Representation of Time Petri Nets using Interval Weighted Automata

B. Daviaud, S. Lahaye, M. Lhommeau, J. Komenda

▶ **To cite this version:**

B. Daviaud, S. Lahaye, M. Lhommeau, J. Komenda. Representation of Time Petri Nets using Interval Weighted Automata. 2023 9th International Conference on Control, Decision and Information Technologies (CoDIT), Jul 2023, Rome, Italy. pp.99-104, 10.1109/CoDIT58514.2023.10284333 . hal-04419106

HAL Id: hal-04419106

https://univ-angers.hal.science/hal-04419106

Submitted on 26 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Representation of Time Petri Nets using Interval Weighted Automata

B. Daviaud[1], S. Lahaye[1], M. Lhommeau[1] and J. Komenda[2]

Abstract— **Interval Weighted Automata are a modeling formalism for timed systems that can be viewed as an alternative (more algebraic) to timed automata. We present here a way of deriving a deterministic interval weighted automaton to represent any bounded T-time Petri net subject to strong and single-server semantics. This approach consequently contributes to the characterization of the expressiveness of weighted automata with respect to other formalisms for timed Discrete Event Systems. In addition, the time language of the obtained abstraction has the characteristics of being included or equal to the time language of the T-time Petri net, and, in other words, any accessible state in the obtained Interval Weighted Automaton corresponds to an equally accessible state in the T-time Petri net. This property should allow the future use of this abstraction for the verification of properties that are expressed as an accessibility issue and/or for control.**

I. INTRODUCTION

Weighted automata (also known as automata with multiplicities) generalize finite state automata by adding weights to their transitions. The weights have values taken from a semiring, and automata with multiplicities can be conveniently analyzed within an algebraic setting. They have been thoroughly studied in the last fifty years by computer scientists, with applications in various domains such as natural language processing (speech recognition) and digital image compression [1]. This formalism has also been used to address various important problems for timed Discrete Event Systems (DES) such as performance evaluation [2] and control [3], [4].

This paper is a continuation of works that have aimed to identify the expressiveness of weighted automata w.r.t. other DES formalisms. In [5] and [6], it is shown that safe timed Petri nets can be modeled by max-plus automata, i.e. weighted automata with weights in the max-plus semiring. The recent contribution [7] generalizes [8] to show that any bounded timed Petri net under race policy has a behavior-equivalent representation by means of a deterministic max-plus automaton. [9] explores the description of safe P-Time Petri Nets using Interval Weighted Automata (IWA), that is automata with weights corresponding to intervals defined in the product of semirings.

In this article, we focus on the transformation of T-time Petri nets (with the so-called strong and single-server semantics) into IWA. Let us stress the difference with [9] because it is well known that, unlike timed Petri nets, P-time Petri nets and T-time Petri nets (TPN) are not equally expressive [10]. To further explain, situate and motivate our work, several directly related approaches in the literature are mentioned below. For this purpose, let us first remind that in a TPN, a state consists of a marking and a set of constraints, one for each transition enabled at the corresponding marking, that specify when transitions may actually fire. The state spaces of TPNs are typically infinite. In order to make enumerative analysis techniques possible to apply, numerous finite abstractions have been proposed for the state spaces of TPNs. Below we mention only a part of the literature on this subject, limiting ourselves to those references that, to our knowledge, are most relevant to our work.

The seminal work in [11] proposed a graph of so-called state classes. State classes represent some infinite sets of states by a marking of the net and a polyhedron capturing the times at which the enabled transitions may fire [12]. The State Class Graph (SCG) produced is finite if and only if the TPN is bounded. In addition, the SCG edges are labeled by transitions of the TPN, and it has been shown that the language accepted by the SCG is equal to the untimed language of the TPN. Then, SCGs have been successfully used for analyzing TPNs and more precisely for checking untimed reachability properties. An approach preserving branching structure has been proposed in [13] with an alternative definition of state classes, resulting in the atomic state class graph (ASCG). An alternative construction of this graph has been proposed in [14], applicable to a larger class of nets. Nevertheless, the algorithm used to construct the graph seems inefficient and no results can be exploited to compare with other methods [15].

In [16], an approach is proposed to build the SCG as a timed automaton (TA), called State Class Timed Automaton, thus keeping the temporal information of the TPN and making it possible to check timed properties. Obtaining a TA instead of a graph allows one to use all the model checking tools available for TA. To exploit this opportunity, translation techniques from TPNs to TA have been proposed in [17] and [18]. Our work is similar to these approaches dealing with the translation of TPNs, but differs in the expected result, since the aim here is to obtain an IWA.

In order to address state estimation and fault diagnosis problems, the so-called Modified State Class Graph (MSCG) has been proposed in [19] and [20]. Briefly, the MSCG is a directed graph whose nodes are called classes, namely a reachable marking and a set of inequalities that define the timing constraints pertaining to all enabled transitions. Such

[1]Univ Angers, LARIS, SFR MATHSTIC, F-49000 Angers, France. The work of Bérangère Daviaud is financed by Angers Loire Métropole and Université d'Angers {berangere.daviaud, sebastien.lahaye, mehdi.lhommeau}@univ-angers.fr

[2]Institute of Mathematics - Brno Branch, Czech Academy of Sciences, Czech Republic. The work of Jan Komenda is supported by RVO 67985840 and GAČR grant 19-06175J komenda@ipm.cz

inequalities depend on variables, denoted Δ, which take into account how long a transition has been enabled. An edge is labeled by the transition to be fired and an interval which is function of variables Δ and that captures the time spent in the tail node of the edge. The abstraction of TPNs proposed here is also a graph with each edge labeled by the transition and an interval, but this interval is computed a priori to cover the possible enabling durations.

Beyond the differences with the studies from the literature highlighted above, we must stress that our goal in the presented approach is to obtain an abstraction that only accepts time-transition sequences that are firable in the original TPN. More explicitly, let us denote $(a_0, \tau_0), (a_1, \tau_1), \ldots$ a possible time-transition sequence in the TPN, i.e. τ_i, $i \geq 1$ is a possible firing date for a_i after $(a_0, \tau_0), \ldots, (a_{i-1}, \tau_{i-1})$. We propose here to build a finite abstraction that accepts only time-transition sequences, $(a_0, \tau_0), (a_1, \tau_1), \ldots$ which are possible in the TPN (i.e. the time language of the abstraction should be included or equal to the time language of the TPN). To the best of our knowledge, to efficiently obtain an abstraction of a TPN that satisfies this objective is an originality of our approach compared to the literature on the subject, which is motivated by the prospect of being able to use it for control synthesis or for the verification of properties. Indeed, obtaining a sub-approximation of the TPN semantics guarantees that any state accessed in the IWA corresponds to an accessible state in the TPN, which, by extension, guarantees that the verification of properties expressing an accessibility problem in the IWA implies these properties to be satisfied by the TPN.

In summary, this contribution proposes a procedure to derive a deterministic IWA to represent any bounded TPN (considering single-server and strong semantics) so that all the accepted time-transition sequences in the IWA are possible timed sequences of firings in the TPN.

In the next section, Interval Weighted Automata are defined. Time Petri Nets are introduced in Section III and preliminary results are given to describe their time-behavior according to the selected semantics. The proposed representation of a TPN by an IWA is presented and illustrated in Section IV. Conclusions together with hints on future developments are given in Section V.

II. INTERVAL WEIGHTED AUTOMATA

In this section, some necessary concepts on idempotent semirings are briefly recalled, and interval weighted automata (IWA) are then introduced. For more exhaustive presentations, the reader is invited to consult the references BICOQ) and [1].

Definition 1 An *idempotent semiring* is a set $\mathcal{D}$ equipped with two operations, denoted respectively $\oplus$ and $\otimes$. Addition $\oplus$ is commutative, associative, has a zero element ε ($\varepsilon \oplus a = a$ for all $a \in \mathcal{D}$), and is idempotent ($a \oplus a = a$ for each $a \in \mathcal{D}$). Multiplication $\otimes$ is associative, has a unit element e, and distributes over $\oplus$.

Example 1 Let $\mathbb{Q}$ denote the set of rational numbers, $\mathbb{Q}_{max}$ is the set $(\mathbb{Q} \cup \{-\infty\})$ endowed with the maximum that plays the role of addition $\oplus$ and the standard addition playing the role of multiplication $\otimes$, with $e = 0$ and $\varepsilon = -\infty$.

The direct product of semirings simply consists of the Cartesian product of their underlying carrier sets, and it is equipped with componentwise addition and multiplication. We consider here the direct product of semiring $\mathbb{Q}_{max}$ with itself, denoted $\mathbb{Q}_{max}^{max}$.

Definition 2 The idempotent semiring $\mathbb{Q}_{max}^{max}$ is defined by $\langle (\mathbb{Q} \cup \{-\infty\}) \times (\mathbb{Q} \cup \{-\infty\}), \oplus, \otimes \rangle$ with

$$\begin{aligned} [c_1, d_1] \oplus [c_2, d_2] &= [\max(c_1, c_2), \max(d_1, d_2)] \\ [c_1, d_1] \otimes [c_2, d_2] &= [c_1 + c_2, d_1 + d_2] \\ \varepsilon &= [-\infty, -\infty] \\ e &= [0, 0] \end{aligned}$$

We observe that this product of semirings is related to the set of intervals. In particular, product $\otimes$ coincides with the addition of intervals.

To be able to introduce weighted automata, let us recall that if A is an *alphabet* (a finite set of letters), A^* is defined as the set of finite words (or strings) with letters in A. A *word* $w \in A^*$ can be written as a sequence $w = a_1 a_2 \ldots a_p$ with a_1, a_2, …, $a_p \in A$ and p a natural number. The empty word is denoted by λ.

Definition 3 A $\mathcal{D}$-weighted automaton over an alphabet A is a quintuple $G = (Q, A, \alpha, \mu)$, where Q is a finite set of states, $\alpha : Q \to \mathcal{D}$ defines the input weights, and $\mu : A \to \mathcal{D}^{|Q| \times |Q|}$ defines the transition weights.

The *morphism* μ represents the state transitions given by the family of matrices $\mu(a) \in \mathcal{D}^{|Q| \times |Q|}$, $a \in A$. For $q, q' \in Q$, $\mu(a)_{qq'}$ represents the possible activation weights for label a before it can occur for the transition from q to q'. If there is no transition from q to q' labeled by a, $\mu(a)_{qq'} = \varepsilon$. For a string $w = a_1 a_2 \ldots a_p$, we have $\mu(w) = \mu(a_1) \otimes \mu(a_2) \otimes \ldots \otimes \mu(a_p)$. A state $q \in Q$ is said to be an *initial state* iff $\alpha(q) \neq \varepsilon$.

Equivalently, a $\mathcal{D}$-weighted automaton G can be defined by the tuple (Q, A, Q_i, σ, t), in which Q_i denotes the set of *initial* states, $t : Q \times A \times Q \to \mathcal{D}$ is the *transition function*, $\sigma : Q_i \to \mathcal{D}$ is the *initial weights function*:

$$Q_i \triangleq \{q \in Q : \alpha_q \neq \varepsilon\}; \quad \forall q_i \in Q_i, \sigma(q_i) \triangleq \alpha_{q_i};$$
$$\forall q, q' \in Q, t(q, a, q') \triangleq \mu(a)_{qq'}.$$

Interval weighted automata (IWA) are defined here as weighted automata with weights in semiring $\mathbb{Q}_{max}^{max}$. We restrict our attention to *deterministic IWA*, that is IWA for which

- there exists only one initial state,
- $\forall a \in A$, $t(q, a, q') \neq \varepsilon$ and $t(q, a, q'') \neq \varepsilon$ implies $q' = q''$.

An IWA can be represented by a graph (see Example 2) as follows:

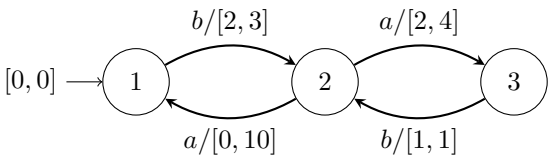

Fig. 1. An interval weighted automaton

- the states $q \in Q$ are represented by nodes;
- there is an arrow from state $q \in Q$ to state $q' \in Q$ whenever there exists $a \in A$ such that $t(q, a, q') \neq \varepsilon$: the arrow is then labeled by $a/t(q, a, q')$.
- an input edge entering a state means this state is an initial state.

Example 2 Figure 1 shows an IWA. We have $A = \{a, b\}$, $Q = \{1, 2, 3\}$, $Q_i = \{1\}$, $\sigma(1) = [0, 0]$, $t(2, a, 1) = [0, 10]$, $t(2, a, 3) = [2, 4]$, $t(1, b, 2) = [2, 3]$, $t(3, b, 2) = [1, 1]$, and $t(q, a, q') = \varepsilon$ all the other tuples (q, a, q').

III. TIME PETRI NETS

This section recalls the definition of Time Petri Nets (TPN) and specifies the selected semantics. Preliminary results are also stated in order to describe the behavior of such TPN.

A. Definitions

Definition 4 A Petri net is defined as a quintuple :

$$(P, T, Pre, Post, M_0)$$

- P is a finite set of *places*,
- T is a finite set of *transitions*,
- $Pre : T \to \mathbb{N}^P$ and $Post : T \to \mathbb{N}^P$ are the backward and the forward *incidence functions*, and
- $M_0 \in \mathbb{N}^P$ is the *initial marking*.

A marking M associates, to each place, a number of tokens. For all $a \in T$, $Pre(a) \in \mathbb{N}^P$ establishes the number of tokens in each place required for a to be fired. Transition a is said enabled at marking M, iff $M \geq Pre(a)$. Notation $En(M)$ refers to the set of transitions enabled at M. If transition a is fired, $Post(a) \in \mathbb{N}^P$ corresponds to the number of tokens released by a to each of its output places.

A TPN is a Petri net with time intervals associated with transitions.

Definition 5 A T-time Petri net is a tuple

$$(P, T, Pre, Post, M_0, Is)$$

in which $(P, T, Pre, Post, M_0)$ is a Petri net and $Is : T \to \mathbb{Q}_{\geq 0} \times (\mathbb{Q}_{\geq 0} \cup \{\infty\})$ is the static firing interval function. In this paper, we denote $\underline{I}$ (resp. $\overline{I}$), the lower bound (resp. the upper bound) of an interval I. It is assumed that $\underline{I} \leq \overline{I}$.

In a TPN, two transitions are said to be *non conflicting*, or *parallel*, if they have no common input place, i.e. a and b are parallel if $Pre(a) \cap Pre(b) = \emptyset$. Otherwise, both transitions are said to be *conflicting*.

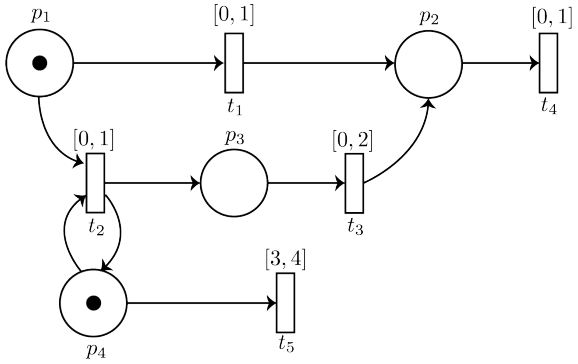

Fig. 2. Example of a Time Petri net

A TPN can be represented by a graph (see Example 3) as follows:

- a node corresponds to a place $p \in P$ (circle) or a transition $a \in T$ (box),
- tokens into places symbolize available resources,
- an edge from place to transition specifies an input place for that transition, defined by the backward incidence function,
- an edge from transition to place specifies an output place for that transition, defined by the forward incidence function.

Example 3 Fig. 2 shows a TPN $(P, T, Pre, Post, M_0, Is)$ where the set of places is $P = \{p_1, p_2, p_3, p_4\}$ and the set of transitions is $T = \{t_1, t_2, t_3, t_4, t_5\}$. The incidence functions for each transition are :

- $Pre(t_1) = (1000)$, $Post(t_1) = (0100)$,
- $Pre(t_2) = (1001)$, $Post(t_2) = (0011)$,
- $Pre(t_3) = (0010)$, $Post(t_3) = (0100)$,
- $Pre(t_4) = (0100)$, $Post(t_4) = (0000)$,
- $Pre(t_5) = (0001)$, $Post(t_5) = (0000)$.

The static firing interval function is defined by: $Is(t_1) = [0, 1]$, $Is(t_2) = [0, 1]$, $Is(t_3) = [0, 2]$, $Is(t_4) = [0, 1]$ and $Is(t_5) = [3, 4]$. The initial marking is $M_0 = (1001)$.

B. Semantics

In this paper, as in [12], the single-server and strong semantics are considered for bounded TPNs. When a transition is multi-enabled, only one enabling instance is considered (single-server semantics). The strong semantics forces an enabled transition to fire if the elapsed time since its last enabling reaches its maximum static firing time, i.e. the firing of this transition disables other potentially enabled transitions that are in conflict with it, but whose firing is not urgent (the upper bound of their firing interval is not yet reached). Furthermore, let us repeat that the static firing interval function is limited to $Is : T \to \mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$.

The behavior of a TPN is characterized by its states and the transitions between states.

First, a state of a TPN is a pair denoted (M, E), where M is a marking, and E reflects temporal information on each of the enabled transitions. In the literature, E can

be formalized in several different ways, see [21]. In this paper, E is defined as a vector of intervals associated with transitions. A component E_a is an interval capturing the possible times elapsed since the last enabling of $a \in T$. This way, E can be seen as a vector of clocks. By convention, the notation $E_a = \ddagger$ means the transition a is not enabled. Hence, the initial state is (M_0, E_0) where M_0 is defined in Def. 4, and E_0 is defined by:

$$\forall a \in T,\ E_{0,a} = \begin{cases} [0,0], & \text{if } a \in En(M_0), \\ \ddagger, & \text{otherwise.} \end{cases} \quad (1)$$

Secondly, a TPN may evolve according to discrete and continuous transitions. A discrete transition occurs when a transition a is fired and is denoted

$$(M, E) \xrightarrow{a} (M', E'). \quad (2)$$

The following conditions must be satisfied for such a discrete transition:

(i) $a \in T$ and $a \in En(M)$,
(ii) $M' = M - Pre(a) + Post(a)$ (the standard marking transformation),
(iii) $\forall b \in En(M')$, $E'_b = [0,0]$ if b is newly enabled, $E'_b = E_b$ otherwise.

For (iii), let us give more precision on newly enabled transitions. Considering discrete transition (2), a transition a' is said to be newly enabled by marking M' if a' is enabled by M' and it was not enabled by intermediate marking M'' obtained by removing the tokens from the upstream places of a involved in the firing of a (while the tokens are not yet added to the downstream places of a). For a multi-enabled transition, the transition is considered as newly enabled if it is still enabled after its firing, or the firing of a conflicting transition.

A continuous transition reflects the progression of time and is denoted

$$(M, E) \xrightarrow{\theta} (M, E'), \quad (3)$$

with the conditions:

(iv) $\theta \in \mathbb{Q}_{\geq 0}$,
(v) $\forall b \in En(M)$, $\overline{E_b} + \theta \leq \overline{Is}(b)$
(vi) $\forall b \in En(M)$, $E'_b = E_b + \theta$

C. Aggregated transition

In this contribution, we propose to describe the behavior by transitions which aggregate both discrete and continuous transitions. More explicitly, such a transition is denoted

$$(M, E) \xrightarrow{a/D_a} (M', E'), \quad (4)$$

in which conditions (i)-(vi) must all be satisfied. In this transition, $d_a \in \mathbb{Q}_{\geq 0}$, $d_a \in D_a$ can be interpreted as the duration that may elapse since its last enabling before transition a is fired. In the following lemmas, we identify several constraints on the value for d_a that come from the TPN semantics.

It is essential to specify that we seek here to define the widest possible evolution domain for d_a such that the TPN semantics is respected for sure. Hence, we define a lower and upper bound for the value of d_a, and doing so, D_a is defined as an interval in $\mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$.

This is done with the objective of obtaining in the next section an abstraction of the time behavior of the TPN for which each accepted sequence includes only time-transition sequences $(a_0, d_{a_0}), (a_1, d_{a_1}), \ldots$ that are firable in the TPN, i.e. where each d_{a_i}, $i \in \mathbb{N}$, of the sequence is a possible firing date for a_i. In other words, we are looking for an abstraction whose time language is included or equal to the time language of the TPN. Based only on conditions (i)-(vi), the SCGs proposed in [12] may lead to so-called *firing domains* which accept time-transition sequences that are not firable in the corresponding TPN. As illustrated in Section IV-C, the approach in [19] and [20] may also lead to an abstraction which accepts time-transition sequences that are not firable in the TPN. And to our knowledge, to efficiently obtain an abstraction of a TPN that satisfies this objective is an originality of our approach.

Lemma 1 *From state (M, E), a transition $a \in En(M)$* can effectively be fired *(i.e. transition a can be fired while respecting the TPN semantics for sure) if $\exists d_a \in \mathbb{Q}_{\geq 0}$ satisfying the following inequalities*

$$d_a \geq \max(0, \underline{Is}(a) - \underline{E_a}) \quad (5)$$
$$d_a \leq \overline{Is}(b) - \overline{E_b}\ , \quad \forall b \in En(M) \quad (6)$$

For $a \in En(M)$, we define the set $Para(M, a)$ of enabled transitions that are in parallel (or nonconflicting) with a:

$$Para(M, a) \triangleq \{b \in En(M) : Pre(a) \cap Pre(b) = \emptyset\}$$

The following lemma makes explicit a constraint on d_a, original to the best of our knowledge, which derives from the order of occurrence between the firings of parallel transitions. Indeed, while defining transition (4) it is implicitly considered that any parallel transition b is to be fired after a. This aspect is not explicitly taken into account in conditions (i)-(vi) and we believe that this omission leads to so-called firing domains in [12] for SCGs, so that these SCGs may accept time-transition sequences that are not firable in the corresponding TPN.

Lemma 2 *Let a be a transition that can effectively be fired from (M, E) as defined in Lemma 1, $d_a \in \mathbb{Q}_{\geq 0}$ must satisfy the following inequality so that the TPN semantics can be respected for parallel transitions, i.e. $\forall b \in Para(M, a)$:*

$$d_a \leq \overline{Is}(b) - \overline{E_b} - \max(0, (\underline{Is}(b) - \underline{E_b}) - (\underline{Is}(a) - \underline{E_a})). \quad (7)$$

IV. REPRESENTATION OF TPN BY IWA

A. Procedure

The proposed construction for a deterministic IWA $G = (Q, A, Q_i, \rho, t)$ from a bounded TPN $\mathcal{P} = (P, T, Pre, Post, M_0, I_s)$ is based on the behavior of the TPN described in sections III-B and III-C.

The alphabet A of G is defined as the set of transitions T in $\mathcal{P}$.

The initial state of G is defined as $Q_i = (M_0, E_0)$ (where E_0 is given by (1)) with initial weight $[0, 0]$.

The set of states Q and the transition function t are now defined in an iterative way. Let us consider that a state (M, E) has been defined in Q, with M a marking of the TPN and E a vector of intervals where E_a captures the possible times elapsed since the last enabling of $a \in T$ (with the convention $E_a = \ddagger$ if $a \notin En(M)$). For all transition a that can effectively be fired (as stated in Lemma 1), a state (M', E') and a transition in G are defined by

$$t((M, E), a, (M', E')) = D_a \tag{8}$$

where

- M' is given by the standard marking transformation (see (ii)),
- $D_a \in \mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$, $D_a = [\underline{D_a}, \overline{D_a}]$ is defined in (4) and is computed as the largest solution (according to the inclusion-order of intervals) of the following set of inequalities from Lemmas 1 and 2:

$$\underline{D_a} \geq \max(0, \underline{Is}(a) - \underline{E_a}) \tag{9}$$
$$\overline{D_a} \leq \overline{Is}(b) - \underline{Is}(b), \forall b \in En(M) \tag{10}$$
$$\overline{D_a} \leq \overline{Is}(b) - \overline{E_b} - \max(0, (\underline{Is}(b) - \underline{E_b}) - (\underline{Is}(a) - \underline{E_a})), \quad \forall b \in Para(M, a) \tag{11}$$

- E' is the updated vector given for all $b \in T$ by

$$E'_b = \begin{cases} \ddagger, & \text{if } b \text{ is disabled by } M', \\ [0, 0], & \text{if } b \text{ is newly enabled by } M', \\ E_b \otimes D_a, & \text{otherwise.} \end{cases} \tag{12}$$

To conclude with the description of the procedure, two states (M, E) and (M', E') in G are considered as equivalent if $M = M'$ and $E = E'$ (just as for state-classes in [12] and [22]), and they can be merged in G.

B. Properties

We now mention important characteristics of the IWA obtained through the procedure introduced in Section IV-A.

Proposition 1 *Let $\mathcal{P}$ be a bounded TPN. The IWA G built according to the proposed procedure is deterministic.*

Proposition 2 *Let P be a bounded TPN. The IWA G built according to the procedure proposed is finite, i.e. its set of states Q is finite.*

Proposition 3 *Let $\mathcal{P}$ be a bounded TPN. Any time-transition sequence accepted in the IWA G built according to the procedure proposed is a possible timed sequence of firings in the TPN.*

C. Illustrations

Let us first illustrate the application of the proposed procedure in section IV-A. Below, we detail several of the steps involved in constructing the IWA of Fig. 4 as an abstraction of the TPN in Fig. 3.

The initial state of the IWA $(M_0, E_0) = (21, ([0, 0][0, 0]))$ is defined with the initial marking of the TPN and considering that the transitions enabled by this marking are newly enabled.

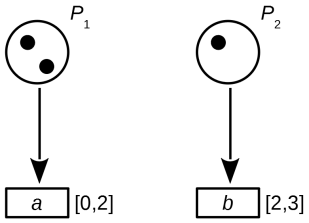

Fig. 3. A bounded TPN

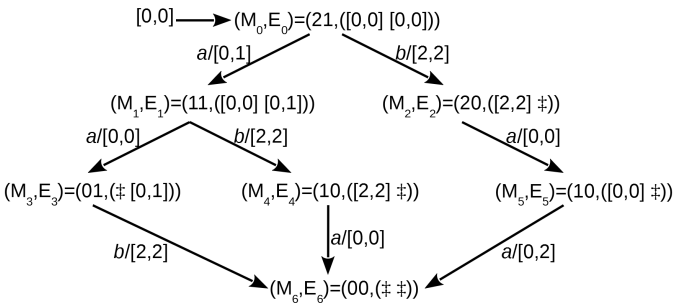

Fig. 4. IWA obtained with the proposed procedure for the bounded TPN in Fig. 3

At state (M_0, E_0), the set of enabled transition is $En(M_0) = \{t_1, t_2\}$, and applying Lemma 1, we obtain that both transitions a and b can be effectively fired. For the transition according to a, (9)-(11) can be written as

$$\begin{cases} \underline{D_a} & \geq & \max(0, 0 - 0) & \text{(applying (9))} \\ \\ \overline{D_a} & \leq & 2 - 0 & \text{(applying (10) for } a \in En(M)) \\ \overline{D_a} & \leq & 3 - 0 & \text{(applying (10) for } b \in En(M)) \\ \\ \overline{D_a} & \leq & 3 - max(0, 2 - 0) & \text{(applying (11))} \end{cases}$$

which leads to $D_a = [0, 1]$. This weight may seem surprising whereas the enabling time of a in the TPN is $[0, 2]$ but is justified by the interpretation and purpose of our abstraction. Indeed, transition b is enabled in parallel with a and, therefore, transition b is to be fired after a. However, if we accept $[0, 2]$ as the weight associated with a, we should have a weight associated with b at least equal to 2 (to respect its minimum enabling time) and the weight associated with sequence ab should then include $[0, 2] + [2, 2] = [2, 4]$. This would result in a timed sequence that does not correspond to a possible firing sequence in the TPN, and one can be convinced that $[0, 1]$ is the largest possible interval to weight a such that timed sequence ab in the IWA remains a possible firing sequence in the TPN. Let us note also that since we consider a single-server semantics, transition a is considered as newly enabled by M_1, that is by the second token in P_1 after the first firing of a. This is why we have $E_{1,a} = [0, 0]$. The construction (not detailed here) of the other states and transitions of the IWA shown in figure 4 proceeds in the same way as presented in the Procedure of the section IV-A.

Let us now consider the TPN depicted in Fig. 2. Note that this example is considered in [20] to illustrate the construction of a Modified State-Class Graph (MSCG). The

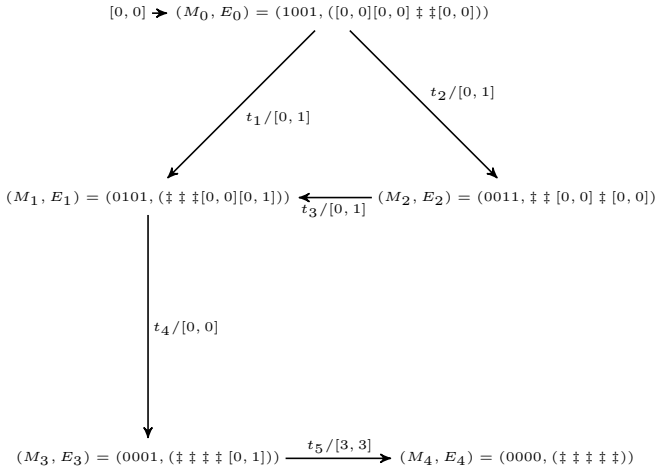

Fig. 5. IWA obtained with the proposed procedure for the bounded TPN in Fig. 2

application of our procedure leads to the IWA shown in Fig. 5. The obtained IWA may differ significantly from other abstractions proposed in the literature. In particular, the MSCG obtained in [20] accepts from the initial state the sequence, t_1t_5 whereas it is not firable in the TPN. Indeed, after the firing of t_1, t_5 is still enabled and t_4 is newly enabled. At least 2 units of time must elapse before t_5 is firable. However, no more than 1 unit of time may elapse before the strong semantic forces t_4 to fire. So, t_4 is necessarily fired before t_5. Fig. 5 shows that the IWA obtained with the proposed procedure does not accept the sequence t_1t_5.

V. CONCLUSION

A procedure has been proposed to derive a deterministic IWA representing any bounded TPN subject to strong and single-server semantics. This approach helps to characterise the expressiveness of weighted automata with respect to other formalisms for DES. Moreover, it provides an abstraction in which any timed sequence is a possible firing sequence in the TPN.

This will allow future work to address algebraically the verification of guaranteed properties. Note that since the resulting abstraction (IWA) is a deterministic WA, this enables efficient verification of properties and computation of controllers. However, since it is a sub-approximation of the TPN, it will be useful for some particular properties based on reachability (accessibility) and minimal required behavior. This is why it is interesting to provide also a non deterministic IWA abstraction, which will cover the behavior of the TPN exactly, but the verification of properties and supervisory control are no longer efficient unless we choose a finite horizon.

REFERENCES

[1] M. Droste, W. Kuich, and H. Vogler, *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 2009.

[2] S. Gaubert, "Performance Evaluation of (max,+) Automata," *IEEE TAC*, vol. 40, no. 12, pp. 2014–2025, 1995.

[3] J. Komenda, S. Lahaye, and J. Boimond, "Supervisory control of (max,+) automata: A behavioral approach," *Discrete Event Dyn Syst*, vol. 19, no. 525, 2009.

[4] R. Su, J. van Schuppen, and J. Rooda, "The synthesis of time optimal supervisors by using heaps-of-pieces," *IEEE TAC*, vol. 57, no. 1, pp. 105–118, 2012.

[5] S. Gaubert and J. Mairesse, "Modeling and Analysis of Timed Petri Nets using Heaps of Pieces." *IEEE TAC*, vol. 44, no. 4, pp. 683–698, 1999.

[6] S. Lahaye, J. Komenda, and J.-L. Boimond, "Compositions of (max, +) automata," *Discrete Event Dynamic Systems*, vol. 25, pp. 323–344, June 2015 2015.

[7] L. Triska and T. Moor, "Behaviour equivalent max-plus automata for a class of timed petri nets," *IFAC PapersOnLine*, vol. 53-4, pp. 75–82, 2020.

[8] J. Komenda, S. Lahaye, and J.-L. Boimond, "Determinization of timed Petri nets behaviors," *Discrete Event Dynamic Systems*, pp. 1–25, 2015.

[9] J. Komenda, A. Lai, J. G. Soto, S. Lahaye, and J. louis Boimond, "Modeling of safe time petri nets by interval weighted automata," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 187–192, 2020.

[10] M. Boyer and O. Roux, "On the compared expressiveness of arc, place and transition time petri nets," *Fundamenta Informaticae*, vol. 88, pp. 225–249, 01 2008.

[11] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time petri nets," *IEEE Transactions on Software Engineering*, vol. 17, no. 3, pp. 259–273, 1991.

[12] B. Berthomieu and F. Vernadat, "State space abstractions for time petri nets," in *Handbook of Real-Time and Embedded Systems*, 2007.

[13] T. Yoneda and H. Ryuba, "On the compared expressiveness of arc, place and transition time petri nets," *IEICE Transactions on Information and Systems*, vol. 1-2, pp. 123–133, 1998.

[14] B. Berthomieu and F. Vernadat, "State class constructions for branching analysis of time petri nets," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 442–457.

[15] G. Gardey, O. H. Roux, and O. F. Roux, "State space computation and analysis of time petri nets," *Theory and Practice of Logic Programming*, vol. 6, no. 3, p. 301–320, 2006.

[16] D. Lime and O. H. Roux, "Model checking of Time Petri Nets using the State Class Timed Automaton," *Discrete Event Dynamic Systems*, vol. 16, no. 2, pp. 179–205, 2006.

[17] F. Cassez and O. H. Roux, "Structural translation from time Petri nets to timed automata," *Journal of Systems and Software*, vol. 29, no. 1, pp. 1456–1468, 2006.

[18] B. Berard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux, "The Expressive Power of Time Petri Nets," *Theoretical Computer Science*, vol. 474, pp. 1–20, 2013.

[19] F. Basile, M. P. Cabasino, and C. Seatzu, "State estimation and fault diagnosis of labeled time petri net systems with unobservable transitions," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 997–1009, 2015.

[20] Z. He, Z. Li, A. Giua, F. Basile, and C. Seatzu, "Some remarks on "state estimation and fault diagnosis of labeled time petri net systems with unobservable transitions"," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 5253–5259, 2019.

[21] H. Boucheneb and K. Barkaoui, "Relevant timed schedules/clock vectors for constructing time petri net rachability graphs," *Discrete Event Dyn Syst*, vol. 21, pp. 171–204, 2011.

[22] B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time petri nets," in *IFIP Congress Series*, 1983.