$. Let $\epsilon := 1 - \delta < 1$. Then there exists some M_0-computation π_ϵ s.t. $\Delta(\pi_\epsilon) \leq \epsilon < 1$. This M_0-computation π_ϵ in N corresponds to an M'-computation in N'. Therefore, in π_ϵ, no original tokens in M_0 reach the next higher integer age by aging, because $M' := int(M_0)$, i.e., because there are no $p(k-)$ tokens in M'.

We now show that there exist arbitrarily fast M_0-computations $\pi_{\epsilon/n}$ with $\Delta(\pi_{\epsilon/n}) \leq \epsilon/n$ for any $n \geq 1$. For any $n \geq 1$ we obtain $\pi_{\epsilon/n}$ by modifying π_ϵ as follows. Every timed transition $\rightarrow_{\delta_i}$ in π_ϵ is replaced by a timed transition $\rightarrow_{\delta_i/n}$ in $\pi_{\epsilon/n}$. In order to ensure that in $\pi_{\epsilon/n}$ the same tokens do (or don't) reach/exceed the next higher integer age during the same timed transition as in π_ϵ, we modify the ages of the newly created tokens. Any token of age x which is newly created in π_ϵ is replaced in $\pi_{\epsilon/n}$ by a newly created token (on the same place) of age $x + (n-1)(\lceil x \rceil - x)/n$. This is possible, because all bounds of the time intervals on transition arcs in the TPN are integers. Since no *original* tokens in M_0 age to the next higher integer age in those runs, this suffices to make $\pi_{\epsilon/n}$ a feasible run from M_0. So we obtain that $\pi_{\epsilon/n}$ is a M_0-computation and $\Delta(\pi_{\epsilon/n}) = \Delta(\pi_\epsilon)/n \leq \epsilon/n$. Therefore, $M_0 \in ALLZENO$. □

Lemma 5.5. *Consider a TPN N with marking M_0 and the corresponding SD-TN N'.* $M_0 \in ALLZENO \iff (int(M_0) \in INF' \vee \tau(int(M_0)) \in INF')$.

Proof. The "$\Rightarrow$" implication holds by Lemma 5.3. For the "$\Leftarrow$" implication there are two cases.

(1) $int(M_0) \in INF'$. Then $M_0 \in ALLZENO$ by Lemma 5.4.
(2) $\tau(int(M_0)) \in INF'$. Let

$$D := \{(\lceil x \rceil - x) \mid \exists p. M_0((p, x)) > 0 \ \wedge \ (\lceil x \rceil - x) > 0\}$$

If $D \neq \emptyset$ then let $\epsilon := \min(D)/2 > 0$ else let $\epsilon := 1/2$. Let $\epsilon_i := \epsilon/i$ for $i \geq 1$. Let M_i be the marking that is reached from M_0 after ϵ_i time passes, i.e., $M_0 \longrightarrow_{\epsilon_i} M_i$. Since $\epsilon_i < \min(D)$ (or $\epsilon_i < 1$ if $D = \emptyset$), we have $int(M_i) = \tau(int(M_0))$ and thus $int(M_i) \in INF'$ for all $i \geq 1$. It follows from Lemma 5.4 that $M_i \in ALLZENO$. Therefore there exist arbitrarily fast time-computations from M_0 and thus $M_0 \in ALLZENO$. □

Similarly as in Section 3, we compute the set $ALLZENO$ as a multi-region upward closure. We compute a MRUC AZ and prove that $[\![AZ]\!] = ALLZENO$.

Definition 5.6. Let N be a TPN with corresponding SD-TN N', as in Subsection 3.1, and INF'_{min} from Def. 3.6. Let INF''_{min} be the restriction of INF'_{min} to markings without tokens on $p(k-)$ places. Let

$$INF''_{min} := \{M \in INF'_{min} \mid \forall p, k. \ M(p(k-)) = 0\}$$

and

$$\Gamma := \{M' \in \Omega' \mid M' \in INF''_{min} \vee \tau(M') \in INF''_{min}\}$$

and

$$AZ := \bigcup_{M' \in \Gamma} \ \bigcup_{w_+ \in perm(M'^+)} \{Reg(M', w_+, \epsilon)\}$$

Note that it follows from the definition of the function τ and the finiteness of INF''_{min} that Γ is finite.

Lemma 5.7. $[\![AZ]\!] = ALLZENO$.

Proof. Let $M \in [\![AZ]\!]$. Then there is an $M' \in \Gamma$ and a $w_+ \in perm(M'^+)$ such that $M \in [\![Reg(M', w_+, \epsilon)]\!]^\uparrow$. Thus there exists some marking $M'' \leq M$ s.t. $M'' \in [\![Reg(M', w_+, \epsilon)]\!]$. Therefore $int(M'') = M' \in \Gamma$. Since $INF''_{min} \subseteq INF'$, it follows that $int(M'') \in INF' \lor \tau(int(M'')) \in INF'$. By Lemma 5.5 we have $M'' \in ALLZENO$ and thus $M \in ALLZENO^\uparrow = ALLZENO$.

To prove the reverse inclusion, let $M \in ALLZENO$. Then, by Lemma 5.5, $int(M) \in INF'$ or $\tau(int(M)) \in INF'$.

- Consider the case where $int(M) \in INF'$. From the definition of the function int (Def. 5.2) it follows that $int(M)$ does not contain any tokens on $p(k-)$ places. Therefore, there exists some marking $M'' \in INF''_{min}$ s.t. $int(M) \geq M'' \in \Gamma$.
- Consider the case where $\tau(int(M)) \in INF'$. From the definition of the functions int and τ (Def. 5.2) it follows that $\tau(int(M))$ does not contain any tokens on $p(k-)$ places. Therefore, there exists some marking $M' \in INF''_{min}$ s.t. $\tau(int(M)) \geq M'$. It follows from the definition of the functions int and τ and the fact that $M' \in INF''_{min}$ that there exists some marking $M'' \leq int(M)$ s.t. $\tau(M'') = M'$. Since $M' \in INF''_{min}$, we have $M'' \in \Gamma$. Therefore there exists some marking $M'' \in \Gamma$ s.t. $int(M) \geq M''$.

Thus in both cases there is some marking $M'' \in \Gamma$ s.t. $int(M) \geq M''$.

It follows that there exists some $w_+ \in perm(M''^+)$ such that $M \in [\![Reg(M'', w_+, \epsilon)]\!]^\uparrow \subseteq [\![AZ]\!]$. $\qquad \square$

Theorem 5.8. *Let N be a TPN. The set ALLZENO is effectively constructible as a MRUC.*

Proof. We first construct the SD-TN N' corresponding to N, according to Subsection 3.1. Then we consider the MRUC AZ from Def. 5.6. We have $ALLZENO = [\![AZ]\!]$ by Lemma 5.7. The MRUC AZ is effectively constructible by Lemma 3.41, Definition 5.6, and Lemma 2.10. $\qquad \square$

Finally, we consider the problem whether, for a given marking, there exists an infinite computation which takes no time at all.

ZEROTIME-PROBLEM

Instance: A timed Petri net N, and a marking M of N.
Question: Does there exist an infinite M-computation π such that $\Delta(\pi) = 0$?

A marking M is called a *zerotime-marking* of N iff the answer to the above problem is 'yes'.

For a timed Petri net N, we let *ZEROTIME* denote the set of its zerotime-markings.

The construction of the set *ZEROTIME* as a MRUC is similar to the construction of *ALLZENO*. The differences are that in the construction of the SD-TN N' the transitions which encode the time-passing phase (i.e., Subsubsection 3.1.4) are left out. (Thus N' is a normal Petri net.) Furthermore, the function τ is not needed, since all zerotime-computations are disc-computations.

Lemma 5.9. *Consider a TPN N with marking M_0 and the corresponding Petri net N' as in Subsection 3.1 (without Subsubsection 3.1.4). Then $M_0 \in ZEROTIME \iff int(M_0) \in INF'$.*

Proof. If $M_0 \in ZEROTIME$ then it has an infinite disc-computation π with $\Delta(\pi) = 0$. Thus $int(M_0) \in INF'$ by the proof of Lemma 5.3. If $int(M_0) \in INF'$ then $M_0 \in ZEROTIME$, because there are no time-passing phases in the Petri net N'. ◻

The definition of the needed MRUC ZT is a simplification of Definition 5.6.

Definition 5.10. Let N be a TPN with corresponding Petri net N', as in Subsection 3.1 (without Subsubsection 3.1.4), and INF'_{min} from Def. 3.6. Let INF''_{min} be the restriction of INF'_{min} to markings without tokens on $p(k-)$ places. Let

$$INF''_{min} := \{M \in INF'_{min} \mid \forall p, k.\ M(p(k-)) = 0\}$$

and

$$ZT := \bigcup_{M' \in INF''_{min}} \bigcup_{w_+ \in perm(M'^+)} \{Reg(M', w_+, \epsilon)\}$$

Lemma 5.11. $[\![ZT]\!] = ZEROTIME$.

Proof. This follows directly from the definitions and Lemma 5.9, similarly as in Lemma 5.7. ◻

Theorem 5.12. *Let N be a TPN. The set $ZEROTIME$ is effectively constructible as a MRUC.*

Proof. We first construct the Petri net N' corresponding to N, according to Subsection 3.1 (without Subsubsection 3.1.4). Then we consider the MRUC ZT from Def. 5.10. We have $ZEROTIME = [\![ZT]\!]$ by Lemma 5.11. The MRUC ZT is effectively constructible by Lemma 3.41, Definition 5.10, and Lemma 2.10. ◻

6. Universal Zenoness

The zenoness problem in Section 3 can be seen as existential zenoness, i.e., the question whether there exists an infinite zeno computation, and it is decidable by Theorem 3.42.

Here we consider the *universal* zenoness problem, i.e., the question whether *all* infinite computations from a given marking are zeno (i.e., take only finite time).

Universal Zenoness Problem

Instance: A timed Petri net N and a marking M.
Question: Is it the case that for every infinite M-computation π, there exists a finite number m s.t. $\Delta(\pi) \leq m$?

We will prove the undecidability of the universal zenoness problem by a reduction from an undecidable problem for lossy counter machines [May03]. To simplify the presentation, we no not consider the universal zenoness problem directly, but its negation.

Non-Zenoness-Problem

Instance: A timed Petri net N and a marking M.
Question: Does there exist an infinite M-computation π, such that $\Delta(\pi) = \infty$?

Obviously, a Petri net N with marking M is a positive instance of the Universal Zenoness Problem if and only if it is a negative instance of the Non-Zenoness-Problem.

A marking M is called a *nonzeno-marking* of N iff the answer to the Non-Zenoness-Problem problem is 'yes'.

We consider a timed Petri net N. We let *NONZENO* denote the set of the non-zeno-markings of N. The set *NONZENO* is not the complement of the set *ZENO*. A marking of a TPN can have infinite zeno runs or infinite nonzeno runs or both or neither.

In the following, we show that the Non-Zenoness-Problem is undecidable, which implies the undecidability of the Universal Zenoness Problem. The proof is done by reducing the universal termination problem for *lossy counter machines* to the Non-Zenoness-Problem for TPN.

6.1. Lossy Counter machines.

Lossy counter machines (LCM) [May03] are Minsky-counter machines where the values in the counters can spontaneously decrease (i.e., part of the counter value is lost). Different versions of LCM are defined by the way in which this decrease can happen (e.g., just 1 lower, any lower value, or a reset to zero), which is formally expressed by so-called lossiness relations [May03]. Here we consider the classic variant of LCM where counters can spontaneously change to any lower value. In this model, any test for zero of a counter could always be successful by a spontaneous reset to zero. Thus classic LCM are equivalent to the following model.

A *lossy counter machine* is a tuple $L = (Q, q_0, C, \delta)$, where Q is a finite set of *states*, $q_0 \in Q$ is the initial state, C is a finite set of counters and δ is a finite set of *instructions*. An instruction is a triple of the form $(q, instr, q')$, where $q, q' \in Q$ and $instr$ is either an *increment* (of the form $c++$); a *decrement* (of the form $c--$); or a *reset* (of the form $c := 0$) for a counter c in C.

A *configuration* γ of L is of the form (q, Val), where $q \in Q$ and Val is a mapping from the set C of counters to the set $\mathbb{N}$ of natural numbers. We define a transition relation $\rightsquigarrow$ on the set of configurations such that $(q, Val) \rightsquigarrow (q', Val')$ iff one of the following conditions is satisfied:

(1) $(q, c++, q') \in \delta$, $Val'(c) = Val(c) + 1$ and $Val'(c') = Val(c')$ if $c' \neq c$.
(2) $(q, c--, q') \in \delta$, $Val(c) > 0$, $Val'(c) = Val(c) - 1$ and $Val'(c') = Val(c')$ if $c' \neq c$.
(3) $(q, c := 0, q') \in \delta$, $Val'(c) = 0$ and $Val'(c') = Val(c')$ if $c' \neq c$.
(4) $q' = q$, $Val'(c) = Val(c) - 1$ for some $c \in C$, and $Val'(c') = Val(c')$ if $c' \neq c$.

We use $\overset{*}{\rightsquigarrow}$ for denoting the reflexive, transitive closure of $\rightsquigarrow$. For a configuration γ, a γ-*computation* π of L is a sequence of configurations $\gamma_0, \gamma_1, \gamma_2, \ldots$, where $\gamma_0 = \gamma$ and $\gamma_i \rightsquigarrow \gamma_{i+1}$, for $i \geq 0$.

The *universal termination* problem for LCMs is defined as follows (see [May03]).

$\exists n. LCM^\omega$

Instance: A LCM L with 4 counters and a control-state q_0.
Question: Does there exist a finite number n such that there is an infinite computation of L from the configuration $\gamma_0 = (q_0, n, 0, 0, 0)$?

Theorem 6.1. [May03] $\exists n. LCM^\omega$ *is undecidable.*

6.2. **Undecidability.** We show the undecidability of the non-zenoness problem for TPNs through a reduction from $\exists n. LCM^\omega$.

Given an instance of $\exists n. LCM^\omega$, i.e., an LCM L and a state q_0 of L, we construct an equivalent instance of the non-zenoness problem, i.e., we derive a TPN N and a marking M of N, such that non-zenoness problem for TPNs has a positive answer if and only if $\exists n. LCM^\omega$ has a positive answer.

The idea is as follows. First the TPN performs a loop, taking zero time, which puts a number n of tokens on a certain place. This encodes guessing the number n. Then the TPN faithfully simulates the computation of the LCM from configuration $(q_0, n, 0, 0, 0)$ in such a way that every single step takes at least one time unit. This simulation of the LCM is the only possible infinite non-zeno run of the TPN since the initial guessing-loop takes zero time. Thus the TPN has an infinite non-zeno run iff there exists a number n s.t. the LCM has an infinite run from $(q_0, n, 0, 0, 0)$.

The following encoding of LCM into TPN is similar to the constructions in [dFERA00, AN02], except that we enforce that every simulation step takes at least one time unit. This delay is crucial for our proof.

Consider the LCM $L = (Q, q_0, C, \delta)$. We construct a corresponding timed Petri net (TPN) $N = (P, T, In, Out)$ as follows. For each state $q \in Q$ there is a place in P which we call place q. We use P_Q to denote the set of places of N corresponding to the states Q. Also, for each counter $c \in C$ there is a place in P which we call place c. We use P_C to denote to the set of places corresponding to counters. There are also six intermediate places for simulating each increment and decrement instructions and five such places for simulating each reset instruction of the LCM.

A configuration γ of L is encoded by a marking M in N when the following conditions are satisfied.

- The state of γ is defined in N by the element of P_Q which contains a token. (The TPN N satisfies the invariant that there is at most one place in P_Q which contains a token).
- The value of a counter c in γ is defined in M by the number of tokens in place c which have ages equal to 0. (Tokens which have ages greater than 0 are considered to have been lost and do not affect the value of the counter).

Losses in L are simulated either by making the age of the token strictly greater than 0, or by firing a special $loss_c$ transition which can always remove tokens from the place c in P_C. Transitions in L are encoded by functions In and Out in N reflecting the above properties and are defined as follows.

- An *increment* $\imath = (q, c++, q')$ in δ is simulated by a set of transitions in T which are of the form in Figure 14. These transitions effectively move a token from place q to place q' and adds a token of age 0 to place c. However, we let at least one time unit pass during these transitions. To achieve this, we use two intermediate places r_i^1 and r_i^2 for each increment instruction $\imath$. The transition t_i^1 is fired by moving a token from place q to place r_i^1 and resets its age to 0. The token in r_i^1 has to stay there for a time equal to 1 and then the transition t_i^2 is fired. If more time passes, then this token in r_i^1 will forever stay in place r_i^1 after which no tokens will ever reside in any place in P_Q and thus the net will deadlock. The idea is that the TPN should not have any zeno-run during the simulation of any operation of the LCM. So, during the simulation of the increment-operation, we need to wait at least for one time unit. This makes the ages of all tokens in places P_C at least equal to 1. Thus, in order to avoid resetting the values of the counters, we add,
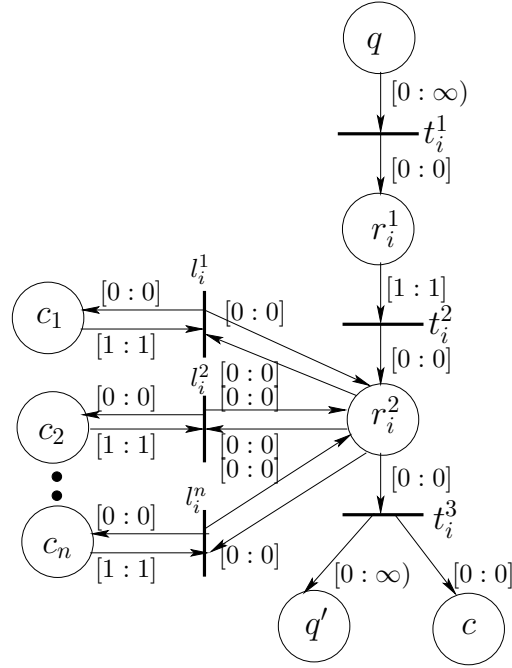
Figure 14: Simulating the operation of increasing the counter c.
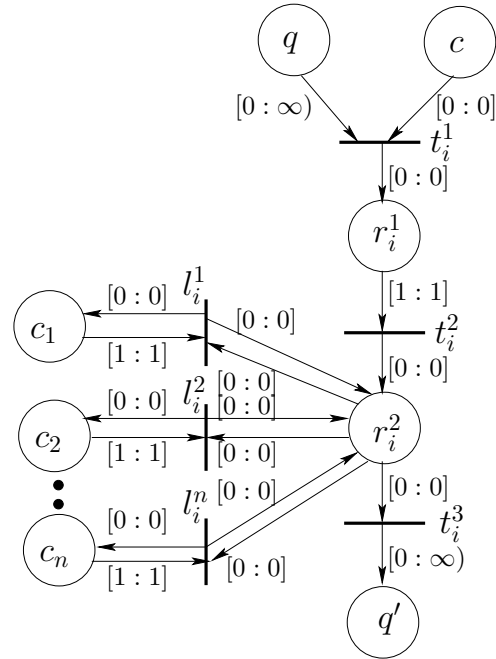
Figure 15: Simulating the operation of decreasing the counter c.

for each counter in C a new transition. In Figure 14, we assume that $P_C = \{c_1, \ldots, c_n\}$ and thus we add the transitions $\ell_i^1, \ell_i^2, \ldots, \ell_i^n$. These transitions are used to *refresh* the ages of the tokens in the places in P_C. Now, if a token in place c_1 has its age equal to 1, and thus has become too old for firing other transitions (*decrements*), it is replaced by a fresh token of age 0. Notice that the refreshment phase either does not take any time at all or it deadlocks. Finally, when the transition t_i^3 is fired, the new control state will be q' and there will be a new token of age 0 in c. The resulting marking will therefore correspond to the counter c having an increment by the value 1. The refreshing process for the counters $c_1, \ldots, c_n$ will be stopped after firing t_i^3, since the token in r_i^2 will now be removed. Notice that some tokens in $c_1, c_2, \ldots, c_n$ may be lost (i.e., may still have age greater or equal to 1), since the TPN has a lazy semantics and these tokens may not have been refreshed. Possibly losing tokens is allowed in the simulation of LCM by TPN, since the semantics of LCM allows spontaneous decreases in counters.

- A *decrement* $\imath = (q, c--, q')$ in δ is simulated by a similar set of transitions in T which are of the form in Figure 15. These transitions also move a token from place q to place q' and remove a token of age 0 from place c. Again, we let at least one time unit pass during these transitions. The description is similar to the case for the increment-operation.
- For each place c in $P_C = \{c_1, \ldots, c_n\}$, there is a transition which we call $loss_c$ (Figure 16). A transition $loss_c$ removes a token of age 0 from the counter $c \in P_C$ and thus simulates the lossiness of counter c.
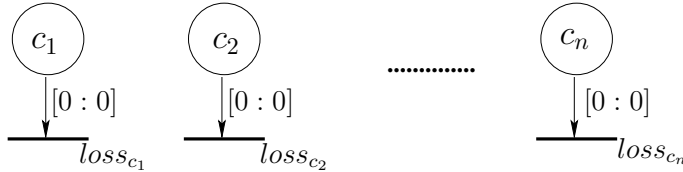

Figure 16: Simulating losses.

- The construction for the *reset* instruction $\imath = (q, c := 0, q')$ in δ is shown in Figure 17.

  The idea is that we reset the value of counter c to 0, by making the ages of all tokens in place c at least equal to 1. Observe that we simulate resetting the counter in L by resetting the counter in N. All tokens in each of the places in P_C which had age 0 have now age equal to 1. Thus, in order to avoid resetting the values of the counters other than c, we add, for each counter in $C - \{c\}$ a new transition. In Figure 17, we assume that $P_C - \{c\} = \{c_1, \ldots, c_n\}$ and thus we add the transitions $\ell_i^1, \ell_i^2, \ldots, \ell_i^n$. These transitions are used to *refresh* the ages of the tokens in the places in $P_C - \{c\}$, i.e., all counters can be refreshed expect c. Now, if a token in place c_i has its age equal to 1, and thus has become too old for firing other transitions (*decrements*), it is replaced by a fresh token of age 0. Finally, when the transition t_i^3 is fired, the new control state will be q', and each token in place c will have an age which is at least one. The resulting marking will therefore correspond to the counter c having the value 0.
- Initialization. To guess the initial value in counter c_1 of the LCM, we add an extra place q_{init} in P and add two transitions in T, shown in Figure 18. First the transition $t_{\imath_1}$ is enabled if there is a token in q_{init} with age 0. By executing this transition n times (for some $n \geq 0$) without letting any time pass, we can produce n tokens in the counter c_1. This simulates an initial value n of c_1 in LCM. Then, we switch control for simulating the
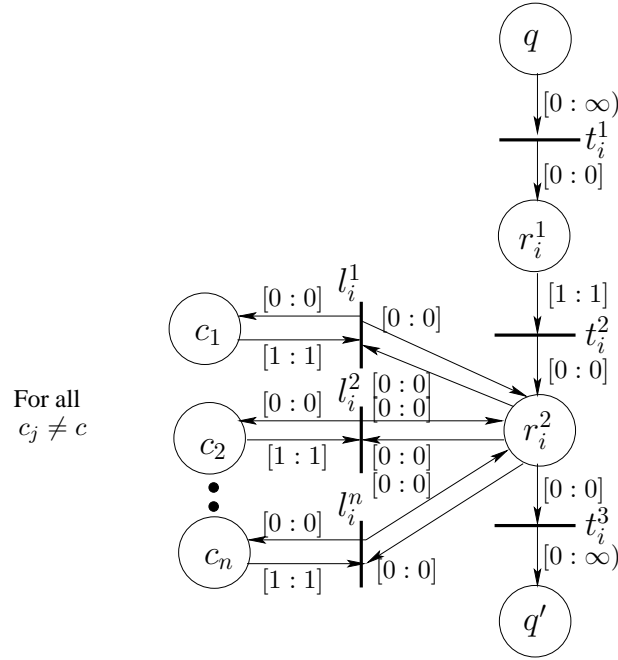
Figure 17: Simulating the operation of resetting the value of the counter c to 0. All other counters c_j with $c_j \neq c$ can be refreshed.

usual operations of the LCM by executing the transition t_{ι_2} in Figure 18, which moves the token from q_{init} to q_0.
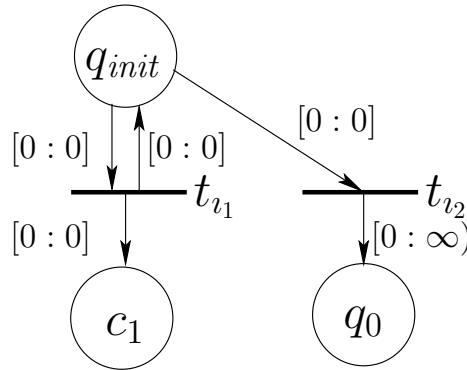
Figure 18: Initialization.

Consider a marking M of N and a configuration $\gamma = (q,\, Val)$ of L. We say that M is an *encoding* of γ if M contains a token in place q and the number of tokens with ages equal to 0 in place c is equal to $Val(c)$ for each $c \in C$. Furthermore, all other places in M are empty.

We also use the following notion of intermediate markings. A marking is called *intermediate* if it has a token in place r_ι^1 (r_ι^2) where ι is of the form $(q, c := 0, q')$ and there are no tokens in other intermediate places and in those belonging to P_Q.

We derive N from L as described above. We define M_0 to be $[(q_{init}, 0)]$.

Lemma 6.2. *N has an infinite non-zeno $M_0 = [(q_{init}, 0)]$-computation if and only if there exists an $n \geq 0$ s.t. the LCM L has an infinite $\gamma_0 = (q_0, n, 0, 0, 0)$-computation.*

Proof.

$\Leftarrow$: Let $\gamma_0 := (q_0, n, 0, 0, 0)$ and $M_0 := [(q_{init}, 0)]$. Given an infinite γ_0-computation π of L, we show that there is an infinite non-zeno M_0-computation π'.

To show this, it is enough to prove the following.

(a) Starting from a marking M_0 in TPN, there is a zero-time computation from M_0 to a marking M which is an encoding of γ_0. In fact, $M_0 \xrightarrow{n}_{t_{\iota_1}} \longrightarrow_{t_{\iota_2}} M$ (see Figure 18).

(b) After the initialization step, given two configurations γ, γ' of L such that $\gamma \rightsquigarrow \gamma'$ and a marking M which is an encoding of γ, there is a sequence in N of the form $M = M_0 \longrightarrow M_1 \longrightarrow \cdots \longrightarrow M_k = M'$ where $k \geq 1$ and the following holds.

- M' is an encoding of γ'.
- M_i is an intermediate marking for $0 < i < k$.

Since $\gamma \rightsquigarrow \gamma'$, we know that γ' is derived from γ, using one of the four possible types of transitions described for LCMs. We show the claim only for the first case, namely when γ' is derived from γ by executing an increment instruction ι. The other cases can be explained in a similar manner. Let $\gamma = (q, Val)$ and $\gamma' = (q', Val')$. Since M is an encoding of γ, it means that place q in M contains a token. From the construction described above (Figure 14) we know that from M, we can fire t_ι^1 and produce a marking $\mathcal{M}_1$ such that $M \longrightarrow_{t_\iota^1} \mathcal{M}_1$. $\mathcal{M}_1$ is obtained from M by removing the token from q and adding a token of age 0 in r_ι^1. This means that both M and $\mathcal{M}_1$ contains exactly equal number of tokens of age 0 at each place in P_C.

Next we let time pass by one time unit and obtain a marking M_1 such that $\mathcal{M}_1 \longrightarrow_1 M_1$. This means that $M \longrightarrow M_1$. Notice that all the tokens with age 0 in the places of P_c in $\mathcal{M}_1$ have transformed into tokens of age 1 in M_1. Now, firing the transition t_ι^2 from M_1 results in a marking $\mathcal{M}_2$ such that $M_1 \longrightarrow_{t_\iota^2} \mathcal{M}_2$. The transition t_ι^2 removes the token of age 1 from r_ι^1 and adds a token of age 0 in r_ι^2. Here, for each place in P_C, there are no tokens with age less than 1. Furthermore, the number of tokens of age 1 in each place $c' \in P_C$ is the same in both M_1 and $\mathcal{M}_2$. We define $M_2 = \mathcal{M}_2$. So, $M_1 \longrightarrow M_2$.

To restore the ages of the tokens of age 0 at each place in P_C in the marking M_0 (these tokens correspond to the values of the counters in γ), we start a *refreshment* phase. Suppose for a counter $c_1 \in P_C$, $Val(c_1) = x$. Then we fire the transition ℓ_ι^1 x times from M_2 and *refresh* all x tokens of age 1 in c_1 to age 0. Similarly we refresh all tokens of age 1 in the other counters in P_C. Notice that we do not let time pass between these discrete transitions.

The markings $M_1, M_2, \ldots$, etc. in the above are all intermediate markings. Now we fire the transition t_ι^3 by moving the token from r_ι^2 to q' and adding a token of age 0 to place c, yielding a marking M'. This means that for each counter $c' \in P_C \setminus \{c\}$, the number of tokens of age 0 in c' for M' is the same as that for M. Furthermore, in comparison to marking M, there is exactly one extra token of age 0 at place c in M'. This means that the new marking M' will be an encoding of γ' and $M \xrightarrow{*} M'$.

The simulation of other operations can be explained in a similar manner.

Now, if there exists some number n s.t. the LCM has an infinite computation from $(q_0, n, 0, 0, 0)$ then the TPN has an infinite non-zeno computation from an initial marking that corresponds to $(q_0, n, 0, 0, 0)$. This is ensured by the initialization step and the above

simulation of operations in LCM. The non-zenoness of the computation in TPN is ensured by passage of time during each operation of LCM. Notice that the initialization step takes zero-time.

$\Rightarrow$: Suppose that there is an infinite M_0-computation π of N taking infinite time. It follows that π must contain the transition $t_{\imath_2}$, since the initial $t_{\imath_1}$-loop takes zero time. Consider the maximal subsequence π' of π, where each marking in π is an encoding of some configuration of L. The sequence π' exists for the following reasons.

- Since π is non-zeno and infinite, the computation π is infinite even after the zeno initialization step.
- Furthermore, each operation (increment, decrement, etc) takes a finite non-zero amount of time (this follows from the constructions (see the Figures) for increment, decrement and resetting).

From the initialization step, it is straightforward that in zero time we reach a marking which is an encoding of $\gamma_0 = (q_0, n, 0, 0, 0)$ for some $n \geq 0$, i.e., the encoding of γ_0 is the configuration reached immediately after firing transition $t_{\imath_2}$ at the end of the initial guessing-phase. In the following, we show that there is an infinite γ_0-computation.

To prove this, it is enough to show that given two consecutive encodings M and M' (with only intermediate markings in between) in π' and a configuration γ which is an encoding of M, there is a configuration γ' such that $\gamma \overset{*}{\leadsto} \gamma'$. Let $\gamma = (q, \mathit{Val})$.

Since $M \overset{*}{\longrightarrow} M'$ we know that there are markings $M_0, \ldots, M_k$ such that $M = M_0 \longrightarrow M_1 \longrightarrow \cdots \longrightarrow M_k = M'$ where $k \geq 1$ and $M_1, \ldots, M_{k-1}$ are intermediate markings.

There are two cases. Either $k = 1$ or $k > 1$.

If $k = 1$, i.e., $M \longrightarrow M'$, we know that M' can be derived from M by firing a discrete transition. This means that there is a marking $\mathcal{M}$ such that $M \longrightarrow_t M'$ where the discrete transition t corresponds to Figure 16.

If $k > 1$, then M' is obtained from M by firing transitions corresponding to those in Figure 14, 15, and 17. For instance, consider that $\imath = (q, c++, q')$ is an instruction in L, for some counter c. From the construction of Figure 14, we know that the ages of some of the tokens in P_C may exceed 1, since not all tokens need to be refreshed. We can derive γ' from γ by first performing loss transitions corresponding to tokens which become too old followed by executing the instruction $(q, c++, q')$. Similarly, we can perform loss transitions followed by a decrement or a reset instruction of the LCM. $\square$

Theorem 6.3. *The Non-Zenoness-Problem for TPN is undecidable.*

Proof. Directly from Lemma 6.2 and Theorem 6.1. $\square$

Since Non-Zenoness-Problem is the negation of the Universal Zenoness Problem, this implies the following result.

Theorem 6.4. *The Universal Zenoness Problem for TPN is undecidable.*

7. Token Liveness

First, we define the *liveness* of a token in a marking.

Let M be a marking in a TPN N. A token in M is called *syntactically k-dead* if its age is $\geq k$. It is trivial to decide whether a token is k-dead from a marking.

A token is called *semantically live* from a marking M, if we can fire a sequence of transitions starting from M which eventually consumes the token. Formally, given a token (p, x) and a marking M, we say that (p, x) can be *consumed* in M if there is a transition t satisfying the following properties:

- t is enabled in M.
- $In(t, p)$ is defined and $x \in In(t, p)$.

Definition 7.1. A token (p, x) in a marking M is *semantically live* if there is a finite M-computation $\pi = M M_1 \cdots M_r$ such that the aged token $(p, x + \Delta(\pi))$ can be consumed in M_r. By $L(M)$ we denote set of of all live tokens in M.

Note that token liveness is defined here for individual tokens, not sets of tokens. There are nets and markings where two tokens (p, x) and (q, y) are both live, but where it is impossible to consume both of them.

Semantic liveness of tokens in TPN

Instance: A timed Petri net N with marking M and a token $(p, x) \in M$.
Question: Is (p, x) live, i.e., $(p, x) \in L(M)$?

We show decidability of the semantic token liveness problem by reducing it to the *coverability problem* for TPNs (which is decidable due to Lemma 2.12).

Coverability problem

Instance: A TPN N, a finite set of initial markings M_{init} of N, and an upward closed set $\mathsf{M}_{fin} \uparrow$ of markings of N, where M_{fin} is finite.
Question: $\mathsf{M}_{init} \xrightarrow{*} \mathsf{M}_{fin} \uparrow$?

Theorem 7.2. *The coverability problem is decidable for TPN* [AN01].

Suppose that we are given a TPN $N = (P, T, In, Out)$ with marking M and a token $(p, x) \in M$. We shall translate the question of whether $(p, x) \in L(M)$ into (several instances of) the coverability problem. To do that, we construct a new TPN N' by adding a new place p^* to the set P. The new place is not input or output of any transition. Either there is no transition in N which has p as its input place. Then it is trivial that $(p, x) \notin L(M)$. Otherwise, we consider all instances of the coverability problem defined on N' such that

- M_{init} contains a single marking $M - (p, x) + (p^*, x)$.
- M_{fin} is the set of markings of the form $[(p_1, x_1), \ldots, (p_n, x_n), (p^*, x')]$ such that there is a transition t and
  - the set of input places of t is given by $\{p, p_1, \ldots, p_n\}$.
  - $x' \in In(t, p)$ and $x_i \in In(t, p_i)$ for each $i : 1 \le i \le n$.

In the construction above, we replace a token (p, x) in the initial marking by a token (p^*, x); we also replace a token (p, x') in the final marking where $x' \in In(t, p)$ by a token (p^*, x'). The fact that the token in the question is not consumed in any predecessor of a marking in M_{fin}, is simulated by moving the token into the place p^* (in both the initial and final markings), since $p^* \notin P$ and not an input or output place in N'. Therefore, the token is live in M of N iff the answer to the coverability problem is 'yes'.

From Theorem 7.2, we get the following.

Theorem 7.3. *The token liveness problem is decidable.*

8. Boundedness

Given a system and an initial configuration, the boundedness problem is the question whether the size of any reachable configuration is bounded by a constant. In the context of a TPN, this is the question whether the number of tokens in any reachable marking is bounded.

Every marking M is a multiset of timed tokens. The size of a marking M is defined as the size of this multiset, denoted as $|M|$ (see Def. 2.1). In other words, $|M|$ denotes the number of timed tokens in M. Given a set of markings M, we define $maxsize(\mathsf{M}) := \max\{|M| \,|\, M \in \mathsf{M}\}$ as the maximal size of any marking in M.

In Section 2 we defined $Reach(M_0) := \{M' \,|\, M_0 \xrightarrow{*} M'\}$ as the set of markings reachable from M_0.

The boundedness problem for a TPN with an initial marking M_0 is then the question whether $maxsize(Reach(M_0))$ is bounded.

Remark 8.1. Note that, unlike for normal untimed Petri nets, the boundedness problem for TPNs is *not* equivalent to the question whether $|Reach(M_0)|$ is bounded. By the lazy semantics of our TPNs (see Section 2) time can always pass and increase the values of the clocks of the tokens. Thus (unless the initial marking is empty) one obtains infinitely many (even uncountably many) different reachable markings, even if the number of tokens stays constant. For example, consider a TPN with just one place p and no discrete transitions and initial marking $M_0 := \{(p, 0)\}$. Then $Reach(M_0) = \{\{(p, x)\} \,|\, x \in \mathbb{R}^{\geq 0}\}$ is infinite, but $maxsize(Reach(M_0)) = 1$.

In this section we consider two different variants of the boundedness problem for TPNs. In *syntactic boundedness* all tokens in a marking count towards its size, while in *semantic boundedness* only semantically live tokens (see Section 7) count.

Syntactic Boundedness of TPN

Instance: A timed Petri net N with initial marking M_0.
Question: Is $maxsize(Reach(M_0))$ bounded ?

We give an algorithm similar to the Karp-Miller algorithm [KM69] for solving the syntactic boundedness problem for TPNs. The algorithm builds a tree, where each node of the tree is labeled with a region. We build the tree successively, starting from the root, which is labeled with R_{M_0}: the unique region satisfied by M_0 (it is easy to compute this region). At each step we pick a leaf with label R and perform one of the following operations:

(1) If $Post(R)$ is empty we declare the current node *unsuccessful* and close the node.
(2) If there is a previous node on the branch which is labeled with R then declare the current node *duplicate* and close the node.
(3) If there is a predecessor of the current node labeled with $R' <^r R$ then we declare $maxsize(Reach(M_0))$ infinite (the TPN is unbounded), and terminate the procedure.
(4) Otherwise, declare the current node as an *interior* node, add a set of successors to it, each labeled with an element in $Post(R)$. This step is possible due to Lemma 2.11.

If the condition of step 3 is never satisfied during the construction of the tree, then we declare $maxsize(Reach(M_0))$ finite (the TPN is bounded).

The proof of correctness of the above algorithm is similar to that of original Karp-Miller construction [KM69]. The termination of the algorithm is guaranteed due to the fact that the ordering $\preceq$ on the set of regions is a well-quasi-ordering (follows from [Hig52]).

Theorem 8.2. *Syntactic boundedness of TPN is decidable.*

A consequence of this result is that we can solve the non-termination problem for TPNs, i.e., the problem whether a given marking M has at least one infinite run. (Remember that, by our definition of TPN computations (see Section 2), every infinite run must contain infinitely many discrete transitions.)

NON-TERMINATION OF TPN

Instance: A timed Petri net N, and a marking M of N.
Question: Does there exist an infinite M-computation?

A marking M is called a *non-terminating marking* of N iff the answer to the above problem is 'yes'. For a given timed Petri net N we let *NONTERM* denote the set of the non-terminating markings of N.

Theorem 8.3. *Non-Termination of TPN is decidable.*

Proof. By Theorem 8.2 we can decide syntactic boundedness. If the system is syntactically unbounded then it is certainly non-terminating. If the system is syntactically bounded, then all the markings in $Reach(M_0)$ can be symbolically represented by the finitely many regions computed by the algorithm above. In this case we have non-termination iff there exists a cyclic (and thus repeatable) path among these regions which contains at least one discrete transition. (Cyclic paths containing only timed transitions do not induce valid infinite runs, since we require that every infinite run contains infinitely many discrete transitions.)

This condition can easily be checked in the algorithm above as follows. If condition (3) is true on some branch then the system is non-terminating. If some branch stops with condition (2), then check if at least one step on the path from the previous node R to the duplicate node R was a discrete step. If yes, then there exists a repeatable path from R to R which contains at least one discrete transition and thus the system is non-terminating. $\square$

Since semantically dead tokens cannot influence the behavior of a TPN (see Section 7), one would like to abstract from them.

Let N be a TPN with marking M. Then we define the live part of the TPN marking M as $Reach^l(M) := \{L(M') \mid M \xrightarrow{*} M'\}$, i.e, $Reach^l(M)$ is the set of reachable markings where the semantically dead tokens have been removed.

SEMANTIC BOUNDEDNESS OF TPN

Instance: A timed Petri net N with initial marking M_0.
Question: Is $maxsize(Reach^l(M_0))$ bounded ?

Theorem 8.4. *Semantic boundedness of TPN is undecidable.*

Proof. Using slightly modified constructions of [RGdFE99] or [AN02], we can easily derive the undecidability of semantic boundedness even for dense-timed Petri nets (see [Mah05]). The idea is to use the same encoding of lossy counter machines (LCM) into TPN as in Section 5 (or a similar encoding, as shown in [Mah05]). In this encoding, the semantically live tokens (with age < 1) correspond to the counter values of the LCM while the older (semantically dead) tokens count as lost. Thus the TPN is semantically bounded iff the LCM is bounded. Since boundedness of LCM is undecidable [May03], the result follows. $\square$

9. Summary and Conclusions

9.1. **Problems and their Relation to each other.** We considered the following sets of markings of a given TPN.

- *NONTERM*, the set of markings which have an infinite run.
- *NONZENO*, the set of markings which have an infinite non-zeno run.
- *ZENO*, the set of markings which have an infinite zeno run.
- *ALLZENO*, the set of markings which have arbitrarily fast infinite runs.
- *ZEROTIME*, the set of markings which have an infinite run taking no time at all.

Note that *NONZENO* is not the complement of *ZENO*. A marking of a TPN can have infinite zeno runs or infinite non-zeno runs or both or neither. However, $NONTERM = NONZENO \cup ZENO$.

First we consider the relationships between these sets, both for dense-timed Petri nets and discrete-timed Petri nets.

For discrete-timed Petri nets, we trivially have $ALLZENO = ZEROTIME$, but for dense-timed nets $ZEROTIME \subset ALLZENO$, in general. For example, in the TPN of Figure 13 we have that the marking $[(X,1),(Y,1),(A,1),(B,1)] \in ALLZENO$, but the marking $[(X,1),(Y,1),(A,1),(B,1)] \notin ZEROTIME$.

For discrete-timed nets, every zeno-computation has an infinite suffix that takes no time at all and thus $Pre^*(ZEROTIME) = ZENO$. However, for dense-timed Petri nets, it was shown in Lemma 5.1 that there exist instances (e.g., Figure 13) where $Pre^*(ALLZENO) \subset ZENO$, i.e., a strict subset.

The inclusion $ALLZENO \subseteq Pre^*(ALLZENO)$ follows directly from the definition of Pre^*. The following example shows that there exist instances where the inclusion is strict, i.e., $ALLZENO \subset Pre^*(ALLZENO)$. (This works for both dense- and discrete time.) One constructs a TPN and marking M_0 such that at M_0 one must first wait 1 time unit before the first transition can fire. This transition then creates a marking $M_1 \in ALLZENO$. Thus $M_0 \in Pre^*(ALLZENO)$, but $M_0 \notin ALLZENO$.

Furthermore, it is trivial (for both dense- and discrete time) that $ZENO \subseteq NONTERM$ and $NONZENO \subseteq NONTERM$, and that there exist instances where these inclusions are strict. In general, the sets *ZENO* and *NONZENO* are incomparable. Finally, $ZENO \cup NONZENO = NONTERM$. The following theorem summarizes these results.

Theorem 9.1. *In general for dense-timed Petri nets*

$$ZEROTIME \subseteq ALLZENO \subseteq Pre^*(ALLZENO) \subseteq ZENO \subseteq ZENO \cup NONZENO = NONTERM$$

and for each inclusion there is an instance where it is strict.
In general for discrete-timed Petri nets

$$ZEROTIME = ALLZENO \subseteq Pre^*(ALLZENO) = ZENO \subseteq ZENO \cup NONZENO = NONTERM$$

and for each inclusion there is an instance where it is strict.

9.2. **Decidability Results.** It has been shown in this paper that the sets *ZEROTIME*, *ALLZENO*, $Pre^*(ALLZENO)$, and *ZENO* are effectively constructible as MRUC (multi-region upward closures; see Def. 2.9) for dense-timed nets and thus also for discrete-timed

nets. Furthermore, we have shown in Section 6 that *NONZENO* is undecidable for dense-timed nets. This undecidability proof carries over directly to discrete-timed nets, since all delays are of length ≥ 1.

The situation is slightly more complex for the set *NONTERM*. Theorem 8.3 showed the decidability of the non-termination problem for dense-timed nets. This decidability result trivially carries over to discrete-timed nets. Like all the other sets of markings considered here, the set *NONTERM* is closed under the relation $\equiv$ on regions (see Def. 2.3) and it is also upward-closed. Thus it is representable as a MRUC. However, this MRUC is not effectively constructible. It has been shown by Escrig et al. [dFERA00] that *NONTERM* is not effectively constructible even for discrete-timed Petri nets. Their proof is similar to the construction in Section 6 (except for the initial guessing phase). A timed Petri net can simulate a lossy counter machine (or a reset Petri net). Thus, if one could effectively construct *NONTERM*, then one could decide the universal termination problem for lossy counter machines $\exists n. LCM^{\omega}$ (see Section 6) which is known to be undecidable [May03].

The following table summarizes the results on decidability and effective constructibility of the considered sets of markings of TPN. Note that all those results coincide for discrete-timed nets and dense-timed nets. However, the proofs are harder for dense-timed nets.

Set	Decidable?	Effectively constructible?
NONTERM	Yes (Thm. 8.3)	No ([dFERA00])
NONZENO	No (Thm. 6.3)	No (Thm. 6.3)
ZENO	Yes (Thm. 3.42)	Yes (Thm. 3.42)
$Pre^*(ALLZENO)$	Yes (Thm. 5.8 and Lemma 2.12)	Yes (Thm. 5.8 and Lemma 2.12)
ALLZENO	Yes (Thm. 5.8)	Yes (Thm. 5.8)
ZEROTIME	Yes (Thm. 5.12)	Yes (Thm. 5.12)

9.3. **Conclusion and Future Work.** We have solved several open problems about the verification of *dense-timed Petri nets (TPNs)* in which each token has an age represented by a real number, where the transitions are constrained by the ages of the tokens and the firing semantics is lazy. This class is closely related to the class of parameterized systems of timed processes where each process is restricted to have a single clock [AJ03].

We have shown decidability of zenoness, existence of arbitrarily fast computations, token-liveness and syntactic boundedness for TPNs, as well as the undecidability of universal zenoness.

To solve the zenoness problem, we defined a new class of untimed Petri nets (SD-TN) which is more general than standard Petri nets, but which is a subclass of transfer nets. For these SD-TN, we gave a method to compute a characterization of the set of markings from which there are infinite computations. This is interesting in itself, since for general transfer nets such a characterization is not computable [DJS99, May03].

We have considered TPNs with just one real-valued clock per token. For all the problems studied so far, the decidability results coincide for dense-time and discrete-time (although the proofs for dense-time are harder).

However, if we consider TPNs with *two* clocks per token, there is a decidability gap between the dense-time and the discrete-time domain. The coverability problem becomes undecidable for dense-timed TPNs with only *two* clocks per token, while it remains decidable

for discrete-timed TPNs with any finite number of clocks per token [ADM04]. The class of TPNs with multiple clocks per token is related to parameterized systems of timed processes, with multiple clocks per process [ADM04]. It is therefore worth investigating whether this more general class induces a similar gap for the problems we have considered in this paper.

Acknowledgement

The authors wish to thank the anonymous referees for their detailed comments.

References

[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.

[AD94]   R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[ADM04]   P. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *Proc. LICS' 04*, pages 345–354. IEEE Computer Society Press, 2004.

[ADMN04]   P. Abdulla, J. Deneux, P. Mahata, and A. Nylén. Forward reachability analysis of timed Petri nets. In *Proc. FORMATS-FTRTFT'04*, volume 3253 of *Lecture Notes in Computer Science*, pages 343–362. Springer, 2004.

[AJ98]   P. A. Abdulla and B. Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Proc. TACAS '98, 4th 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.

[AJ03]   P. A. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*, 290(1):241–264, 2003.

[Alu91]   R. Alur. *Techniques for Automatic Verification of Real-Time Systems*. PhD thesis, Dept. of Computer Sciences, Stanford University, 1991.

[AMM04]   P. Abdulla, P. Mahata, and R. Mayr. Decidability of zenoness, syntactic boundedness and token-liveness for dense-timed petri nets. In *Proc. of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3328 of *LNCS*. Springer Verlag, 2004.

[AN01]   P. A. Abdulla and A. 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.

[AN02]   P. A. Abdulla and A. Nylén. Undecidability of LTL for timed Petri nets. In *INFINITY 2002, 4th International Workshop on Verification of Infinite-State Systems*, 2002.

[BA93]   M. Ben-Ari. *Mathematical Logic for Computer Science*. Prentice Hall, 1993.

[BCH$^+$05] B. Bérard, F. Cassez, S. Haddad, O. Roux, and D. Lime. Comparison of Different Semantics for Time Petri Nets. In *Proceedings of the Third International Symposium on Automated Technology for Verification and Analysis (ATVA'2005)*, volume 3707 of *Lecture Notes in Computer Science*, pages 81–94, Taipei, Taiwan, October 2005. Springer.

[BM99]   A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *Proc. of STACS'99*, volume 1563 of *LNCS*. Springer Verlag, 1999.

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

[dFERA00]   D. de Frutos Escrig, V. Valero Ruiz, and O. Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *ICATPN 2000*, number 1825 in LNCS, pages 187–206, 2000.

[Dic13]   L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35:413–422, 1913.

[DJS99]   C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of Reset P/T Nets. In *Proc. of ICALP'99*, volume 1644 of *LNCS*. Springer Verlag, 1999.

[FS98]   A. Finkel and P. Schnoebelen. Fundamental structures in well-structured infinite transition sytems. In *Proc. LATIN '98*, volume 1380 of *Lecture Notes in Computer Science*, pages 102–118, 1998.

[FS01]   A. Finkel and Ph. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[Gin66]   S. Ginsburg. *The Mathematical Theory of Context-free Languages*. McGraw-Hill, 1966.

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

[GS66]   S. Ginsburg and E. Spanier. Semigroups, Presburger formulas, and languages. *Pacific J. of Mathematics*, 16:285–296, 1966.

[Hei82]   B. Heinemann. Subclasses of self-modifying nets. In Girault, C. and Reisig, W., editors, *Informatik-Fachberichte 52: Application and Theory of Petri Nets: Selected Papers from the First and Second European Workshop on Application and Theory of Petri Nets, Strasbourg, Sep. 23-26, 1980, Bad Honnef, Sep. 28-30, 1981*, pages 187–192. Springer-Verlag, 1982.

[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.

[Mah05]   P. Mahata. *Model Checking Parameterized Timed Systems*. PhD thesis, Dept. of Information Technology, Uppsala University, Sweden, 2005.

[May03]   R. Mayr. Undecidable problems in unreliable computations. *TCS*, 297(1-3):337–354, 2003.

[Mur89]   T. Murata. Petri nets: Properties, analysis und applications. *Proc. of the IEEE*, 77(4):541–580, 1989.

[Par66]   R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

[Pet62]   C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.

[Pet77]   J.L. Peterson. Petri nets. *Computing Surveys*, 9(3):221–252, 1977.

[RGdFE99]   V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proc. 8th International Workshop on Petri Nets and Performance Models*, pages 188–196, 1999.

[Tri99]   S. Tripakis. Verifying progress in times systems. In *Proc. ARTS '99*, pages 299–314, 1999.

[VJ85]   R. Valk and M. Jantzen. The Residue of Vector Sets with Applications to Decidability Problems in Petri Nets. *Acta Informatica*, 21:643–674, 1985.