On Non-Decidability of Reachability for Timed-Arc Petri Nets

V. Valero Ruiz

Dpto. Informática
Esc. Politécnica
Univ. Castilla-La Mancha
Albacete, SPAIN 02071
valentin@info-ab.uclm.es

D. de Frutos Escrig

Dpto. Sist. Inf. y Programación
Fac. Matemáticas
Univ. Complutense
Madrid, SPAIN 28040
defrutos@dia.ucm.es

F. Cuartero Gómez

Dpto. Informática *
Esc. Politécnica
Univ. Castilla-La Mancha
Albacete, SPAIN 02071
fernando@info-ab.uclm.es

Abstract

Timed-arc Petri nets are not Turing powerful, because, in particular, they cannot simulate a counter with test on zero. Thus, we could think that this model does not extend significatively the expressiveness of untimed Petri nets. But this is not true; in this paper we show that the differences between them are big enough to make the reachability problem undecidable. We also define dead tokens as those that cannot be used for firing any transitions in the future and we present some particular cases where we can identify them on this kind of timed nets.

1. Introduction

Petri nets are widely used for the modeling and analysis of concurrent systems, because of their graphical nature and the solid mathematical foundations supporting them. Several timed extensions of the basic model have been proposed to expand their application areas to those systems which exhibit a time-dependent behaviour that should be considered both in the modeling and the analysis process, such as distributed systems, communication systems and real-time systems.

A survey of the different approaches to introduce time into Petri nets is presented in [6]. We can identify a first group of models, which assign time delays to transitions, either using a fixed and deterministic value [12, 13, 14] or choosing it from a probability distribution [3]. Other models use time intervals to establish the enabling times of transitions [9]. Finally, we have also some models that introduce time on tokens [1, 2, 5]. In such a case tokens become classified into two different classes: available and unavailable ones. Available tokens are those that can be immediately used for firing a transition, while unavailable cannot. We have to wait for a certain period of time for these tokens to become available, although it is also possible for a token to remain unavailable forever (such tokens are said to be *dead*). More recently, Cerone and Maggiolo-Schettini [7] have defined a very general model (statically timed Petri nets), where timing constraints are intervals statically associated with places, transitions and arcs. Thus, models with timing constraints attached only to places, transitions or arcs can be obtained by considering particular subclasses of this general framework.

In this work we analyse timed-arc Petri nets (TAPN's) [5, 15, 8], a timed extension of Petri nets in which tokens have associated a non-negative real value indicating the elapsed time from its creation (*its age*), and arcs from places to transitions are also labelled by time intervals, which establish restrictions on the age of the tokens that can be used to fire the adjacent transitions. As a consequence of these restrictions some tokens may become *dead*, because they will be never available, since they are too old to fire any transitions in the future, and thus they stay attached to its place forever, growing and growing.

The only difference with respect to the model presented in [5] is that we are a bit more general, since we allow real values as extremes of the time intervals, instead of just natural numbers. In that paper, Bolognesi *et. al* describe timed-arcs Petri nets, comparing them with Merlin's model in the framework of design and analysis of concurrent systems. The interpretation and use of timed-arcs Petri nets can be obtained from a collection of processes interacting with one another according to a rendez-vous mechanism. Each process may execute either local actions or synchronization ones. Local actions are those that the process may execute without cooperation from another process, and thus in the Petri net model of the whole system they would appear as transitions with a single precondition place, while synchro-

*This work has been supported by the CICYT project "Formal Development of Distributed Systems", TIC97-0669-c02.

nization actions would have several precondition places, which correspond to the states at which each one of the involved processes is ready to execute the action. Then, each time interval establishes some timing restrictions related to a particular process (for instance the time that a local processing may require). In consequence, the firing of a synchronization action can be done in a time window, which depends on the age of the tokens on its precondition places.

One of the main advantages of timed-arc Petri nets is that it is quite easy to get a timed-arc Petri net modeling a system that has been described by a *Timed-LOTOS* specification [11]. Therefore, also from a methodological point of view, the interest of the model is justified, as a graphical tool for the design and analysis of concurrent systems.

In [4], it is proved that timed-arc Petri nets are not Turing complete, since in particular they cannot correctly simulate a 2-counter machine. Thus, we could expect that the differences with untimed Petri nets in terms of expressiveness would not be rather significant. Nevertheless, we show in this paper that the difference is big enough to make undecidable the reachability problem for timed-arc Petri nets.

Timed-arc Petri nets are also considered in [8], but considering natural numbers for the annotations of the timed-arcs, and enforcing the firing of transitions with an earliest and maximal firing rule. In that paper timed-arc Petri nets are used for the modeling and analysis of batch production systems. With this goal, a dynamic state graph is defined for bounded nets on the basis of a maximal value for the age of a token on a place to influence the activation of its postcondition transitions. This graph can also be constructed for TAPN's without imposing the urgent firing of transitions, but let us note that even for bounded timed nets, with this graph we cannot determine in general if a given marking is reachable or not, because we lose some information concerning the age relations.

The paper is structured as follows. In Section 2 we present timed-arc Petri nets and their semantics, in Section 3 we prove the undecidability of reachability, and finally, in Section 4 we present some approaches to the problem of detecting and eliminating dead tokens.

2. Timed-Arc Petri Nets

We deal with timed-arc Petri nets, which have their tokens annotated with an age (a real value indicating the elapsed time from its creation) and arcs connecting places with transitions have associated a time interval, which limits the age of the tokens to be consumed to fire the adjacent transition.

However, a transition is not forced to be fired when all its preconditions contain tokens with an adequate age, and the same is true even if the age of any of these tokens is about to expire. More in general, in the model there is not any kind of urgency, what we can interpret in the sense that the model is *reactive*, as transitions will be only fired when the external context requires it. But then, it can be the case that the external context may lose the ability to fire a transition if some needed tokens become too old. Even more, it is possible that some tokens become *dead*, which means definitely useless because the increasing of their age will not allow in the future the firing of any of their postcondition transitions.

Definition 1 (Timed-arc Petri nets)
We define a timed-arc Petri net (TAPN) as a tuple [1] $N = (P, T, F, \mathit{times})$, where P is a finite set of *places*, T is a finite set of transitions ($P \cap T = \emptyset$), F is the *flow relation* ($F \subseteq (P \times T) \cup (T \times P)$), and *times* is a function that associates a closed time interval to each arc (p, t) in F, i.e.: $\mathit{times} \,:\, F|_{P \times T} \longrightarrow \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$.

When $\mathit{times}(p, t) = [t_1, t_2]$ we write $\pi_i(p, t)$ to denote t_i, for $i = 1, 2$.

As we previously mentioned, tokens are annotated with real values, so markings are defined by means of multisets on $\mathbb{R}_0^+$. More exactly, a marking M is a function:

$$M \,:\, P \longrightarrow \mathcal{B}(\mathbb{R}_0^+)$$

where $\mathcal{B}(\mathbb{R}_0^+)$ denotes the set of finite multisets of non-negative real numbers. Thus, as usual, each place is annotated with a certain number of tokens, but each one of them has associated a non-negative real number (its *age*). We will denote the set of markings of N by $\mathcal{M}(N)$, and using classical set notation, we will denote the number of tokens on a place p by $|M(p)|$.

As initial markings we only allow markings M such that for all p in P, and any $x > 0$ we have $M(p)(x) = 0$ (i.e., the *initial age* of any token is 0). Then, we define *marked timed-arc Petri nets* (MTAPN) as pairs (N, M), where N is a timed-arc Petri net, and M is an initial marking on it. As usual, from this initial marking we will obtain new markings, as the net evolves, either by firing transitions, or by the passage of time. In consequence, given a non-zero marking, even if we do not fire any transitions at all, starting from this marking we get an infinite reachability set of markings, due to the token aging.

A timed-arc Petri net with an arbitrary marking can be graphically represented by extending the usual representation of P/T nets with the corresponding time information. In particular we will use the age of each token to represent it. Therefore, MTAPN's have initially a finite collection of zero values labelling each place. In Fig. 1 we show a MTAPN modeling a producer/consumer system. □

Let us now see how we can fire transitions, and how we model the passage of time.

[1] We consider only arcs with weight 1 to simplify some definitions, but the extension to general arcs with greater weights is straightforward.

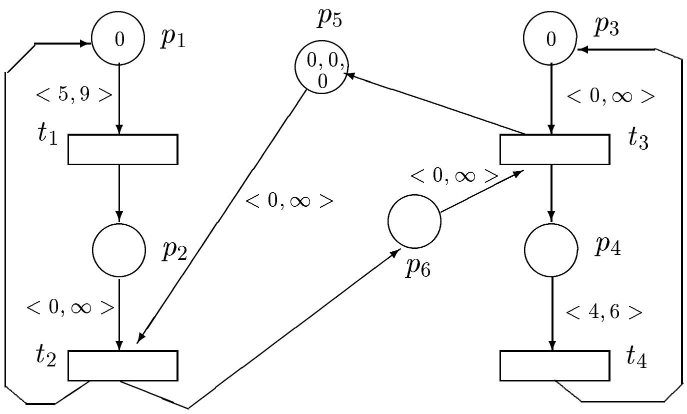

Figure 1. Timed-arc Petri net modeling the PC-problem

Definition 2 (Firing rule)
Let $N = (P, T, F, \mathit{times})$ be a TAPN, M a marking on it, and $t \in T$.

(i) We say that t is *enabled* at the marking M if and only if:

$$\forall p \in {}^\bullet t\ \exists x_p \in \mathbb{R}_0^+ \text{ such that}$$
$$M(p)(x_p) > 0 \land x_p \in \mathit{times}(p,t)$$

i.e., on each precondition of t we have some token whose age belongs to $\mathit{times}(p,t)$.

(ii) If t is enabled at M, it can be fired, and by its firing we reach a marking M', as follows:

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

where both the subtraction and the addition operators work on multisets, and:

- $C^-(p,t) = \begin{cases} \{x_p\} & \text{if } p \in {}^\bullet t, x_p \in \mathit{times}(p,t) \\ & \text{and } x_p \in M(p) \\ \emptyset & \text{otherwise} \end{cases}$
- $C^+(t,p) = \begin{cases} \emptyset & \text{if } p \notin t^\bullet \\ \{0\} & \text{otherwise} \end{cases}$

Thus, from each precondition place of t we remove a token fulfilling (i), and we add a new token (with age 0) on each postcondition place of t.

As usual, we denote these evolutions by $M[t\rangle M'$, but it is noteworthy that these evolutions are in general non-deterministic, because when we fire a transition t, some of its precondition places could hold several tokens with different ages that could be used to fire it. Besides, we see that the firing of transitions does not consume any time. Therefore to model the passage of time we need the function *age*, defined below. By applying it we age all the tokens of the net by the same time:

(iii) The function $\mathit{age} : \mathcal{M}(N) \times \mathbb{R}_0^+ \longrightarrow \mathcal{M}(N)$ is defined by:

$$\mathit{age}(M,x)(p)(y) = \begin{cases} M(p)(y-x) & \mathit{if}\ y \geq x \\ 0 & \mathit{otherwise} \end{cases}$$

The marking obtained from M after x units of time without firing any transitions will be that given by $\mathit{age}(M,x)$.

□

Although we have defined the evolution by firing single transitions, this can be easily extended to the firing of *steps* or *bags* of transitions; those transitions that could be fired together in a single step could be also fired in sequence in any order, since no *aging* is produced by the firing of transitions. In this way we obtain step transitions that we denote by $M[R\rangle M'$. Finally, by alternating step transitions and the passage of time we can define a timed step semantics, where timed step sequences are those sequences $\sigma = M_0[R_1\rangle_{x_1} M_1 \ldots M_{n-1}[R_n\rangle_{x_n} M_n$, where $M_i's$ are markings, $R_i's$ multisets of transitions and $x_i's \in \mathbb{R}_0^+$, in such a way that $M_i[R_{i+1}\rangle M'_{i+1}$ and $M_{i+1} = \mathit{age}(M'_{i+1}, x_{i+1})$. Note that we allow $x_i = 0$ in order to capture the execution in time zero of two causally related steps.

Then, given a MTAPN (N, M_0), we define $[M_0\rangle$ as the set of reachable markings on N starting from M_0, and we say that N is bounded if for every $p \in P$ there exists $n \in \mathbb{N}$ such that for all $M \in [M_0\rangle$ we have $|M(p)| \leq n$.

3. Reachability Analysis

In this section we show that given an arbitrary MTAPN (N, M_0) and a marking M on it, it is undecidable whether $M \in [M_0\rangle$ or not. As a consequence of this result we will also conclude the undecidability of detection of *dead tokens*.

The proof exploits the well-known fact that a Turing machine can be simulated by a 2-counter machine, which consists of two counters r_1, r_2 and a program whose instructions are numbered, satisfying the following requirements:

- The program starts at its first instruction, I_1.
- There is an instruction I_e which signals the end of the program, when it is reached.
- There are two kinds of instructions:
  - *Increment*, which increases by one the value of a counter:

    $$I_j \; : \; r_i := r_i + 1 \; ; \; goto \; I_k$$

  - *Test & Decrement*, which tests the value of a counter, decrementing it by one when it is not zero:

    $$I_j \;:\; \text{if } r_i > 0 \quad \text{then } r_i := r_i - 1 \; ; \; goto \; I_k \quad \text{else } goto \; I_l$$

Given a 2-counter machine *CM*, if its execution terminates, we know by definition that it does at the instruction I_e, but when this happens the registers may have an arbitrary value. But then, we can extend the program by adding after I_e two loop instructions to *empty* both registers. Then, we have that the original *CM* machine stops if and only if the extended machine does it after reaching the zero state ($r_1 = r_2 = 0$).

It is known that MTAPN's cannot simulate 2-counter machines [4]. Nevertheless, given an extended 2-counter machine, we can define a MTAPN which *weakly* simulates it. This means that the net includes transitions which represent the execution of the instructions of the given program, but those corresponding to test & decrement instructions are not able to correctly test on zero, and then, in order to cover the else branches of these instructions, they can freely select their continuation whatever will be the value of the checked registers. As a consequence, the behaviour of the net may be correct or not, but if the latter is the case we obtain some dead tokens which cannot be removed in the future. Thus, the presence of these dead tokens will reveal us a bad behaviour.

Definition 3 (Weak simulation of 2-counter machines) Given a 2-counter machine *CM* with an extended program (to guarantee that it only stops with empty registers), we construct the following MTAPN to simulate it:

1. For each instruction I_j we have an associated place p_j. In particular, for I_e, the final instruction of the extended program, we will have the corresponding place p_e.

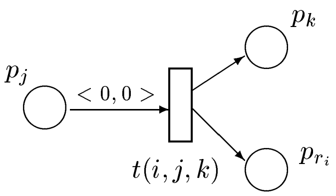

Figure 2. Weak simulation of the increment instruction

2. To simulate each register we have a place, p_{r_i}, $i = 1, 2$. The number of tokens on p_{r_i} will represent the value of r_i.
3. Each instruction $I_j \; : \; r_i := r_i + 1 \; ; \; goto \; I_k$ is simulated by means of a transition $t(i, j, k)$, as indicated in Fig. 2.
4. Each instruction

   $$I_j \;:\; \text{if } r_i > 0 \quad \text{then } r_i := r_i - 1 \; ; \; goto \; I_k \quad \text{else } goto \; I_l$$

   is simulated as indicated in Fig. 3, where we have included two new places $\{q_n(i, j, k, l) \mid n = 1, 2\}$, and five new transitions $\{u_n(i, j, k, l) \mid n = 1, \ldots, 5\}$, while $p_{r_{3-i}}$ is the place corresponding to the register not involved in the instruction.

As initial marking, we just put one token on the place that corresponds to the first instruction, p_1. □

Then, we have the following result, which allows us to conclude the undecidability of the reachability problem for MTAPN's.

Theorem 1 Given an extended 2-counter machine *CM*, and the associated MTAPN (N, M_0) (according to Def. 3), we have that *CM* stops if and only if there is a reachable marking $M \in [M_0\rangle$ such that $|M(p_e)| = 1$ and $|M(p_{r_i})| = 0$, for $i = 1, 2$.

Proof: If *CM* stops, it does so after the execution of a certain finite sequence of instructions terminating at I_e, which can be simulated in (N, M_0). Increment instructions are correctly simulated by firing the corresponding $t(i, j, k)$ transition, which can be executed because the token on p_j will have zero age. As a consequence we obtain a token on p_k (next instruction to be executed) and a new token on p_{r_i} (thus increasing the number of tokens on it). The simulation of the Test & Decrement instructions is much more involved, and we must distinguish two cases that may occur:

- If $r_i > 0$, we can fire u_2, since the token on p_j has zero age, and tokens on p_{r_i} have also zero age. After this firing p_{r_i} loses a token, while p_k becomes marked, as desired.

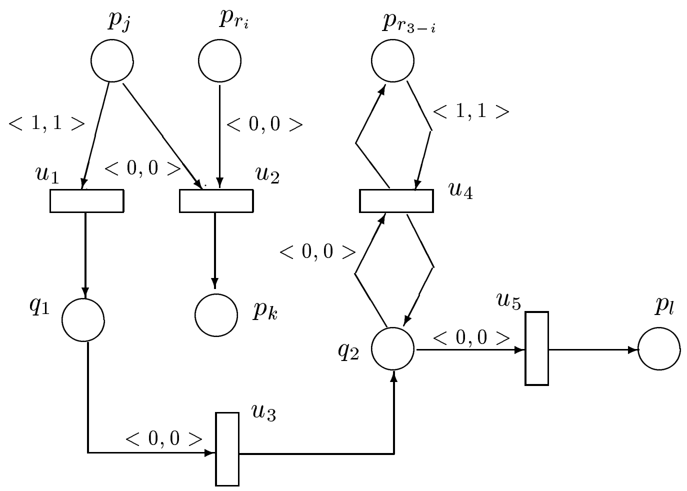

Figure 3. Weak simulation of the test & decrement instruction

- However, when $r_i = 0$, to execute u_1 we must wait for a unit of time. The firing of this transition puts a token on q_1. But since a unit of time has passed, all tokens on $p_{r_{3-i}}$ have now age 1. By firing u_3 we mark q_2, which allows us to fire u_4 as many times as needed in order to *reset* to 0 the age of any token on $p_{r_{3-i}}$. Once this task has been fulfilled we fire u_5 to mark p_l, thus allowing the execution of the instruction I_l. Let us note that no more time passes along this procedure after the execution of u_1.

For the converse, we must take into account that the net semantics only provides a *weak* simulation of the given machine. As explained before, the problem arises when simulating the execution of Test & Decrement instructions, because the net may improperly fire (when p_{r_i} is marked) a u_1 transition. However, when this happens, not only the tokens on $p_{r_{3-i}}$ will be aged by 1, but also the tokens on p_{r_i}. Firing u_3 and u_4, all we can do is to reset the age of tokens on $p_{r_{3-i}}$ to 0. In fact, we are not obliged to reset all of them. Therefore, when we fire u_5 some tokens on $p_{r_{3-i}}$ may have age 1, even if we properly fired u_1. This could be considered to be a second type of bad behaviour of the net. In both cases of bad behaviour we have that the tokens with age 1 on the places p_{r_n} cannot be used any more, because increment instructions require zero age for the tokens they consume, and for Test & Decrement instructions we also have that u_2 transitions take tokens with zero age, and finally, to execute a u_1 transition we must wait for another unit of time, which causes these tokens to have age 2 or older. Thus, there is no way to restore its age to 0, since the firing of u_4 only recovers tokens with age 1. Therefore, we conclude that whenever the net improperly evolves, some *dead tokens* appear on some of the places p_{r_n}, and thus we could never reach a marking M fulfilling the imposed conditions. $\square$

Corollary 1 The reachability problem for MTAPN's is undecidable.

Proof: From the previous theorem we conclude that the *submarking reachability problem* is undecidable for MTAPN's. As an immediate consequence, reachability will be undecidable too, because these problems stay equivalent for MATPN's (the proof for ordinary nets in [10] is also valid in this context). $\square$

According to the previous corollary, we have no way in general to determine if a given marking is reachable or not. Let us observe that even in the case of bounded nets the number of reachable markings is infinite, because of the growing of the token ages. If we restrict ourselves to timed-arc Petri nets with time intervals in $\mathbb{N} \times \mathbb{N} \cup \{\infty\}^2$, we have that for bounded nets we can follow the ideas in [8], in order to construct a state graph, by using the fact that for every place p of a timed-arc Petri net we can find a maximal value $Max(p)$ for the age of its tokens to influence the activation of its postcondition transitions, because the tokens on that place with an age exceeding that maximal value can only be consumed by the firing of some transition $t \in p^\bullet$ for which $\pi_2(p,t) = \infty$.

Concretely we may define:

$$Max(p) = Max\{\pi_i(p,t) \,|\, t \in p^\bullet,\, \pi_i(p,t) < \infty,\, i = 1,2\}$$

and $S(p) = 1 + Max(p)$. Then, we have that once the age of a token on p exceeds $Max(p)$ the only postcondition transitions $t \in p^\bullet$ that could be fired by using that token are those for which $\pi_2(p,t) = \infty$. Obviously, in order to fire such a transition t the age of the involved token on p is

[2] The generalization of this construction to the general case where times are arbitrary real numbers seems much more complicated.

unimportant once it exceeds $S(p)$. This means that in order to construct the state graph we can represent by the single value $S(p)$ the whole interval $[S(p), \infty]$.

Definition 4 (State graph)
Given a bounded MTAPN $N = (P, T, F, \mathit{times}, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, we define its state graph $G(N) = (V, E, M_0)$, where V is the set of markings of N such that the age of the tokens of every place $p \in P$ belongs to $\{0, \ldots, S(p)\}$, M_0 is the initial state and $E \subset V \times T \times \mathbb{N} \times V$.

An arc $(M, t, r, M') \in E$ means that t can be fired at the marking $age(M, r)$ applying Def. 2, and M' is the reached state, but changing for every place p the age of the tokens exceeding $Max(p)$ to $S(p)$.

Thus, we start from M_0 and we apply the modified firing rule to successively get the new reachable states of the graph. We must observe that as we are not imposing the urgent firing of transitions, in principle we could have infinitely many arcs leaving each state. However, for each state $M \in V$ we have that from a certain instant $r \in \mathbb{N}$ onwards, all its outgoing arcs will reach the same state, because the ages of the tokens in each place $p \in P$ at the markings $age(M, s)$, for $s \geq r$, would be greater than their corresponding maximal value, $Max(p)$. As a consequence, for each state $M \in V$ we can limit the application of the firing rule to the markings obtained aging M up to time $r_{max} = Max\{S(p) \,|\, p \in P\}$.

According to this definition, taking into account that V is finite and that we have a finite number of arcs leaving each state, we conclude that $G(N)$ is finite. □

In Fig. 4 we can see a bounded MTAPN and its corresponding state graph. For this net we have that $S(p_1) = 2, S(p_2) = 2, S(p_3) = 1, S(p_4) = 0$ and thus $r_{max} = 2$. Graph states are shown by indicating for each place the number of tokens of a certain age that we have on this place, for instance $1[0]$ means that we have one token with age 0 on the corresponding place.

Then, we have the following result which relates reachable markings of N with states in $G(N)$.

Proposition 1 Given a bounded MTAPN $N = (P, T, F, \mathit{times}, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, and its corresponding state graph $G(N) = (V, E, M_0)$, we have that if $M \in [M_0\rangle$, then there is a state $M' \in V$ such that $\varphi(M) = \varphi(age(M', r))$, for a certain $r \in \mathbb{N}$, where φ is a function that for every place p changes the age of the tokens on p exceeding $Max(p)$ to $S(p)$:

$$\varphi : \mathcal{M}(N) \longrightarrow V$$

$$\varphi(M)(p)(n) = \begin{cases} M(p)(n) & \text{if } n \leq Max(p) \\ \sum_{m \geq S(p)} M(p)(m) & \text{if } n = S(p) \\ 0 & \text{if } n > S(p) \end{cases}$$

Proof: By induction on the timed step sequence σ such that $M_0[\sigma\rangle M$. The base case ($M = M_0$) is trivial, so let us consider a reachable marking M_1 for which we have a state $M_1' \in V$ such that $\varphi(M_1) = \varphi(age(M_1', r_1))$, and both possibilities of evolution, either by the simple passage of time or by firing a transition:

- If $M = age(M_1, r_2)$, we have that $\varphi(M) = \varphi(age(M_1', r_1 + r_2))$, as desired.
- If $M_1[t\rangle M$, we have that t is also enabled at $\varphi(age(M_1', r_1))$, because the change of age that φ makes for the tokens exceeding their maximal value does not affect the enabledness of transitions. Besides, we may use for firing t the adequate tokens to get a state M' such that $\varphi(M) = \varphi(M') = \varphi(age(M', 0))$. □

Thus, the state graph provides us with a necessary condition for a marking to be reachable in a bounded MTAPN N, but in general it will not allow us to determine if a given marking is reachable or not.

4. Eliminating dead tokens

Let us now turn our attention to *dead tokens*.

Definition 5 (Dead tokens)
Given a TAPN $N = (P, T, F, \mathit{times})$, and a marking M on it, we say that a token in M is *dead* if there is no reachable marking $M' \in [M\rangle$ such that a transition $t \in T$ is enabled at it, and it can be fired consuming that token. □

It is clear that dead tokens can be eliminated without affecting the future behaviour of the net, although it is obvious that the reachability set will be somehow affected when we take them out. To be exact, the markings that are reachable from a marking M containing a set of dead tokens D are exactly those that can be obtained by applying the following procedure:

- We remove the tokens in D from M to obtain M^*.
- We consider the reachable markings $M' \in [M^*\rangle$ and for each one of them the duration t of the corresponding time step sequence.
- We add to M' the tokens in D aged by t.

Thus, we are interested in eliminating these dead tokens, because doing so we decrease the number of reachable markings, thus slightly reducing the state explosion.

Unfortunately, it seems not easy to decide whether a token is *dead* or not. In fact, it remains open for us to show if this property is decidable or not. Anyway, we can distinguish a number of situations where dead tokens can be

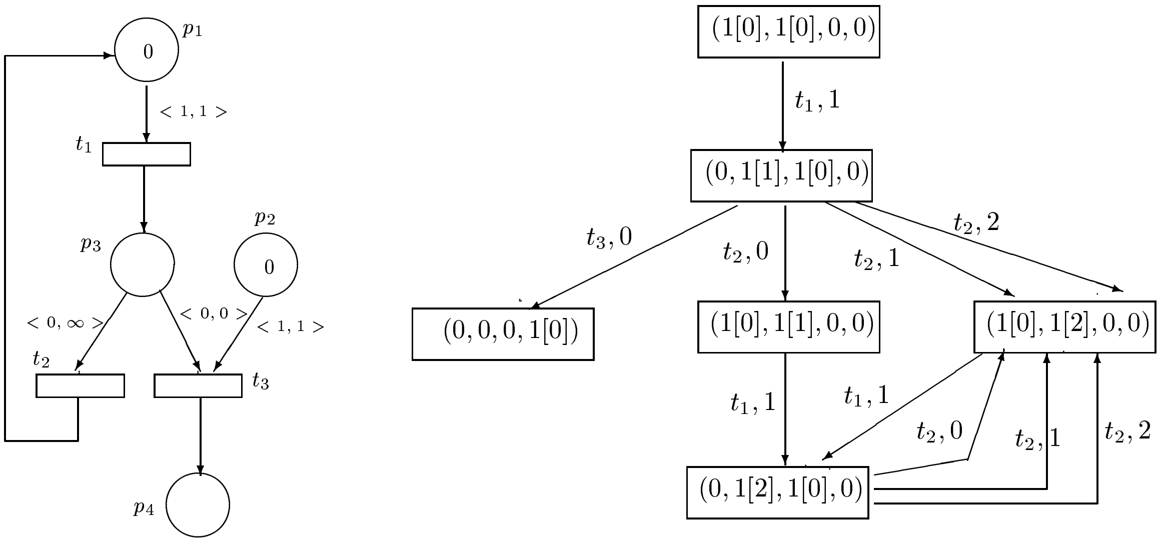

Figure 4. A bounded MTAPN and the corresponding state graph

effectively detected. Next, we present some of these situations. In each one of them we will be able to detect and remove some dead tokens when the corresponding hypotheses are fulfilled. But, since these hypotheses depend on the surrounding tokens, it is possible that the elimination of a token will make another dead token to become detectable, and therefore in some cases it will be useful to reiterate the application of the different detection algorithms in order to try to capture as many dead tokens as possible.

Proposition 2 Given a TAPN $N = (P, T, F, times)$, and a marking M on it, we have that a token in M with age $e \in \mathbb{R}_0^+$ on a place $p \in P$ is *dead* when any of the following conditions is fulfilled:

1. When either $p^\bullet = \emptyset$, or $e > \pi_2(p,t),\ \forall t \in p^\bullet$.
2. If $\forall t \in p^\bullet\ \exists q \in {}^\bullet t$ such that $q \in P' \subseteq P$, where P' is an unmarked siphon.
3. If $\forall t \in p^\bullet\ \exists q \in {}^\bullet t$ such that ${}^\bullet q = \emptyset\ \wedge\ \forall e' \in \mathbb{R}_0^+$, $M(q)(e') > 0 \Rightarrow e' > \pi_2(q,t)$.
4. Whenever $\forall t \in p^\bullet$: $[e + d_t,\ e + D_t] \cap times(p,t) = \emptyset$, where d_t and D_t are the minimum and maximum delay to use this token for the firing of t, which are calculated as follows:

   For each $t \in p^\bullet$ we take ${}^\bullet t = \{p, p_1, \ldots, p_n\}$. Then, we define $A_i = \{e \in \mathbb{R}_0^+,\ \ M(p_i)(e) > 0, e < \pi_2(p_i,t)\}$, for $i = 1, \ldots, n$. These are the ages of the tokens on p_i that could be used to fire t. Let $e_{min}(i)$, $e_{max}(i)$ be the minimum and maximum of these ages, where in the special case $A = \emptyset$ it is adequate to take 0 for both.

   Then, we consider d_i and D_i, which are the minimum and maximum delays for using a token on p_i for the firing of t. They are obtained as follows:

   - $d_i = \pi_1(p_i,t) \ \dot{-}\ e_{max}(i)$
   - $D_i = \begin{cases} \infty & \mathit{if}\ {}^\bullet p_i \neq \emptyset \ \vee\ A_i = \emptyset \\ \pi_2(p_i,t) - e_{min}(i) & \text{otherwise} \end{cases}$

   where $x \ \dot{-}\ y = Max\{0, x - y\}$.

   Finally we take:

   $$\begin{array}{ll} d_t = & \underset{i=1,\ldots,n}{Max} \{d_i\} \\ D_t = & \underset{i=1,\ldots,n}{Min} \{D_i\} \end{array}$$

Proof: The first three cases are immediate, only (4) requires some comments.
For it, we have that in order to fire t at the instant r using the token in study, we would need to consume an available token from each precondition place of t. Then, for each $p_i \in {}^\bullet t$ we will be in one of the following two cases:

- Either we take from p_i a token from M, which had age e_i. In such a case, $e_i \in A_i$, and then when we fire t we have:
  $$\pi_1(p_i,t) \leq e_i + r \leq \pi_2(p_i,t)$$
  Hence:
  $$\pi_1(p_i,t) \ \dot{-}\ e_i \leq r \leq \pi_2(p_i,t) - e_i$$
  And therefore:
  $$d_i \leq r \leq D_i$$
  since $e_{min}(i) \leq e_i \leq e_{max}(i)$.

- Or for firing t we consume a new token that has arrived in the meanwhile to p_i by the firing of some other transition.

  Then, ${}^\bullet p_i \neq \emptyset$, which implies $D_i = \infty$. Besides, we must have $r \geq \pi_1(p_i, t)$, and thus $d_i \leq e_i$.

In any of these cases we obtain $d_i \leq r \leq D_i$, and hence $d_t \leq r \leq D_t$, which is impossible, if the suggested conditions to detect the dead token are fulfilled, since we have that $e + r \in [e + d_t, e + D_t] \cap times(p, t)$. □

5. Conclusions

We have studied timed-arc Petri nets, for which we have proved that reachability is undecidable, by means of a *weak* simulation of a 2-counter machine. We have also seen that even for bounded MTAPN's we have infinitely many reachable markings, because of the token aging. In this case, it is possible to construct a state graph for timed nets with time intervals in $\mathbb{N}$, which provides necessary conditions for reachability.

We have also defined *dead tokens* as those that cannot be used for firing any transitions in the future, but up to now it remains open for us if this property is decidable or not. In order to prove that this property whould be undecidable we tried to reduce it to reachability. But as noticed by an annonymous referee of the former version of the paper, there was a mistake in this reduction. He or she also noted that it is coverability and not reachability the problem with which the detection of dead tokens is related. Therefore, we think now that this problem should be decidable, and we are currently working on the definition of the adequate generalization of the coverability tree for MTAPN's, although we have found that the construction of such a finite structure, and therefore the proof of decidability of coverability, are far from trivial.

Anyway, we have seen that there are some contexts in which dead tokens can be effectively detected. The elimination of *dead* tokens does not affect the timed step sequence semantics, but slightly reduces the reachability set. Then, the elimination of these dead tokens could be used together with some other known techniques for reducing the state explosion (persistent transition sets, stubborn sets, etc.).

Currently, we are also developing some other techniques for the detection and elimination of dead tokens. Also we plan to extend our decidability results to other properties on MTAPN's, like strict reachability (i.e, if a given marking is reachable in a given time), liveness, deadlock freeness, etc.

Finally, we intend to use timed-arc Petri nets as the semantic framework for the definition of the semantics of timed process algebras. In particular we are developing a tool to implement the translation of specifications written in a timed process algebra into the corresponding MTAPN, as well as a set of tools to analyse this kind of nets.

Acknowledgement

We are very grateful to the referees of the former version of the paper for their comments and suggestions, and particularly to one who pointed out the mistake in our attempt to prove the undecidability of the detection of dead tokens. We are also very indebted to Narciso Martí for his careful reading and his constructive comments, which have contributed to improve the presentation of the paper.

References

[1] W.M.P. van der Aalst. *Interval Timed Coloured Petri Nets and their Analysis*. Lecture Notes in Computer Science, vol. 691, pp. 451-472. 1993.

[2] W.M.P. van der Aalst and M.A. Odijk. *Analysis of Railway Stations by Means of Interval Timed Coloured Petri Nets*. Real-Time Systems, vol. 9, pp. 241-263. 1995.

[3] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani. *On Petri Nets with Stochastic Timing*. Proc. of the International Workshop on Timed Petri Nets, IEEE Computer Society Press, pp. 80-87. 1985.

[4] T. Bolognesi and P. Cremonese. *The Weakness of Some Timed Models for Concurrent Systems*. Technical Report CNUCE C89-29. CNUCE-C.N.R. October. 1989.

[5] T. Bolognesi, F. Lucidi and S. Trigila. *From Timed Petri Nets to Timed LOTOS*. Proceedings of the Tenth International IFIP WG6.1 Symposium on Protocol Specification, Testing and Verification. North-Hollan, 1990.

[6] Fred D.J. Bowden. *Modelling time in Petri nets*. Proc. Second Australia-Japan Workshop on Stochastic Models. 1996.

[7] Antonio Cerone and Andrea Maggiolo-Schettini. *Time-based expressivity of time Petri nets for system specification*. Theoretical Computer Science (216)1-2, pp. 1-53. 1999.

[8] Hans-Michael Hanisch. *Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control*. Application and Theory of Petri Nets, LNCS vol. 691, pp:282-299. 1993.

[9] P. Merlin. *A Study of the Recoverability of Communication Protocols*. PhD. Thesis, Univ. of California. 1974.

[10] J. L. Peterson. *Petri net Theory, and the Modeling of Systems*. Prentice-Hall. 1981.

[11] J. Quemada, A. Azcorra, and D. de Frutos. *A Timed Calculus for LOTOS*. Proc. FORTE 89, pp. 245-264. 1989.

[12] C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets*. PhD. Thesis, Massachusetts Institute of Technology, Cambridge. 1973.

[13] J. Sifakis. *Use of Petri Nets for Performance Evaluation*. Proc. of the Third International Symposium IFIP W.G.7.3., Measuring, Modelling and Evaluating Computer Systems. Elsevier Science Publishers, pp. 75-93. 1977.

[14] V. Valero, D. de Frutos, and F. Cuartero. *Decidability of the Strict Reachability Problem for TPN's with Rational and Real Durations*. Proc. 5th. International Workshop on Petri Nets and Performance Models, pp. 56-65. 1993.

[15] B. Walter. *Timed Petri-Nets for Modelling and Analysing Protocols with Real-Time Characteristics*. Proc. 3rd IFIP Workshop on Protocol Specification, Testing and Verification, North-Holland. 1983.