Properties of Distributed Timed-Arc Petri Nets

Mogens Nielsen[1], Vladimiro Sassone[2,⋆], and Jiří Srba[1,⋆⋆]

[1] **BRICS**[⋆⋆⋆], Dept. of Computer Science, University of Aarhus
[2] University of Sussex

Abstract. In [12] we started a research on a distributed-timed extension of Petri nets where time parameters are associated with tokens and arcs carry constraints that qualify the age of tokens required for enabling. This formalism enables to model e.g. hardware architectures like GALS. We give a formal definition of process semantics for our model and investigate several properties of local versus global timing: expressiveness, reachability and coverability.

Introduction

Verification of concurrent and parallel systems plays nowadays an important role in the concurrency theory, with a number of successful applications. Algorithmic methods have been developed for process algebras generating infinite state systems, timed process algebra, Petri nets, lossy vector addition systems, counter machines, real time systems and many others. In particular, the idea to equip automata with real time appeared to be very fruitful and there are even automatic verification tools for such systems as UPPAAL [9] and KRONOS [5].

The main idea behind timed automata is to equip a standard automaton with a number of synchronous clocks, and to allow transitions to be conditioned on clock values and to affect (reset) clocks. One of the objections to this formalism is the assumption of perfect synchrony between clocks. For many applications this assumption is justified, but for others it is unrealistic. Clearly, geographically highly distributed systems are prime examples, but the issue has been addressed also for hardware design; e.g. in work on so-called Globally Asynchronous Locally Synchronous (GALS) systems [11].

Following these arguments we suggested in [12] a new model called *distributed timed-arc Petri nets* (DTAPN). One of the reasons for choosing Petri net formalism is the explicit representation of locality.

Several models that take time features into account have been presented in the literature (for a survey see [4,17]). For example *timed transitions Petri nets* were proposed in [13] where transitions are annotated with their durations. A model in which time parameters are associated to places is the *timed places*

⋆ Author partly supported by MUST project *TOSCA*.
⋆⋆ Author partly supported by the GACR, grant No. 201/00/0400.
⋆⋆⋆ Basic Research in Computer Science, Centre of the Danish National Research Foundation.

© Springer-Verlag Berlin Heidelberg 2001

Petri nets of [16]. We shall analyse *timed-arc Petri nets* [3,7], a time extension of Petri nets where time (age) is associated to tokens and transitions are labelled by time intervals which restrict the age of tokens that can be used to fire them. In this model, time is considered to be *global*, i.e., all tokens grow older with the same speed. In spite of the fact that reachability is decidable for ordinary Petri nets [10], it is undecidable for global timed-arc Petri nets [15]. On the other hand, coverability is decidable for such a model [14,1], which is also known to offer 'weak' expressiveness, in the sense that it cannot simulate Turing machines [2].

In [12] a new model is suggested where time elapses in a place independently on other places, taking the view that places represent 'localities'. The idea of local clocks is generalised in the DTAPN model, where we use an equivalence relation on places to specify which pairs of places must synchronise. As special instances we get *local* timed-arc Petri nets (LT nets), where no synchronisations are forced, and *global* timed-arc Petri nets (GT nets), with full synchronisation.

In this paper we give a formal definition of process semantics which provides a reading of the differences between the LT and the GT net models and of their relative strengths. Among the motivations behind LT nets, they seem to be a weaker model than the global time one and some interesting properties could be verified algorithmically. Nevertheless, we prove that the general reachability problem for LT nets is undecidable. However, we show that a small modification of the problem (a slight restriction of the set of allowed initial markings) makes reachability decidable for LT nets, but not for GT nets. Finally, we argue that coverability is decidable for all DTAPNs.

1 Distributed Timed-Arc Petri Nets

Definition 1 (Distributed timed-arc Petri net).
A *place/transition Petri net* (PT) is a tuple (P, T, F), where P is a finite set of *places*, T is a finite set of *transitions* such that $T \cap P = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation.*

A *distributed timed-arc Petri net* (DTAPN) is a tuple $N = (P, T, F, c, E, D)$, where (P, T, F) is a Petri net and:

- $c : F|_{P\times T} \to D \times (D \cup \{\infty\})$ is a *time constraint* on transitions such that for each arc $(p, t) \in F$, if $c(p, t) = (t_1, t_2)$ then $t_1 \le t_2$,
- $E \subseteq P \times P$ is an equivalence relation on places (*synchronisation relation*),
- $D \in \{\mathbb{R}_0^+, \mathbb{N}_0\}$ is either *continuous* or *discrete* time.

Let $x \in D$ and $c(p, t) = (t_1, t_2)$. We write $x \in c(p, t)$ whenever $t_1 \le x \le t_2$. We also define ${}^\bullet x = \{y \mid (y, x) \in F\}$, $x^\bullet = \{y \mid (x, y) \in F\}$, for $x \in P \cup T$ and use $\mathcal{B}(X)$ to denote the set of all finite multisets on a set X. In what follows, we assume that ${}^\bullet t \neq \emptyset$ for every $t \in T$.

A *marked* PT net is a net (P, T, F) together with an *initial marking* $M \in \mathcal{B}(P)$. A *marking* of a DTAPN (P, T, F, c, E, D) is a function $M : P \to \mathcal{B}(D)$. A *marked* DTAPN is a pair (N, M), for M a marking of N with all tokens of age 0. Each place is thus assigned a number of tokens, and each token is annotated with a real (natural) number (*age*). Let $x \in \mathcal{B}(D)$ and $a \in D$. We define $x \lessdot\!\!+ a$ to add the value a to every element of x, i.e., $x \lessdot\!\!+ a = \{b + a \mid b \in x\}$.

The dynamics of DTAPNs is defined by two types of transition relations: *firing* of a transition and *time-elapsing*.

Definition 2 (Transition rules).
Let $N = (P, T, F, c, E, D)$ be a DTAPN, M a marking and $t \in T$.

– We say that t is *enabled* by M iff $\forall p \in {}^\bullet t.\ \exists x \in M(p).\ x \in c(p, t)$.
– If t is enabled by M, it can *fire* producing a marking M', in symbols $M[t\rangle M'$, such that:

$$\forall p \in P.\ M'(p) = \Big(M(p) \smallsetminus C^-(p,t)\Big) \cup C^+(t,p)$$

where C^- and C^+ are chosen to satisfy the following equations (which may have several solutions):

$$C^-(p,t) = \begin{cases} \{x\} \text{ such that } x \in M(p) \text{ and } x \in c(p,t) & \text{if } p \in {}^\bullet t \\ \emptyset & \text{otherwise} \end{cases}$$

$$C^+(t,p) = \begin{cases} \{0\} & \text{if } p \in t^\bullet \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that the new tokens added to places $t^\bullet$ are of initial age 0.
– We define a *time-elapsing* transition ϵ, for $\epsilon \colon P/E \to D$, as follows, where $[p]_E$ denotes the E-equivalence class of p:

$$M[\epsilon\rangle M' \quad \text{iff} \quad \forall p \in P.\ M'(p) = M(p) \lessdot\!\!+ \epsilon([p]_E).$$

We write $M \longrightarrow M'$ iff either $M[t\rangle M'$ or $M[\epsilon\rangle M'$ for some t or ϵ.

Two classes of DTAPNs play prominent roles. The first one requires an absolute synchronisation and was studied in the past, while the other one is a new model suggested in [12] — completely asynchronous:

– *Global timed-arc Petri nets (GT nets)*: $E = P \times P$,
– *Local timed-arc Petri nets (LT nets)*: $E = \Delta_P = \{(p,p) \mid p \in P\}$.

2 GALS Architectures

In high-performance VLSI the clock management is the main source of power consumption. Keeping one global clock synchronised is usually the bottleneck of a processor design. In [11] the authors suggest a method to decrease the pitfalls of global clock distribution. A processor design is partitioned into synchronous blocks that communicate globally with each other on asynchronous basis using a handshake mechanism. This architecture is called *Globally Asynchronous and*

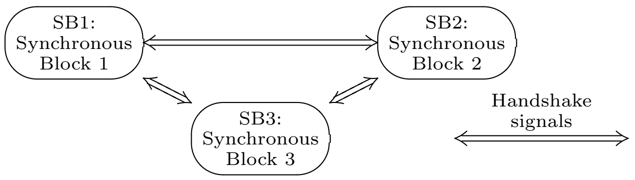

Fig. 1. GALS architecture

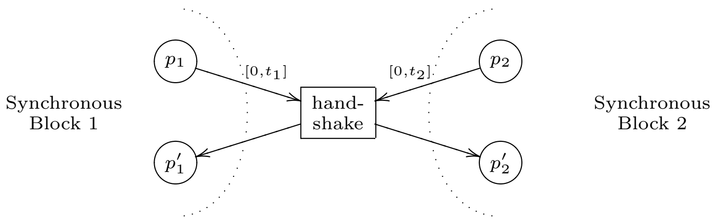

Fig. 2. Modelling of handshake mechanism between SB1 and SB2

Locally Synchronous (GALS) architecture. The authors applied this technology to a realistic design with million gates, saving about 30% of power energy with negligible overhead. Fully asynchronous solutions to the problem have been also examined, for an overview see e.g. [8].

Distributed timed-arc Petri nets, in particular LT nets in the fully asynchronous case, appear to be a good model for such architectures. Let us now focus on the design of GALS. Figure 1, from [11], represents the basic concept of GALS architecture for three synchronous components. For each of the synchronous blocks SB1, SB2 and SB3 we design a GT net, joining them together by means of a handshake communication as in Figure 2 (in which every arc from a place p to a transition t is labelled by the time interval $c(p,t)$). This creates the final DTAPN with a synchronisation relation respecting the place partitioning given by the blocks SB1, SB2 and SB3. The transition 'handshake' forces the blocks SB1 and SB2 to synchronise and after this transition is fired new tokens of age 0 appear in places p'_1 and p'_2. Then SB1 and SB2 can continue their un-synchronised performance. If we set $t_1 = t_2 = \infty$ then there are no time constrains on the maximal waiting time for the handshake communication. However, by changing the values t_1 and t_2 we may forbid a late handshake synchronisation.

Further examples of LT nets and DTAPNs, as e.g. timed producer/consumer systems or Fischer's mutual exclusion protocol, have been described in [12].

3 Process Semantics for DTAPN

We aim at providing a common ground on which to assess relative expressiveness of GT nets and LT nets. In this section, building upon the idea of PT net

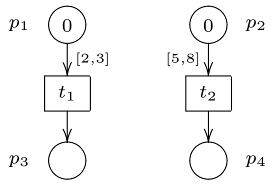

Fig. 3. Dependent transitions in a GT net and independent in an LT net

processes [6], we formalise a notion of processes of DTAPNs and establish their properties with respect to firing sequences.

The subtle differences between computations of LT and GT nets that we want to address can be illustrated with the help of the net of Figure 3. Were this net an ordinary net, transitions t_1 and t_2 would be completely independent. Things are not so neat when we consider the time constraints. If the net is a GT net, i.e. time is global, after firing t_2, the transition t_1 cannot possibly fire anymore. If instead we consider the net under the local time interpretation, t_1 and t_2 can again be considered completely independent, as the one's firing cannot affect the other's enabledness.

Definition 3 (Process Nets).
A *process net* is a PT net $\Pi = (P, T, F)$ such that Π is acyclic, i.e., for $x, y \in \Pi$, $x \prec_\Pi y$ implies $y \not\prec_\Pi x$, and Π is deterministic, i.e., for each $p \in P$, $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$, where $\prec_\Pi$ denotes the transitive closure of F.

Each place p of a process net Π has exactly zero or one transition in its preset. We define ${}^\star p = t$ if ${}^\bullet p = \{t\}$ and ${}^\star p = \perp$ if ${}^\bullet p = \emptyset$. By $\min(\Pi)$ we denote the set of $\prec_\Pi$-minimal places of Π, i.e., $\min(\Pi) = \{p \in P \mid {}^\star p = \perp\}$. Process nets are implicitly considered marked, with initial marking $\min(\Pi)$. With abuse of notation, in the following we shall write $p \in \Pi$ and $t \in \Pi$ avoiding explicit mention of the components P and T of Π, as this is not likely to create ambiguity. Analogously, we usually drop the subscript from $\prec_\Pi$.

Definition 4 (PT Process).
A *map* $\sigma\colon (P, T, F, M) \to (P', T', F', M')$ of marked PT Petri nets is a function $\sigma\colon P \cup T \to P' \cup T'$ mapping P to P' and T to T' such that $\sigma(M) = M'$, and for all $t \in T$, $\sigma({}^\bullet t) = {}^\bullet\sigma(t)$ and $\sigma(t^\bullet) = \sigma(t)^\bullet$.
A *process* π of (N, M) is a map $\pi\colon \Pi \to (N, M)$, for Π a finite process net.

The notion of slice, which provides a snapshot of a running process' state, plays a role in our development. We say that $x, y \in \Pi$ are *concurrent* if neither $x \prec y$ nor $y \prec x$. A *slice* of π is a maximal set of concurrent places of Π. E.g., the PT net underlying the net of Figure 3 has precisely four slices: $\{p_1, p_2\}$, $\{p_1, p_4\}$, $\{p_2, p_3\}$, $\{p_3, p_4\}$. We use $S_\prec = \{t \mid t \prec s, s \in S\}$ and $S^\prec = \{t \mid s \prec t, s \in S\}$ to indicate the two parts a slice S partitions the transitions of Π into.

Processes of DTAPNs rest upon the notion of PT net process, enriching it with a suitable treatment of time constraints. Each firing of a transition is time-stamped with the time elapsed since the process began, according to each of the 'clocks' (E-equivalence classes) involved.

Definition 5 (DTAPN Net Processes).
Let $N = (P, T, F, c, E, D, M)$ be a marked DTAPN. A process of N is a process $\pi\colon \Pi \to N$ of the underlying PT net together with a $\lesssim$-totally preordered family $\delta = \{\delta_t\colon P/E \rightharpoonup D\}_{t\in\Pi}$ of partial functions such that $\delta_t(x)$ is defined if and only if $\pi({}^\bullet t \cup t^\bullet) \cap x \neq \emptyset$ and for each arc (p, t) of Π

$$\delta_t([\pi(p)]_E) - \delta_{{}^\star p}([\pi(p)]_E) \in c(\pi(p), \pi(t)),$$

where, by convention, $\delta_\perp(x) = 0$ for all x, and where $\delta_t \lesssim \delta_{t'}$ if $\delta_t(x) \leq \delta_{t'}(x)$ for all $x \in \mathrm{dom}(\delta_t) \cap \mathrm{dom}(\delta_{t'})$.

The condition above enforces the time constraints c on the arcs by bounding appropriately the difference between tokens' creation and consumption times. The special case of $\delta_\perp$ deals with 0-aged tokens in the initial marking. For GT nets, each δ_t reduces to a single time-stamp according to the (unique) global clock. In the case of multiple clocks, the preorder $\lesssim$ ensures that the time domain is consistent, ruling out situations in which concurrent transitions have incompatible perceptions of the time elapsed. Notice that the linearity condition does not mean sequential processes, as we may have both $\delta_t \lesssim \delta_{t'}$ and $\delta_{t'} \lesssim \delta_t$.

Slices need refinement to adapt to our timed model. Observe, in fact, that $\{p_1, p_4\}$ can never be a slice of any process when the net in Figure 3 is considered as a GT net, as the behaviour in which t_2 occurs before t_1 is not realisable. We shall thus define a slice of a DTAPN process to be a slice S of the underlying PT process such that $\delta_t \lesssim \delta_{t'}$, for all $t \in S_\prec$ and all $t' \in S^\prec$.

We now proceed to prove an important sanity condition for our processes, by relating them to firing sequences and markings. In order to extract a marking from a slice, the following definition determines the age of tokens in each places as the difference between the time-stamp of the slice according to clock x (viz. $\max(S, x)$) and the time (according to the same x) when the token was generated. This allows us to pass from the absolute time on processes' transitions to the relative one found in firing sequences' markings.

Definition 6 (Markings Compatible with a Slice).
Let $(\pi\colon \Pi \to N, \delta)$ be a process of a marked DTAPN N and S a slice thereof. Marking M_S is associated to S if only places in $\pi(S)$ are marked, and for each $\bar{p} \in \pi(S)$,

$$M_S(\bar{p}) = \{\max(S, [\bar{p}]_E) - \delta_x([\bar{p}]_E) \mid x = {}^\star p,\ \pi(p) = \bar{p}\}$$

where $\max(S, x) = \max\{\delta_t(x) \mid t \in {}^\bullet S,\ x \in \mathrm{dom}(\delta_t)\}$, convening that $\max\emptyset = 0$. The set of markings of N compatible with S is

$$m(\pi, S) = \{M \mid M_S[\epsilon\rangle M,\ \text{for } \epsilon \text{ a } \textit{time-elapsing} \text{ transition of } N\}.$$

Theorem 1. *Let (N, M) be a DTAPN and (π, δ) a process thereof. For each slice S of π and each $M' \in m(\pi, S)$ there exists a firing sequence $M \longrightarrow^* M'$.*

Proof. By induction on the size of $S_\prec$. The base case is easy, for $S_\prec = \emptyset$. In the induction step, we must have $t \in S_\prec$ with $t^\bullet \subseteq S$. Among these, choose t

one with $\lesssim$-maximal δ_t, that exists by hypothesis on δ, so to ensure that $S \setminus t^\bullet$ is a slice. By induction hypothesis, there is a firing sequence $M \to^* M_1$, where M_1 is a marking in $m(\pi, S \setminus t^\bullet)$ such that $M_1[t\rangle M_2$. Then, by an appropriate time-elapsing transition, $M_2[\epsilon\rangle M'$, as required. □

Theorem 2. *Let (N, M) be a DTAPN. For each firing sequence $M \longrightarrow^* M'$ of N, there exists a process (π, δ) such that $M' \in m(\pi, S)$, for S a slice of π.*

Proof. Easy, by induction on the length of the firing sequence. □

The difference between GT nets and LT nets is reflected in our formalisation above in two related aspects. Firstly, GT nets have fewer processes, due to the more stringent synchronisation constraints. Secondly, these processes have fewer slices, that is a smaller internal concurrency. This is nicely summarised in $\lesssim$, that is a 'loose' preorder in the case of LT nets, and essentially becomes a 'tight' linear order for GT nets.

4 Reachability and Coverability

LT and GT nets can be compared on the grounds of various decidability questions. Ruiz, Gomez and Escrig recently proved in [15] that reachability is undecidable for GT nets. Their proof does not imply undecidability for LT nets, because it relies on synchronised places. In principle, it may seem that the model of LT nets is less powerful than the one of GT nets. Nevertheless, we demonstrate that reachability for LT nets is undecidable as well. The proof is based on a reduction from the halting problem of Minsky machine with two counters. Notice that this contrasts with the result by Mayr [10] stating the decidability of reachability for ordinary Petri nets. The reachability problem for local timed-arc Petri nets can be formulated as follows.

Problem:	Reachability for LT nets.
Instance:	A marked LT net (N, M) and a final marking M'.
Question:	$M \longrightarrow^* M'$?

Definition 7 (Minsky machine with two counters).
Minsky machine R with two counters c_1 and c_2 is a finite sequence

$$R = (L_1 : I_1,\quad L_2 : I_2,\quad \ldots,\quad L_n : I_n)$$

where $L_1, \ldots, L_n$ are pairwise different *labels*, $I_1, \ldots, I_n$ are *instructions* such that $I_1, \ldots, I_{n-1}$ are exactly of one of the following types:

– *increment:* $c_r := c_r + 1$; `goto` L_j
– *test and decrement:* `if` $c_r = 0$ `then goto` L_j `else` $c_r := c_r - 1$; `goto` L_k

where $1 \leq r \leq 2$ and $1 \leq j, k \leq n$. The last instruction I_n is always a special instruction `halt`.

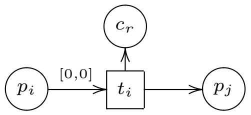

Fig. 4. Increment instruction

A machine R starts its execution (with given input values of c_1 and c_2) from the instruction I_1 and it *halts* if it reaches the instruction `halt` in a finite number of steps. Otherwise it *diverges*. The halting problem for a machine R with the initial values of counters $c_1 = c_2 = 0$ is known to be undecidable. The following variant of the problem is easily seen to be undecidable as well.

Problem:	Halting problem with empty counters.
Instance:	A Minsky machine R with $c_1 = c_2 = 0$.
Question:	Does R halt and both counters are empty?

Given a Minsky machine $R = (L_1 : I_1,\ \ L_2 : I_2,\ \ \dots,\ \ L_n : I_n)$ we construct a local timed-arc Petri net N with continuous time which weakly simulates the machine R. We define $N = (P, T, F, c, \Delta_P, \mathbb{R}_0^+)$ where

$$\begin{aligned} P = \quad & \{p_i \mid 1 \le i < n,\ I_i \text{ of type increment}\} \\ & \cup \{p_i, p_i^1, p_i^2, p_i^3, p_i^4, p_i^5 \mid 1 \le i < n,\ I_i \text{ of type test and decrement}\} \\ & \quad \cup \{p_n, c_1, c_2, p_s, p_e\}, \end{aligned}$$

$$\begin{aligned} T = \quad & \{t_i \mid 1 \le i < n,\ I_i \text{ of type increment}\} \\ & \cup \{t_i, t_i^1, t_i^2, t_i^3, t_i^4, t_i^5, t_i^6 \mid 1 \le i < n,\ I_i \text{ of type test and decrement}\} \\ & \quad \cup \{t_s, t_e\}, \end{aligned}$$

F and c are described in the text below.

For every instruction $L_i : c_r := c_r + 1;$ `goto` L_j, with $1 \le i < n$ and $1 \le r \le 2$, we add the arcs between places and transitions depicted in Figure 4. For every instruction L_i : `if` $c_r = 0$ `then goto` L_j `else` $c_r := c_r - 1;$ `goto` L_k, with $1 \le i < n$ and $1 \le r \le 2$, we add the arcs depicted in Figure 5. Moreover we add a starting and an ending transition, as illustrated in Figure 6. Initial and final markings M and M' are

$$M(p) = \begin{cases} \{0\} & \text{if } p = p_s \\ \{0,0\} & \text{if } p \in \{c_1, c_2\} \\ \emptyset & \text{otherwise,} \end{cases} \qquad M'(p) = \begin{cases} \{0\} & \text{if } p = p_e \\ \emptyset & \text{otherwise.} \end{cases}$$

We prove that $M \longrightarrow^* M'$ if and only if R halts on the input $c_1 = c_2 = 0$ with both counters empty. Thus we show that reachability for LT nets is undecidable.

Lemma 1. *If R halts with both counters empty then $M \longrightarrow^* M'$.*

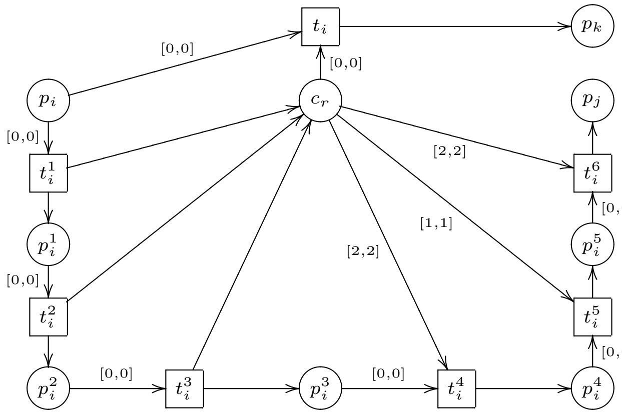

Fig. 5. Test and decrement instruction

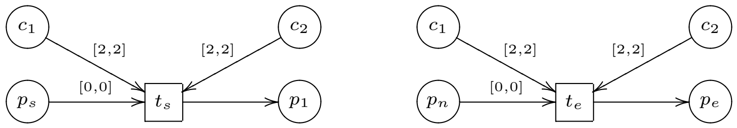

Fig. 6. Start and end of the simulation

Proof. Suppose that after a finite sequence of instructions executed by R the machine stops in the instruction L_n : `halt` with both counters empty. Then we can simulate this sequence in N as follows.

First, we let the four tokens in places c_1 and c_2 reach the age 2 and then we fire the starting transition t_s, which puts a token into place p_1 and in both c_1 and c_2 remains one token of age 2. An increment instruction I_i is simulated by firing the transition t_i without any time elapsed. If I_i is a test and decrement instruction and the corresponding place c_r contains a token of age 0, we fire the transition t_i. Again there is no time-elapsing transition. If the place c_r contains only one token of age 2 then we fire the sequence of transitions $t_i^1, t_i^2, t_i^3, t_i^4, t_i^5, t_i^6$. First, three tokens of age 0 are added and then the token of age 2 is removed by the transition t_i^4. Then we allow to pass one time unit in the place c_r, which means that c_r now contains three tokens of age 1. We consume one of them by firing the transition t_i^5 and let pass another time unit in c_r. Then we fire the transition t_i^6. The resulting marking contains one token of age 2 in c_r, one token of age 0 in p_j and the place c_{3-r} is untouched. Eventually a token of age 0 appears in the place p_n and the places c_1 and c_2 contain one token of age 2 each. That means that we can fire the ending transition t_e and reach M'. □

Lemma 2. *If $M \longrightarrow^* M'$ then the machine R halts with both counters empty.*

Proof. We can naturally simulate the behaviour of the net N by executing the corresponding instructions of the machine R. The only problematic case is when a transition t_i^1 is fired and the counter c_r is non-empty. However, we show that if this happens then the marking M' cannot be reached.

First observe that the only transition that can be fired from M is t_s and there must be a time-elapsing transition before it. Thus the resulting marking contains one token of age 0 in p_1 and one token of age 2 in both c_1 and c_2. Notice that whenever a token of age strictly greater than 2 appears in c_1 or c_2 (we call such a token *dead*), M' is not reachable. The same happens if a token of age different from 0 appears in some of the places $p_1, p_2, \ldots, p_n$. Thus a time-elapsing transition cannot occur if we aim to reach the marking M'. The values of counters c_1 and c_2 are represented by the corresponding number of tokens of age 0 in the places c_1 and c_2 respectively, with one additional token of age 2. Suppose that we fire a 'cheating' sequence $t_i^1, t_i^2, t_i^3, t_i^4, t_i^5, t_i^6$ such that c_r contains except for one token of age 2 also a non-zero number of tokens of age 0. By examining all possibilities of firing this sequence (we want to avoid dead tokens), we end up with having at least two tokens of age 2 in c_r and moreover all tokens in c_r are of age 2. Notice that we cannot fire the transition t_e if there is more than one token in c_1 or c_2. Should M' be reachable, we have to fire another sequence of $t_i^1, t_i^2, t_i^3, t_i^4, t_i^5, t_i^6$. However, now there are at least two tokens of age 2 in c_r and all other tokens are of age 0. During firing of t_i^1, t_i^2, t_i^3 no time-elapsing is allowed (otherwise dead tokens appear). After t_i^4 is fired, there still remains at least one token of age 2 but there is no token of age 1 to enable t_i^5. This means that a time unit must pass in c_r to enable t_i^5, which causes that a dead token of age 3 appears in c_r. Thus whenever a 'cheating' sequence is fired, the marking M' cannot be reached. This means that the simulation is faithful and if $M \longrightarrow^* M'$ then R halts with both counters empty. $\square$

Notice that the same construction works also for discrete time.

Theorem 3. *Reachability for LT nets is undecidable.*

On the other hand, it is sufficient to restrict the class of nets we consider very slightly in order to separate local and global timed nets.

Definition 8 (Safe marking and safe DTAPN).
A marking $M\colon P \to \mathcal{B}(D)$ is *safe* if $|M(p)| \leq 1$ for every $p \in P$. A marked DTAPN (N, M) is *safe* if the initial marking M is safe.

We now turn to show that the reachability problem for safe LT nets is decidable. Before proving the key decidability lemma, we fix some notation. For $N = (P, T, F, c, E, D)$ a DTAPN, let N^o denote the underlying ordinary Petri net (P, T, F). We define a time-forgetting function $f\colon [P \to \mathcal{B}(D)] \to \mathcal{B}(P)$, mapping DTAPN markings to PT net markings; the multiset $f(M)$ has precisely as many copies of p as there are numbers in $M(p)$. For $M^o \in \mathcal{B}(P)$ a marking of N^o, we use $\mathcal{T}(M^o) \subseteq 2^{[P \to \mathcal{B}(D)]}$ to denote the set of all timed-markings that have in each place the same number of tokens as M^o, i.e., $\mathcal{T}(M^o) = f^{-1}(M^o)$.

Lemma 3. *Let (N, M) be a safe marked LT net. If $f(M) \longrightarrow^* M_1^o$ in N^o then $M \longrightarrow^* M_1$ in N for every $M_1 \in \mathcal{T}(M_1^o)$.*

Proof. By induction on k we prove that if $f(M) \longrightarrow^k M_1^o$ in N^o then $M \longrightarrow^* M_1$ in N for every $M_1 \in \mathcal{T}(M_1^o)$.

Base case: If $k = 0$ then $M_1^o = f(M)$ and it can be easily seen that $M[\epsilon\rangle M_1$ for every $M_1 \in \mathcal{T}(f(M))$. Hence $M \longrightarrow M_1$ for every $M_1 \in \mathcal{T}(M_1^o)$.

Induction step: Let $k > 0$. Assume that $f(M) \longrightarrow^{k-1} M_1^o\ [t\rangle\ M_2^o$ in N^o. Let us fix an arbitrary $M_2 \in \mathcal{T}(M_2^o)$. We show that $M \longrightarrow^* M_2$ in N. By $\min_p$ we denote $\min\{M_2(p)\}$. We define a marking M_1 in N as follows:

$$M_1(p) = \begin{cases} M_2(p) & \text{if } p \in P \setminus ({}^\bullet t \cup t^\bullet) \\ M_2(p) \cup \{x\} & \text{if } p \in {}^\bullet t \setminus t^\bullet \text{ and } c(p,t) = (x, \_) \\ \big(M_2(p) \twoheadleftarrow (-\min_p)\big) \setminus \{0\} & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ \big(M_2(p) \twoheadleftarrow (-\min_p)\big) \setminus \{0\} \cup \{x\} & \text{if } p \in {}^\bullet t \cap t^\bullet \text{ and } c(p,t) = (x, \_). \end{cases}$$

Obviously, $M_1 \in \mathcal{T}(M_1^o)$. Because of induction hypothesis we know that $M \longrightarrow^* M_1$ in N. We prove our lemma by showing that $M_1 \longrightarrow^* M_2$. It is, however, easy to see that $M_1\ [t\rangle\ M_1'\ [\epsilon\rangle\ M_2$, where

$$M_1'(p) = \begin{cases} M_2(p) & \text{if } p \in P \setminus t^\bullet \\ M_2(p) \twoheadleftarrow (-\min_p) & \text{if } p \in t^\bullet \end{cases} \qquad \epsilon(\{p\}) = \begin{cases} 0 & \text{if } p \in P \setminus t^\bullet \\ \min_p & \text{if } p \in t^\bullet. \end{cases}$$

□

Theorem 4. *Reachability is decidable for safe LT nets, but undecidable for safe GT nets.*

Proof. Let (N, M) be a safe LT net. Trivially, for any marking M_1 reachable from M it is the case that $f(M_1)$ is reachable from $f(M)$ in N^o. Hence, using Lemma 3, the reachability problem for safe LT nets is reduced to the reachability problem for ordinary Petri nets, and this problem is decidable [10]. The reason for undecidability of reachability of safe GT nets is that in the undecidability proof from [15] the initial marking is safe. □

The coverability problem for GT nets was shown to be decidable — for discrete time in [14] and for continuous time in [1]. Following these results one proves that coverability is decidable even for DTAPNs.

Problem:	Coverability for DTAPNs.
Instance:	A marked DTAPN (N, M) and a final marking M'.
Question:	$\exists M''.\ M \longrightarrow^* M'' \ \wedge\ \forall p \in P.\ M'(p) \subseteq M''(p)$?

Theorem 5. *Coverability for DTAPNs is decidable.*

5 Conclusion

We have studied a recently introduced model of distributed timed-arc Petri nets. The model is well motivated and captures e.g. the ideas behind the Globally

Asynchronous Locally Synchronous paradigm. We provide a formal process semantics for the model and compare the expressiveness of LT nets versus GT nets. We give answers to the most frequently studied decidability problems for Petri net models — reachability and coverability — finding a very delicate decidability borderline in the case of reachability for LT nets.

References

1. P.A. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proc. of ICATPN 2001*, volume 2075 of *LNCS*, pages 53–70, 2001.
2. T. Bolognesi and P. Cremonese. The weakness of some timed models for concurrent systems. Technical Report CNUCE C89-29, CNUCE–C.N.R., 1989.
3. T. Bolognesi, F. Lucidi, and S. Trigila. From timed Petri nets to timed LOTOS. In *Proc. of the IFIP WG 6.1 Tenth International Symposium on Protocol Spec., Testing and Verification*, pages 1–14, Amsterdam, 1990.
4. Fred D.J. Bowden. Modelling time in Petri nets. In *Proceedings of the Second Australia-Japan Workshop on Stochastic Models*, 1996. `http://www.itr.unisa.edu.au/~fbowden/pprs/stomod96/`.
5. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A model-checking tool for real-time systems. In *Proc. of CAV'98*, volume 1427 of *LNCS*, pages 546–550, 1998.
6. U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Computation*, 57:125–147, 1983.
7. H.M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. In *Application and Theory of Petri Nets*, volume 691 of *LNCS*, pages 282–299, 1993.
8. S. Hauck. Asynchronous design methodologies: An overview. In *Proc. of IEEE*, volume 83, pages 69–93, 1995.
9. K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
10. E.W. Mayr. An algorithm for the general Petri net reachability problem (preliminary version). In *Proc. of 13th Ann. ACM Symposium on Theory of Computing*, pages 238–246. Assoc. for Computing Machinery, 1981.
11. T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Berg, D. Lindqvist, H. Tenhunen, and A.Postula. Evaluating benefits of globally asynchronous locally synchronous VLSI architecture. In *Proceedings of 16th Norchip*, pages 50–57, 1998.
12. M. Nielsen, V. Sassone, and J. Srba. Towards a notion of distributed time for Petri nets. In *Proc. of ICATPN 2001*, volume 2075 of *LNCS*, pages 23–31, 2001.
13. C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets.* Ph.D. Thesis, Massachusetts Inst. of Tech., Cambridge, 1973.
14. V. Valero Ruiz, D. de Frutos Escrig, and O. Marroquin Alosno. Decidability of properties of timed-arc Petri nets. In Proc. of *ICATPN 2000*, volume 1825 of *LNCS*, pages 187–206, 2000.
15. V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proc. of PNPM'99*, pages 188–196, 1999.
16. J. Sifakis. Use of Petri nets for performance evaluation. In *Proc. of the 3rd International Symposium IFIP W.G. 7.3., Measuring, modelling and evaluating computer systems* , pages 75–93. Elsevier Science Publ., 1977.
17. J. Wang. *Timed Petri Nets, Theory and Application.* Kluwer Acad. Publ., 1998.