Discrete Event Dynamic Systems manuscript No.
(will be inserted by the editor)

Compositions of (max,+) automata

Sébastien Lahaye · Jan Komenda ·
Jean-Louis Boimond

Received: date / Accepted: date

Abstract This paper presents a compositional modeling approach by means of (max,+) automata. The motivation is to be able to model a complex discrete event system by composing sub-models representing its elementary parts.
A direct modeling of safe timed Petri nets using (max,+) automata is first introduced. Based on this result, two types of synchronous product of (max,+) automata are proposed to model safe timed Petri nets obtained by merging places and/or transitions in subnets. An asynchronous product is finally proposed to represent particular bounded timed Petri nets.

1 Introduction

Timed Petri nets have been introduced in [20] as an important class of timed discrete event systems (TDES) with deterministic timing of places and/or transitions. However, automata based models have the advantage that durations of sequences of transitions can be computed in a much easier (algebraic) way. Corresponding class of timed automata with deterministic timing of transitions is known as (max,+) automata [6]. These automata with weights (multiplicities) in the idempotent semiring $(\mathbb{R} \cup \{-\infty\}, \max, +)$ form an important class of timed automata. Their modeling power in terms of timed Petri net is

Research supported by the GAČR grants no. P103/11/0517 and by RVO: 67985840.

The previous version of this work was presented at WODES 2012, Guadalajara, Mexico [14].

Sébastien Lahaye and Jean-Louis Boimond
LUNAM Université, LISA, Angers, France
E-mail: [sebastien.lahaye, jean-louis.boimond]@univ-angers.fr

Jan Komenda
Institute of Mathematics - Brno Branch, Czech Academy of Sciences, Czech Republic.
E-mail: komenda@ipm.cz

studied in [8], where it is shown that the behavior of any safe[1] timed Petri net can be expressed by a so-called heap automaton, which is a special type of (max,+) automaton. (Max,+) automata have been applied in particular to performance evaluation [6], scheduling [9] and control [13, 22] problems for a large class of TDES.

This paper is focused on the modeling of TDES using (max,+) automata. As a first result, a direct transformation is proposed to describe safe timed Petri nets by (max,+) automata, whereas a two-steps procedure is specified for this in [8] (using a heap of pieces as intermediate model). Based on this result, two types of synchronous compositions for (max,+) automata are proposed in order to model merging of places and/or transitions between the corresponding Petri nets. These contributions can be seen as natural extensions of results in [8] towards compositional modeling. In fact, the (max,+) automaton of a complex and modularly structured TDES can then be obtained as composition(s) of (max,+) automata corresponding to subsystems.

These compositions lead to nondeterministic (max,+) automata, but these are defined over the standard union alphabet and moreover, their number of states is considerably smaller than the number of states in standard synchronous products of Boolean automata (cf. [4]) or in other approaches to synchronous product of (max,+) automata [3, 22]. Indeed, in all these products the number of states is bounded by the product of the number of states of the component automata, while in our case it is equal to the sum of the number of states. This means that description of timing phenomena does not suffer from the combinatorial state space explosion. Nevertheless, if we want to model the logical phenomena, the refinement by (logical) product is needed (the resulting automaton is obtained by the tensor product with Boolean automaton of the composition of the corresponding Boolean automata).

Note that we have already sketched these results in [15] with restrictions to safe timed state graphs and on the transition structure. Let us also evoke that a synchronous composition of (max,+) automata has been proposed in [12], where the product automaton is defined as a deterministic (max,+) automaton, but over an extended alphabet that is composed of tuples of strings of events that can be executed in parallel. Since the resulting (max,+) automaton is deterministic, this product finds its application in decentralized supervisory control of (max,+) automata, while the synchronous products proposed in this paper is better suited for verification and performance evaluation.

We also propose an asynchronous product of (max,+) automata that have isomorphic state-transition structure (the same number of states and identical morphism matrices). It is shown to be suitable to represent m-bounded timed state graphs (with $m > 1$).

The paper is organised as follows. In the following section, preliminaries necessary to understand the paper are briefly recalled. In Section 3 we propose a new direct modeling of timed Petri nets by (max,+) automata. Section 4 is

[1] The marking of any place is bounded by 1.

dedicated to the definition of synchronous products of (max,+) automata together with their applications. In Section 5, asynchronous product of (max,+) automata is introduced, and we discuss how it can used to model bounded timed Petri nets. Finally, in Section 6, concluding remarks with hints on future extensions of our modeling approach are given.

2 Preliminaries

The necessary concepts and results about idempotent semirings, (max,+) automata and Petri nets are briefly recalled in this section. For some more exhaustive presentations, the reader is invited to consult the references [1], [6] and [5].

Definition 1 (dioid) A *dioid* is a *semiring* in which the addition $\oplus$ is idempotent. The addition (resp, the multiplication $\otimes$) has a null element ε (resp., identity element e).

Example 1 The set $(\mathbb{R} \cup \{-\infty\})$ with the maximum playing the role of addition and the conventional addition playing the role of multiplication is a dioid, denoted $\mathbb{R}_{\max}$ (and usually called *(max,+) algebra*), with $e = 0$ and $\varepsilon = -\infty$. The set of $n \times n$ matrices with coefficients in $\mathbb{R}_{\max}$, endowed with the matrix addition and multiplication conventionally defined from $\oplus$ and $\otimes$, is also a dioid, denoted $\mathbb{R}_{\max}^{n\times n}$. The neutral elements for the addition (resp. multiplication) is the matrix denoted ε_n (resp. I_n) and exclusively composed of ε $(= -\infty)$ (resp. composed of e $(= 0)$ on the diagonal and ε $(= -\infty)$ elsewhere). Note that, to be able to multiply a $1 \times n$ vector with a $n \times n$ matrix, this vector should be embedded in $\mathbb{R}_{\max}^{n\times n}$ by adding $n-1$ lines full of ε. To ligthen the presentation, this construction is often omitted in the following (without affecting the results), and the coefficients equal to ε in the matrices will be replaced by '$\cdot$'.

Example 2 If A is a finite set (*alphabet*), the free monoid on A is defined as the set A^* of finite words with letters in A. A *word* $w \in A^*$ can be written as a sequence $w = a_1a_2 \ldots a_p$ with a_1, a_2, ..., $a_p \in A$ and p a natural number. *Formal languages* are subsets of the free monoid A^*. The set of formal languages, with the union of languages playing the role of addition and concatenation of languages playing the role of multiplication, is a dioid, denoted $(2^{A^*}, \cup, .)$. The zero language is $\varepsilon = \{\}$, the unit language is denoted $e = \{\epsilon\}$ where ϵ is the empty (zero length) string. The *prefix closure* of a language corresponds to the set of prefixes of the words in the language.

Automata with multiplicities in $\mathbb{R}_{\max}$ semiring are called (max,+) automata.

Definition 2 ((max,+) automaton) A *(max,+) automaton* G is a quadruple (Q, A, α, μ) where

- Q and A are finite sets of states and of events ;
- $\alpha \in \mathbb{R}_{\max}^{1\times|Q|}$ is such that $\alpha_q = \varepsilon$ or $\alpha_q = e$, and a state q is said to be an initial state when $\alpha_q = e$;
- $\mu : A^* \to \mathbb{R}_{\max}^{|Q|\times|Q|}$ is a morphism specified by the family of matrices $\mu(a) \in \mathbb{R}_{\max}^{|Q|\times|Q|}$, $a \in A$, and for a string $w = a_1 \ldots a_n$, we have

  $$\mu(w) = \mu(a_1 \ldots a_n) = \mu(a_1) \ldots \mu(a_n),$$

  where the matrix multiplication involved here, is the one of $\mathbb{R}_{\max}^{|Q|\times|Q|}$. A coefficient $[\mu(a)]_{qq'} \neq \varepsilon$ means that, from state q, the occurrence of event a causes a state transition to state q'.

This definition is slightly different from that in [6] where initial and final delays are considered. In the present paper, we restrict our attention to (max,+) automata in which the initial delays (that is the coefficients in α different from ε) are all equal to $e = 0$[2]. In addition, the vector of final delays is not considered, hence all states can be thought of as final states (as it is the case for heap automata [8]).

Example 3 Figure 1 illustrates the graphical representation of a (max,+) automaton:

- the nodes correspond to states $q \in Q$;
- an arrow exists from state $q \in Q$ to state q' if there exists an event $a \in A$ such that $[\mu(a)]_{qq'} \neq \varepsilon$: the arrow is labelled by $a/[\mu(a)]_{qq'}$ and represents the state transition when event a occurs. The value of $[\mu(a)]_{qq'}$ is interpreted as the duration associated to event a (namely, the activation time of event a before it can occur) ;
- an input edge symbolizes an initial state.

For this example, we have $Q = \{q_1, q_2, q_3, q_4\}$, $A = \{a, b\}$, and

$$\alpha = \begin{pmatrix} e & \cdot & e & e \end{pmatrix},\ \mu(a) = \begin{pmatrix} \cdot & 2 & \cdot & 2 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 \\ \cdot & 3 & \cdot & 3 \end{pmatrix}, \mu(b) = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ 2 & \cdot & 2 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \end{pmatrix}.$$

In order to describe the dynamic evolution of a (max,+) automaton, vector $x_G(w) \in \mathbb{R}_{\max}^{1\times|Q|}$ for $w \in A^*$ is defined by

$$x_G(w) = \alpha\mu(w)\ . \tag{1}$$

An element $[x_G(w)]_q$ is interpreted as the date at which state q is reached at the conclusion of sequence w starting from an initial state (with the convention

[2] This assumption is without loss of generality since an automaton with initial delays can always be transformed into an equivalent automaton with null initial delays by adding new states and by considering these delays as state transitions durations associated to new fictive initial events.

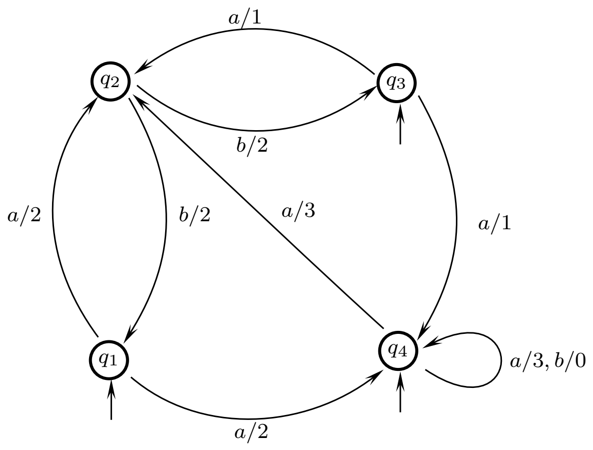

Fig. 1 A (max,+) automaton G_1.

that $[x_G(w)]_q = \varepsilon$ if state q is not reached from an initial state using the input sequence w). The elements of x_G are *generalized daters*, and we have

$$\begin{cases} x_G(\epsilon) = \alpha, \\ x_G(wa) = x_G(w)\mu(a). \end{cases} \tag{2}$$

Vector x_G can be written equivalently as a vector of formal series:

$$x_G = \bigoplus_{w \in \Sigma^*} x_G(w)w,$$

which is sometimes referred to as the behavior of G. The underlying language of a (max,+) automaton corresponds to the support of its behavior series. Note that, as all states are assumed to be final, this language is always prefix-closed.

Definition 3 (Petri net) A *Petri net* $\mathcal{G}$ is a 4-tuple $(\mathcal{P}, \mathcal{T}, \mathcal{F}, M)$, in which $\mathcal{P}$ is a finite set of places, $\mathcal{T}$ is a finite set of transitions, $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a relation between places and transitions , $M : \mathcal{P} \to \mathbb{N}$ defines the *initial marking* of places.

The marking evolves according to the following rules:

1. Transition a is *enabled* at M if there exists at least one token in each of its input places.
2. An enabled transition a can fire. The firing of a transforms M into M' by removing one token from each of the input places and adding one token in each of the output places of a.

We say that a word $w = a_1 a_2 \dots a_n \in \mathcal{T}^*$ is a *firing sequence* starting from marking M_0 if there is a sequence of markings $M_1 M_2 \dots M_n$ such that transition a_i is enabled at M_{i-1} and its firing transforms M_{i-1} into M_i. We call *language* of the Petri net the set $L \subset \mathcal{T}^*$ of firing sequences starting from initial marking M_0.

A Petri net is said to be *safe* (resp. *m-bounded*) if for all accessible marking each place contains at most one token (resp. at most m tokens).

For transition $a \in \mathcal{T}$, ${}^\bullet a$ (resp. $a^\bullet$) denotes the set of its input (resp. output) places. If for all the transitions these sets are singletons, then the Petri net is a *state graph*.

We restrict our attention to *free-labelled* Petri nets in which each transition (resp., each place) has a single label and there are no two transitions (resp., places) with the same label.

We consider timed Petri nets in which

– a finite firing duration τ_a is associated with each transition a: τ_a is the minimal time that must elapse, starting from the time at which a is enabled, until this transition can fire;
– a finite sojourn duration τ_p is associated with each place p: τ_p is the minimal time that must elapse, starting from the time at which a token enters p, until this token becomes available for enabling downstream transitions.

Several assumptions on the functioning of Petri nets are considered hereafter:

– a token from the initial marking is supposed to have arrived in the Petri net at time instant 0;
– if a place has several output transitions, then for each token in this place it needs to be decided which transition is to fire (that is, in case of a *conflict*). In the present work, all the logically feasible choices are considered for the decision (*preselection policy*, in contrast to models for which the decision is rather based on time considerations);
– when a transition is to be fired, then it is fired as soon as possible.

Example 4 A Petri net is usually represented by a bipartite directed graph as in Figure 2. The two types of nodes are places in $\mathcal{P}$ and transitions in $\mathcal{T}$ represented respectively by circles and bars with the associated labels and durations. An element of $\mathcal{F}$ is displayed by an arrow from a place to a transition or from a transition to a place. The marking is figured out by $M(q)$ tokens in place q. Petri net $\mathcal{G}_1$ is safe.

For Petri nets, we define $x_{\mathcal{G}}(w) \in \mathbb{R}_{\max}^{1\times|\mathcal{P}|}$, the vector of variables associated with places $q \in \mathcal{P}$ and function of firing sequence $w \in \mathcal{T}^*$ by

$$[x_{\mathcal{G}}(w)]_q = \begin{cases} \text{instant at which the last token arrived in } q \text{ after } w \\ \qquad \text{(assuming that it is still contained in } q\text{)}, \\ \varepsilon \text{ if } q \text{ does not contain any token.} \end{cases} \tag{3}$$

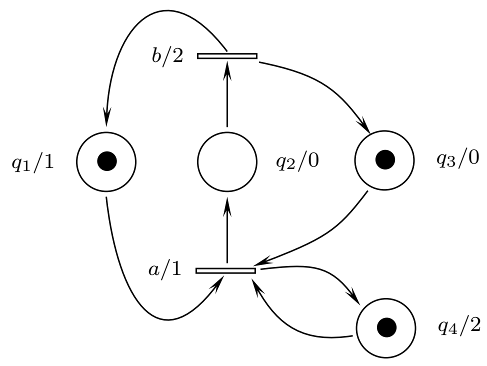

Fig. 2 A Petri net $\mathcal{G}_1$.

3 Direct description of safe timed Petri nets by (max, +) automata

In [8], Stéphane Gaubert and Jean Mairesse have shown that the behavior of any safe timed Petri net can be modeled by a (max,+) automaton. Their approach consists of two steps: a heap representation for such Petri nets is first proposed, and a (max,+) automaton is then derived from the heap model. The following proposition specifies how to directly derive a (max,+) automaton representing a safe timed Petri net.

Proposition 1 *Let $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M)$ be a safe timed Petri Net. A (max,+) automaton $G = (Q, A, \alpha, \mu)$ is derived from $\mathcal{G}$ as follows:*

1. $Q = \mathcal{P}$,
2. $A = \mathcal{T}$,
3. $\forall q \in Q$,
$$\alpha_q = \begin{cases} e & \text{if } M_q = 1, \\ \varepsilon & \text{otherwise.} \end{cases}$$
4. $\forall q, q' \in Q$, $\forall a \in A$,
$$[\mu(a)]_{qq'} = \begin{cases} \tau_a + \tau_q & \text{if } q \in {}^\bullet a \text{ and } q' \in a^\bullet, \\ e & \text{if } q = q' \text{ and } q \notin {}^\bullet a \cup a^\bullet, \\ \varepsilon & \text{otherwise.} \end{cases}$$

We have
$$x_G(w) = x_{\mathcal{G}}(w) \ , \tag{4}$$
for all $w \in \mathcal{T}^*$ a possible firing sequence in $\mathcal{G}$ (i.e., w belongs to the language of $\mathcal{G}$).

Proof For the empty string $w = \epsilon$ we have by construction
$$[x_G(\epsilon)]_q = \alpha_q = \begin{cases} e & \text{if } M_q = 1, \\ \varepsilon & \text{otherwise,} \end{cases} = [x_{\mathcal{G}}(\epsilon)]_q,$$

since we consider that initial tokens arrived at time instant $e = 0$. Let us assume that equality

$$x_{\mathcal{G}}(w) = x_G(w),$$

is true at the conclusion of firing sequence $w \in \mathcal{T}^*$ in the language of $\mathcal{G}$. We check that $x_{\mathcal{G}}(wa) = x_G(wa)$, $\forall a \in \mathcal{T}$ such that wa belongs to the language of $\mathcal{G}$. According to (2), we have

$$\forall q' \in \mathcal{P},\ [x_G(wa)]_{q'} = [x_G(w)\mu(a)]_{q'} = \bigoplus_{q \in Q} [x_G(w)]_q \otimes [\mu(a)]_{qq'}.$$

We consider the following cases for q'.

- If $q' \in a^\bullet$, then we have

$$\begin{aligned} [x_G(wa)]_{q'} &= \max_{q \in {}^\bullet a}([x_G(w)]_q + \tau_q + \tau_a) && \text{(according to the def. of } \mu) \\ &= \max_{q \in {}^\bullet a}([x_{\mathcal{G}}(w)]_q + \tau_q + \tau_a) && \text{(by assumption)} \\ &= [x_{\mathcal{G}}(wa)]_{q'} \end{aligned}$$

  The last equality models that transition a is enabled as soon as a token is available in each of the places in ${}^\bullet a$, that is at time instant $\max_{q \in {}^\bullet a}([x_{\mathcal{G}}(w)]_q + t_q)$. It is then fired and a token enters $q' \in a^\bullet$ at $\max_{q \in {}^\bullet a}([x_{\mathcal{G}}(w)]_q + \tau_q + \tau_a)$.
- If $q' \notin a^\bullet$ but $q' \in {}^\bullet a$, then we have

$$[x_G(wa)]_{q'} = \varepsilon \quad \text{since} \quad [\mu(a)]_{qq'} = \varepsilon, \forall q.$$

  We can claim that $[x_G(wa)]_{q'} = [x_{\mathcal{G}}(wa)]_{q'}$ because, when transition a is fired, any input place (which isn't also an output place of a) is then empty.
- Finally, if $q' \notin {}^\bullet a \cup a^\bullet$, then we have

$$[x_G(wa)]_{q'} = [x_G(w)]_{q'} \quad \text{since} \quad [\mu(a)]_{q'q'} = e \text{ and } [\mu(a)]_{qq'} = \varepsilon \text{ for } q \neq q'.$$

  We can claim that $[x_G(wa)]_{q'} = [x_{\mathcal{G}}(wa)]_{q'}$ because, when transition a is fired, the marking of a place which is neither an input nor an output place doesn't evolve (and the instant at which the last token has arrived remains unchanged).

Example 5 We consider the same Petri net as the one introduced in [8] to model and study a safe jobshop. This Petri net $\mathcal{G}$ is displayed in Figure 3. Using Proposition 1, we define automaton $G = (Q, A, \alpha, \mu)$ (depicted in Figure 4) representing $\mathcal{G}$ by:

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\},\ A = \{a, b, c, d\},\ \alpha = \begin{pmatrix} e \cdot e\ e \cdot e \end{pmatrix},$$

$$\mu(a) = \begin{pmatrix} \cdot & 2 & \cdot & 2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 3 & \cdot & 3 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & e \end{pmatrix},\ \mu(b) = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 2 & \cdot & 2 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & e \end{pmatrix},$$

$$\mu(c) = \begin{pmatrix} e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 2 & \cdot \\ \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 2 & \cdot \end{pmatrix}, \ \mu(d) = \begin{pmatrix} e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 3 & 3 & \cdot & 3 \\ \cdot & \cdot & 1 & 1 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

For example, we obtain

$$x_G(abcdabcd) = \alpha\mu(abcdabcd) = \begin{pmatrix} 13 & \cdot & 16 & 16 & \cdot & 16 \end{pmatrix}$$

which means that at the conclusion of firing sequence *abcdabcd*

- places q_2 and q_5 are empty,
- the token in q_1 (resp., q_3, q_4 and q_6) entered this place at time instant 13 (resp., 16, 16 and 16).

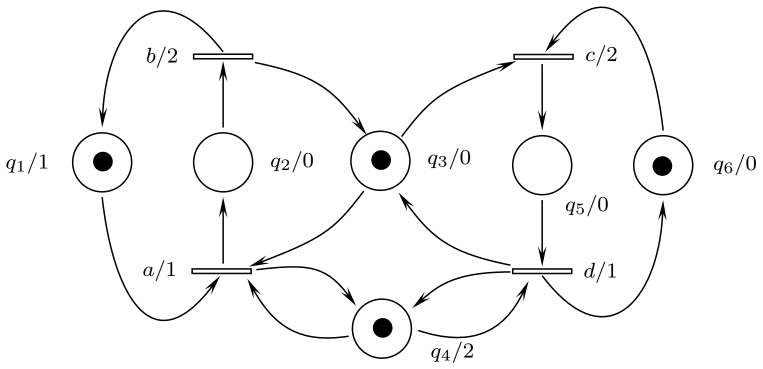

Fig. 3 A safe timed Petri net $\mathcal{G}$ modeling a JobShop.

Remark 1 It must be pointed out that the result from Proposition 1 is nothing more than an alternative to the approach introduced in [8]. In fact, S. Gaubert and J. Mairesse have also explained how to represent a safe timed Petri net by means of a (max,+) automaton: from the Petri net they build a heap model (which provides a convenient and intuitive graphical representation), and from this heap model they derive a heap automaton (particular (max,+) automaton). For a given Petri net, the (max,+) automata obtained by means of the two approaches are not identical but they are equivalent in the sense that they both enable to compute the same dater function (as in Eq. (4)). It would be hazardous to claim that one is better than the other. In particular, these automata are nondeterministic in the same cases and have the same number of states. For example, the heap automaton which can be derived from the heap model proposed in Ex. IV.2 of [8] to model Petri net $\mathcal{G}$ has

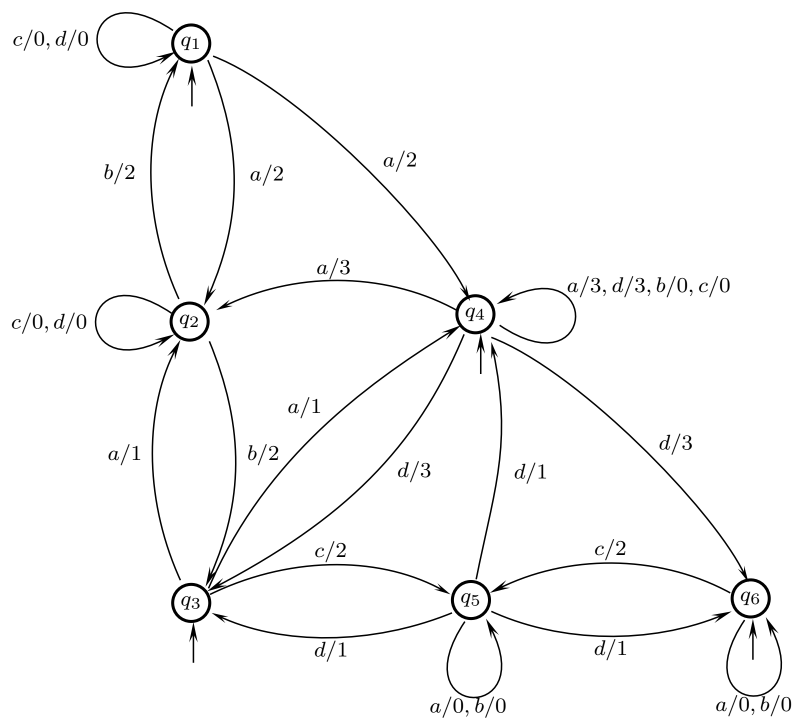

Fig. 4 (Max,+) automaton G representing the safe timed Petri net $\mathcal{G}$.

the same dimension as the (max,+) automaton G obtained in example 5 (note that a procedure to reduce the heap automaton realization is proposed for safe Petri nets admitting a state machine covering in [8]). Nevertheless, in order to justify the present contribution, one can argue that the compositions which are proposed in the next sections are based on Proposition 1 and make possible a compositional approach for the modeling of TDES by means of (max,+) automata.

Remark 2 For a safe timed Petri net $\mathcal{G}$ represented by a (max,+) automaton G according to Proposition 1, it is worth noticing that the language of G is then larger than the one of $\mathcal{G}$, that is G recognizes words which are not possible firing sequences in $\mathcal{G}$. As it is done classically in automata theory, it is possible to get a (max,+) automaton computing the dater of $\mathcal{G}$ and having the same language by using the tensor product of G with the (boolean) marking automaton derived from $\mathcal{G}$.

Remark 3 Note that (max,+) automaton G defined in Proposition 1 is generally nondeterministic. In fact,

– if the corresponding Petri net $\mathcal{G}$ has several places initially marked, then G has several initial states;
– if the corresponding Petri net $\mathcal{G}$ has a transition a with several output places (i.e., $a^\bullet > 1$), then there exists at least a line in $\mu(a)$ which contains more than one non-ε element.

Note that, in similar situations, the (max,+) automata obtained following [8] are also nondeterministic. This is a serious drawback. In fact, unlike any (boolean) finite automaton, a nondeterministic (max,+) automaton cannot always be determinized, that is transformed into an equivalent deterministic (max,+) automaton. Despite the fact that it was studied by numerous researchers (references [7, 17, 11, 16, 10] constitute a very partial survey), this determinization problem is still quite open. This problem is important both from theoretical and practical point of views. For example and regarding only applications, to manipulate deterministic models makes it possible to

– do efficient computations in speech processing [18];
– compute the optimal, as well as the average behavior, for DES [6];
– build supervisors for DES [13, 2].

For a positive example, let us consider (max,+) automaton G_1 displayed in figure 1, which is the (max,+) automaton that represents the timed Petri net $\mathcal{G}_1$ in figure 2 thanks to Proposition 1. One can easily check that deterministic automaton G'_1 displayed in figure 5 has the same dater and language as Petri net $\mathcal{G}_1$. In this case, procedures from [7] and [17] could be used to get such a deterministic representation.

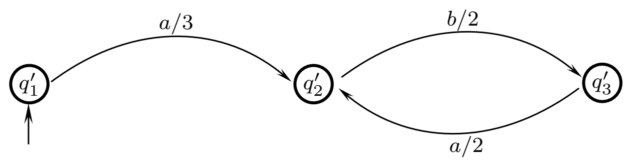

Fig. 5 Deterministic (max,+) automaton G'_1 having the same dater and the same language as the safe timed Petri net $\mathcal{G}_1$.

4 Synchronous compositions of (max,+) automata for the compositional modeling of safe timed Petri nets

In this section, two synchronous compositions are defined for (max,+) automata. More precisely, assuming that Proposition 1 has been applied to get (max,+) automata representing safe timed Petri nets, it is investigated how to compose these automata in order to model that the corresponding Petri nets are combined by merging places and/or transitions.

4.1 Synchronous composition of $(\max, +)$ automata corresponding to merging places

Proposition 1 states how to represent any safe timed Petri net $\mathcal{G}$ by a (max,+) automaton G. Now we consider several safe timed Petri nets $\mathcal{G}_1, \ldots, \mathcal{G}_n$ which are composed by merging places. In the following, a composition for the corresponding (max,+) automata $G_i = (Q_i, A_i, \alpha_i, \mu_i)$, $i = 1, \ldots, n$, is proposed to capture such a combination of subnets. We then assume that $Q_i, i = 1, \ldots, n$, are intersecting sets (common elements correspond to places to be merged). We furthermore consider that sets A_i are all disjoint since, should the opposite occur, the Petri net obtained by merging places from $\mathcal{G}_i$'s wouldn't be free-labelled.

Before defining the synchronous composition, we introduce a notation that shall be used several times in the following (even with sets A_i not necessarily disjoint).

Notation 1 *For matrices $\mu_i(a) \in \mathbb{R}_{\max}^{|Q_i|\times|Q_i|}$, $i = 1, \ldots, n$, $a \in A_i$, we shall denote ${}^\bullet a_{\mu_i}$ and ${}^\bullet a_\mu$ the subsets of $\bigcup_{i=1,\ldots,n} Q_i$ defined by*

$$^\bullet a_{\mu_i} = \{q \in Q_i | \exists q' \in Q_i \textit{ with } [\mu_i(a)]_{qq'} \neq \varepsilon\},$$

and

$$^\bullet a_\mu = \bigcup_{i=1,\ldots,n} {}^\bullet a_{\mu_i},$$

with the convention that $[\mu_i(a)]_{qq'} = \varepsilon$ if $[\mu_i(a)]_{qq'}$ is undefined.
In a similar manner

$$a^\bullet_{\mu_i} = \{q' \in Q_i | \exists q \in Q_i \textit{ with } [\mu_i(a)]_{qq'} \neq \varepsilon\},$$

and

$$a^\bullet_\mu = \bigcup_{i=1,\ldots,n} a^\bullet_{\mu_i}.$$

Proposition 2 below states what is the modeling power obtained by means of the synchronous product introduced in the next definition. It is expressed as a Petri net resulting from the merging of common places in subnets $\mathcal{G}_i$ (represented by automata G_i). It is supposed that such a common place has an identical sojourn duration in the $\mathcal{G}_i$'s it belongs to. This assumption is natural since such a common place typically models a same resource in the TDES.

Definition 4 (synchronous product) We denote $G_1 \bowtie \cdots \bowtie G_n = (Q, A, \alpha, \mu)$ the automaton resulting from the synchronous product of $G_1, \ldots, G_n$ defined by:

1. $Q = \cup_{i=1}^n Q_i$,
2. $A = \cup_{i=1}^n A_i$,

3. $\forall q \in Q$,
$$\alpha_q = \begin{cases} e \text{ if } \exists i \text{ such that } [\alpha_i]_q \text{ is defined and } [\alpha_i]_q = e, \\ \varepsilon \text{ otherwise.} \end{cases},$$
4. $\forall q, q' \in Q$, $\forall a \in A$,
$$[\mu(a)]_{qq'} = \begin{cases} [\mu_i(a)]_{qq'}, & \text{if } q \in {}^\bullet a_{\mu_i} \text{ and } q' \in a^\bullet_{\mu_i}, \\ e, & \text{if } q = q' \text{ and } q \notin {}^\bullet a_\mu \cup a^\bullet_\mu, \\ \varepsilon, & \text{otherwise,} \end{cases}$$

Proposition 2 *Let $\mathcal{G}_i = (Q_i, A_i, \alpha_i, \mu_i)$, $i = 1, \ldots, n$, be safe timed Petri nets represented by (max,+) automata G_i with sets Q_i and A_i being respectively intersecting and disjoint. We consider that common places from $\mathcal{G}_i$, $i = 1, \ldots, n$ are merged and we assume that the resulting timed Petri net $\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n$ is safe. We have*
$$x_{G_1 \bowtie \cdots \bowtie G_n}(w) = x_{\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n}(w),$$
for all $w \in A^*$ corresponding to a firing sequence in $\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n$.

Proof Definition 4 for $G_1 \bowtie \cdots \bowtie G_n$ is identical to the definition according to Proposition 1 that can be obtained for automaton G representing $\mathcal{G} = \mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n$. Then Proposition 1 proves that
$$x_{G_1 \bowtie \cdots \bowtie G_n}(w) = x_{\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n}(w),$$
for all $w \in A^*$ corresponding to a firing sequence in $\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n$.

Remark 4

- In Proposition 2, it is assumed that timed Petri net $\mathcal{G}_1 \bowtie \cdots \bowtie \mathcal{G}_n$ is safe. This is the case when the modeling approach consists in describing a TDES by a safe Petri net (possibly large and complex), which is modularly structured so that it can be decomposed in subnets (generally small and simple) sharing common places. Apart from this favorable case, let us remind that there exist numerous structural criteria characterizing the safeness of a Petri net, see for example [19, §VI.B]. These criteria generally depend on the subclass (state graph, event graph, free-choice net,...) the Petri net belongs to, and they should be combined for examining the safeness of the net resulting from merging places in subnets.
- Associativity and commutativity of this synchronous composition are obvious.
- The attribute 'synchronous' is used with this composition to describe the fact that merged places (resp. common states) in the Petri nets (resp. automata) are marked (resp. reached) at the same instant in the components.
- If we fully extend the reasoning, safe timed Petri nets can be represented as the synchronous compositions of subnets composed of only one transition with its input and output places.

Example 6 Let us return to the timed Petri net $\mathcal{G}$ which is displayed in Figure 3. Now we consider this timed Petri net as two safe event graphs that have

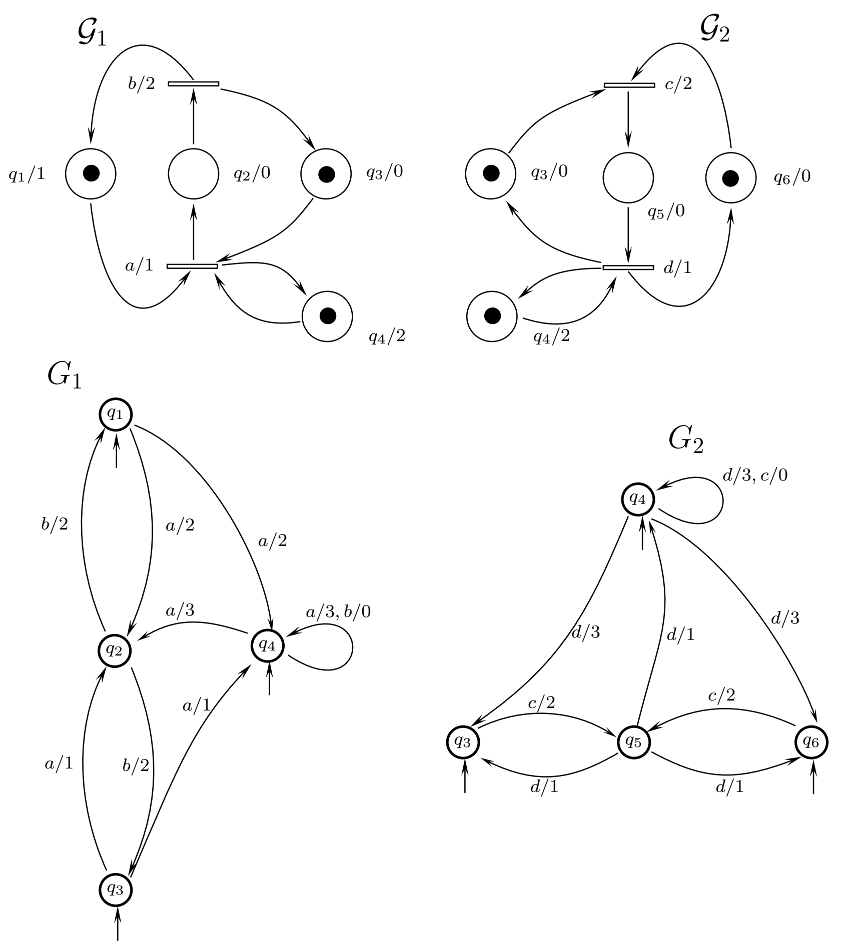

Fig. 6 Petri net $\mathcal{G}$ decomposed as subnets $\mathcal{G}_1, \mathcal{G}_2$ sharing places with their corresponding (max,+) automata G_1, G_2.

common places. They are depicted in Figure 6 with the corresponding (max,+) automata G_1, G_2.

We have

$$\alpha_1 = \begin{pmatrix} e & . & e & e \end{pmatrix}, \quad \mu_1(a) = \begin{pmatrix} . & 2 & . & 2 \\ . & . & . & . \\ . & 1 & . & 1 \\ . & 3 & . & 3 \end{pmatrix}, \quad \mu_1(b) = \begin{pmatrix} . & . & . & . \\ 2 & . & 2 & . \\ . & . & . & . \\ . & . & . & e \end{pmatrix},$$

$$\alpha_2 = \begin{pmatrix} e & e & . & e \end{pmatrix}, \quad \mu_2(c) = \begin{pmatrix} . & . & 2 & . \\ . & e & . & . \\ . & . & . & . \\ . & . & 2 & . \end{pmatrix}, \quad \mu_2(d) = \begin{pmatrix} . & . & . & . \\ 3 & 3 & . & 3 \\ 1 & 1 & . & 1 \\ . & . & . & . \end{pmatrix}.$$

Hence, the composed automaton using Definition 4 and representing the whole timed Petri net coincides with the automaton G obtained in example 5.

4.2 Synchronous composition of (max, +) automata corresponding to merging transitions

Another (symmetric towards Def. 4) type of synchronous composition is introduced for (max,+) automata G_i, $i = 1, \dots, n$. When automata G_i represent safe timed Petri nets $\mathcal{G}_i$, this composition makes it possible to model merging of common transitions in subnets $\mathcal{G}_i$. Sets A_i are then assumed to intersect (common elements are the transitions to be merged), but set Q_i are supposed to be disjoint in order to preserve the free-labelled property of the Petri net obtained by merging transitions.

The next proposition states the modeling power obtained by means of the synchronous product introduced in Definition 5. It is expressed as a Petri net resulting from the merging of common transitions in the corresponding subnets. It is assumed that such a common transition has an identical firing duration in all the $\mathcal{G}_i$'s it belongs to (this assumption is ordinary since it typically models a same event).

Definition 5 (synchronous product) We denote $G_1 \| \dots \| G_n = (Q, A, \alpha, \mu)$ the (max, +) automaton resulting from the synchronous product of $G_1, \dots, G_n$ defined by:

1. $Q = \cup_{i=1}^n Q_i$,
2. $A = \cup_{i=1}^n A_i$,
3. $\forall q \in Q$,
$$\alpha_q = \begin{cases} e \text{ if } \exists i \text{ such that } [\alpha_i]_q \text{ is defined and } [\alpha_i]_q = e, \\ \varepsilon \text{ otherwise.} \end{cases},$$
4. $\forall q, q' \in Q$, $\forall a \in A$,
$$[\mu(a)]_{qq'} = \begin{cases} [\mu_i(a)]_{qq'}, & \text{if } q \in {}^\bullet a_{\mu_i} \text{ and } q' \in a^\bullet_{\mu_i}, \\ \bigoplus_{\{p \in a^\bullet_{\mu_i}\}} [\mu_i(a)]_{qp}, & \text{if } q \in {}^\bullet a_{\mu_i} \text{ and } q' \in a^\bullet_{\mu} \setminus a^\bullet_{\mu_i}, \\ e, & \text{if } q = q' \text{ and } q \notin {}^\bullet a_\mu \cup a^\bullet_\mu, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Proposition 3 *Let $\mathcal{G}_i = (Q_i, A_i, \alpha_i, \mu_i)$, $i = 1, \dots, n$, be safe timed Petri nets represented by (max,+) automata G_i with sets Q_i and A_i being respectively disjoint and intersecting. We consider that common transitions in $\mathcal{G}_i$, $i = 1, \dots, n$ are merged and we denote $\mathcal{G}_1 \| \dots \| \mathcal{G}_n$ the resulting safe timed Petri nets. We have*
$$x_{\mathcal{G}_1 \| \dots \| \mathcal{G}_n}(w) = x_{G_1 \| \dots \| G_n}(w),$$
for all $w \in A^*$ corresponding to a firing sequence in $\mathcal{G}_1 \| \dots \| \mathcal{G}_n$.

Proof Since subnets are free-labelled, and, in particular, since only one transition in $\mathcal{G}_i$ can have label a, the corresponding automaton G_i is such that we have for $q \in {}^\bullet a_{\mu_i}$
$$[\mu_i(a)]_{qp} = [\mu_i(a)]_{qp'},$$

for all $p, p' \in a^{\bullet}_{\mu_i}$. This implies that maximisation in the second item defining $[\mu(a)]_{qq'}$ in Def. 5 doesn't operate for such automata G_i. The definition for $G_1\| \ldots \|G_n$ according to Definition 5 is identical to the definition according to Proposition 1 that can be obtained for automaton G representing $\mathcal{G}_1\| \ldots \|\mathcal{G}_n$. Then Proposition 1 proves that

$$x_{\mathcal{G}_1\|\ldots\|\mathcal{G}_n}(w) = x_{G_1\|\ldots\|G_n}(w),$$

for all $w \in A^*$ corresponding to a firing sequence in $\mathcal{G}_1\| \ldots \|\mathcal{G}_n$.

Remark 5

- Merging transitions of safe subnets always results in a safe Petri net. This is why, unlike in Proposition 2, no additional assumption is required for $\mathcal{G}_1\| \ldots \|\mathcal{G}_n$ in Proposition 3.
- It should be noted here that, compared to the automata definition of [14], where timed subnets are abstracted away, the definition of morphism matrix from Definition 5 is much simpler.
- This composition is said to be synchronous to express the fact that common transitions in the Petri nets (resp. in the automata) are fired (resp. crossed) at the same time.
- Associativity and commutativity of this synchronous composition are obvious.

Example 7 Let us consider again safe timed Petri net $\mathcal{G}$ displayed in Figure 3. Now we consider this net as four safe state graphs that share common transitions. Together with the corresponding (max,+) automata $G_1, \ldots, G_4$, they are depicted on Figure 7.

We have

$$\alpha_1 = \begin{pmatrix} e & . \end{pmatrix}, \quad \mu_1(a) = \begin{pmatrix} . & 2 \\ . & . \end{pmatrix}, \quad \mu_1(b) = \begin{pmatrix} . & . \\ 2 & . \end{pmatrix},$$

$$\alpha_2 = \begin{pmatrix} e & . & . \end{pmatrix}, \quad \mu_2(a) = \begin{pmatrix} . & 1 & . \\ . & . & . \\ . & . & . \end{pmatrix}, \quad \mu_2(b) = \begin{pmatrix} . & . & . \\ 2 & . & . \\ . & . & . \end{pmatrix},$$

$$\mu_2(c) = \begin{pmatrix} . & . & 2 \\ . & . & . \\ . & . & . \end{pmatrix}, \quad \mu_2(d) = \begin{pmatrix} . & . & . \\ . & . & . \\ 1 & . & . \end{pmatrix}$$

$$\alpha_3 = (e), \quad \mu_3(a) = (3), \quad \mu_3(d) = (3),$$

$$\alpha_4 = \begin{pmatrix} e & . \end{pmatrix}, \quad \mu_4(c) = \begin{pmatrix} . & 2 \\ . & . \end{pmatrix}, \quad \mu_4(d) = \begin{pmatrix} . & . \\ 1 & . \end{pmatrix}.$$

Hence, the composed automaton $\mathcal{G}_1\| \ldots \|\mathcal{G}_4$ representing timed Petri net $\mathcal{G}$ is illustrated on Figure 8 and has the following linear representation (with states ordered according to the sequence $(q_1, q_2, q_3, q'_2, q'_5, q_4, q_6, q_5)$).

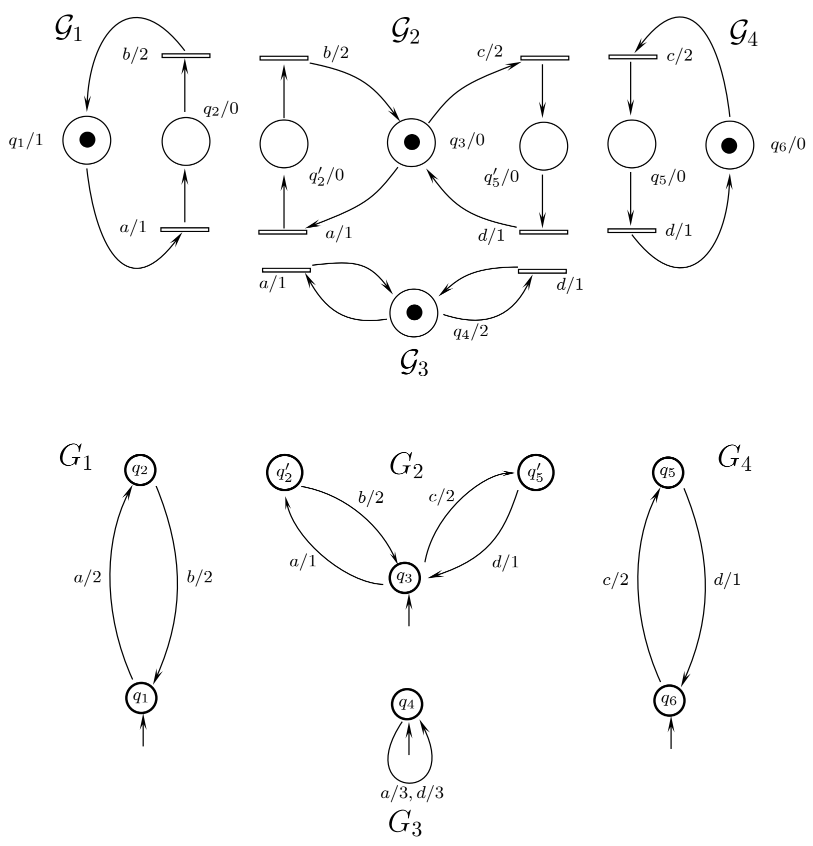

Fig. 7 Petri net $\mathcal{G}$ decomposed as subnets $\mathcal{G}_1, \ldots, \mathcal{G}_4$ sharing transitions with their corresponding (max,+) automata $G_1, \ldots, G_4$.

$$\alpha = \left(e \; . \middle| e \; . \; . \middle| e \middle| e \; . \right),$$

$$\mu(a) = \left(\begin{array}{cc|ccc|c|cc} . & 2 & . & 2 & . & 2 & . & . \\ . & . & . & . & . & . & . & . \\ \hline . & 1 & . & 1 & . & 1 & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ \hline . & 3 & . & 3 & . & 3 & . & . \\ \hline . & . & . & . & . & . & e & . \\ . & . & . & . & . & . & . & e \end{array}\right), \mu(b) = \left(\begin{array}{cc|ccc|c|cc} . & . & . & . & . & . & . & . \\ 2 & . & 2 & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . \\ 2 & . & 2 & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & e & . & . \\ \hline . & . & . & . & . & . & e & . \\ . & . & . & . & . & . & . & e \end{array}\right),$$

$$\mu(c) = \left(\begin{array}{cc|ccc|c|cc} e & . & . & . & . & . & . & . \\ . & e & . & . & . & . & . & . \\ \hline . & . & . & . & 2 & . & . & 2 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ \hline . & . & . & . & . & e & . & . \\ \hline . & . & . & . & 2 & . & . & 2 \\ . & . & . & . & . & . & . & . \end{array}\right), \mu(d) = \left(\begin{array}{cc|ccc|c|cc} e & . & . & . & . & . & . & . \\ . & e & . & . & . & . & . & . \\ \hline . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & 1 & 1 & . \\ \hline . & . & 3 & . & . & 3 & 3 & . \\ \hline . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & 1 & 1 & . \end{array}\right).$$

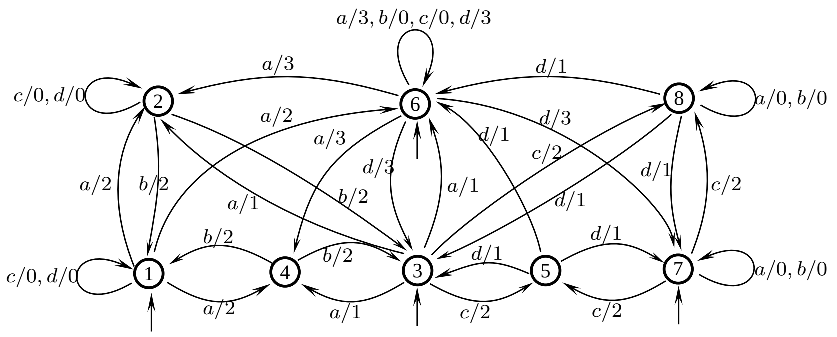

Fig. 8 (Max,+) automaton $G_1\|\ldots\|G_4$ resulting from the synchronous composition of $G_1,\ldots,G_4$ displayed on figure 7.

5 Asynchronous composition of (max,+) automata towards the modeling of bounded timed Petri nets

We now define a so-called *asynchronous composition* of several (max,+) automata admitting isomorphic state-transition structures (i.e. identical structures for matrices μ). This enables us to model similar (max,+) automata operating simultaneously and independently (asynchronously).

Definition 6 (asynchronous product) Let $G_i = (Q_i, A_i, \alpha_i, \mu_i)$, $i = 1,\ldots,n$, be several (max,+) automata defined on the same set of events A_i but with disjoint sets of states Q_i. We assume that for all $i, j \in \{1, 2, \ldots, n\}$, we have $A_i = A_j$, $|Q_i| = |Q_j|$, and $\mu_i(a) = \mu_j(a)$ for all $a \in A_i$, that we denote indifferently A_0, $|Q_0|$ and $\mu_0(a)$. This implies that G_i and G_j possibly differ only through vectors α_i and α_j. We denote $G_1 \nparallel \cdots \nparallel G_n = (\mathsf{Q}, \mathsf{A}, \delta, \nu) = \mathsf{G}$ the (max,+) automaton resulting from the asynchronous product of $G_1,\ldots,G_n$ defined by:

– $\mathsf{Q} = \cup_{i=1}^{n} Q_i$,
– $\mathsf{A} = \cup_{i=1}^{n} A_i = A_0$,

- $\delta = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \end{pmatrix}$,
- for $a \in \mathsf{A}$, $\nu(a) \in \mathbb{R}_{\max}^{n|Q_0| \times n|Q_0|}$,

$$\nu(a) = \mathsf{M} \oplus \lambda(a)$$

with

$$\mathsf{M} = \begin{pmatrix} \varepsilon_{|Q_0|} & \cdots & & \cdots & \varepsilon_{|Q_0|} \\ I_{|Q_0|} & \ddots & & & \vdots \\ \varepsilon_{|Q_0|} & I_{|Q_0|} & & & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \varepsilon_{|Q_0|} & \cdots & \varepsilon_{|Q_0|} & I_{|Q_0|} & \varepsilon_{|Q_0|} \end{pmatrix}, \lambda(a) = \begin{pmatrix} \varepsilon_{|Q_0|} & \cdots & & \varepsilon_{|Q_0|} & \mu_0(a) \\ \vdots & \ddots & & & \varepsilon_{|Q_0|} \\ & & & & \\ \vdots & & & \ddots & \vdots \\ \varepsilon_{|Q_0|} & \cdots & & \cdots & \varepsilon_{|Q_0|} \end{pmatrix}.$$

Remark 6 In Definition 6, if $n = 1$ we then have $\nu(a) = \mu_0(a) = \mu_1(a)$ for all a, and $\delta = \alpha_1$. This implies that $\mathsf{G} = G_1$.

With the considered assumptions on Petri nets (preselection policy considering all feasible choices and earliest functioning, see Section 2), tokens in a n-bounded state graph evolve independently (firings caused by a given token aren't influenced by other tokens). Next proposition shows that asynchronous composition of Definition 6 makes it possible to compute daters which describe the evolution 'in isolation' of each of the tokens in a n-bounded timed state graph, that is, to model properly the behavior of such a Petri net.

Let us consider a n-bounded timed state-graph $\mathscr{G}$. We associate n safe state-graphs $\mathcal{G}_i$ to $\mathscr{G}$, each corresponding to $\mathscr{G}$ in which only one of the initial tokens has been kept. We denote M_i the vectors of initial markings and L_i, the languages associated to $\mathcal{G}_i$, $i = 1, \dots, n$. According to Proposition 1, these safe state graphs are represented by (max,+) automata $G_i = (Q_i, A_0, \alpha_i, \mu_i)$, $i = 1, \dots, n$, defined on the same set of events A_0, with disjoint sets of states and such that for all $i, j \in \{1, 2, \dots, n\}$, $|Q_i| = |Q_j|$ and $\mu_i(a) = \mu_j(a)$ for all $a \in A_0$. The next proposition states that the asynchronous product (cf. Definition 6) of these automata G_i can be used to evaluate the daters associated with safe state graphs $\mathcal{G}_i$, and consequently to represent n-bounded state graph $\mathscr{G}$.

Proposition 4 *Let x_{G} be the vector of generalized daters defined for (max,+) automaton $\mathsf{G} = G_1 \nparallel \cdots \nparallel G_n$ and satisfying the recurrence*

$$\begin{cases} x_{\mathsf{G}}(\epsilon) = \delta, \\ x_{\mathsf{G}}(wa) = x_{\mathsf{G}}(w)\nu(a). \end{cases} \tag{5}$$

Then x_{G} and the vector of generalized daters associated to state graphs $\mathcal{G}_i, i = 1, \dots, n$ are related as follows. For a string

$$\begin{array}{l} w = a_{1,1}a_{1,2} \dots a_{1,n}a_{2,1} \dots a_{2,n} \dots a_{k,1} \dots a_{k,j}, \ k, j \in \mathbb{N} \\ \textit{with } a_{1,i}a_{2,i} \dots a_{l,i} \in L_i \textit{ for } i = 1, \dots, n, \ l = k \textit{ or } k-1, \end{array} \tag{6}$$

we have

$$x_{\mathsf{G}}(w) = \begin{pmatrix} x_{\mathcal{G}_{j+1}}(a_{1,j+1}a_{2,j+1}\dots a_{k-1,j+1}) \\ \vdots \\ x_{\mathcal{G}_n}(a_{1,n}a_{2,n}\dots a_{k-1,n}) \\ x_{\mathcal{G}_1}(a_{1,1}a_{2,1}\dots a_{k,1}) \\ \vdots \\ x_{\mathcal{G}_j}(a_{1,j}a_{2,j}\dots a_{k,j}) \end{pmatrix}^T . \tag{7}$$

Proof We show by induction that for all j, k we have for w defined according to (6)

$$x_{\mathsf{G}}(w) = \begin{pmatrix} \alpha_{j+1}\mu_0(a_{1,j+1}a_{2,j+1}\dots a_{k-1,j+1}) \\ \vdots \\ \alpha_n\mu_0(a_{1,n}a_{2,n}\dots a_{k-1,n}) \\ \alpha_1\mu_0(a_{1,1}a_{2,1}\dots a_{k,1}) \\ \vdots \\ \alpha_j\mu_0(a_{1,j}a_{2,j}\dots a_{k,j}) \end{pmatrix}^T . \tag{8}$$

In fact, for $k = 1$, $j = 1$, $l = 1$ we have $w = a_{1,1}$ and

$$\begin{aligned} x_{\mathsf{G}}(a_{1,1}) &= \delta\nu(a_{1,1}) \\ &= \big(\alpha_2 \dots \alpha_n \ \alpha_1\mu_0(a_{1,1})\big) . \end{aligned}$$

Let us assume that (8) is satisfied for w given by (6), we then check that (8) is also satisfied for $wa_{k,j+1}$ with $a_{1,j+1}a_{2,j+1}\dots a_{k,j+1} \in L_{j+1}$:

$$\begin{aligned} x_{\mathsf{G}}(wa_{k,j+1}) &= x_{\mathsf{G}}(w)\nu(a_{k,j+1}) \\ &= \begin{pmatrix} \alpha_{j+2}\mu_0(a_{1,j+2}a_{2,j+2}\dots a_{k-1,j+2}) \\ \vdots \\ \alpha_n\mu_0(a_{1,n}a_{2,n}\dots a_{k-1,n}) \\ \alpha_1\mu_0(a_{1,1}a_{2,1}\dots a_{k,1}) \\ \vdots \\ \alpha_{j+1}\mu_0(a_{1,j+1}a_{2,j+1}\dots a_{k,j+1}) \end{pmatrix}^T . \end{aligned}$$

In (8), each term $\alpha_j\mu_0(a_{1,j}a_{2,j}\dots a_{k,j})$ corresponds to $x_{G_j}(a_{1,j}a_{2,j}\dots a_{k,j})$, that is to the generalized dater associated with deterministic (max,+) automaton G_j. As G_j represents safe state graph $\mathcal{G}_j$, this term is equal to $x_{\mathcal{G}_j}(a_{1,j}a_{2,j}\dots a_{k,j})$.

Remark 7 Note that only strings corresponding to interlacements of words of L_i (as defined in (6)) are considered in (7) and not all $w \in A^*$. The set of these strings can be obtained by means of the prefix closure of the free product of L_1, $L_2 \dots L_n$ (as defined for example in [21, sec. 6.1]).

Remark 8 If timed state graph $\mathscr{G}$ is 1-bounded, that is safe, it is seen as a single safe state graph $\mathcal{G}_1$ with corresponding (max,+) automaton G_1. As noticed in Remark 6, G derived from Definition 6 is then equal to G_1 and recurrence (5) then coincides with (2) for G_1.

Example 8 Let us consider the 2-bounded state graph $\mathscr{G}$ represented in figure 9. Associated safe state graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are also represented in the figure, and their equivalent (max,+) automata G_1 and G_2 are defined by $\alpha_1 = \begin{pmatrix} e & \cdot \end{pmatrix}$, $\alpha_2 = \begin{pmatrix} \cdot & e \end{pmatrix}$ and for $i = 1, 2$

$$\mu_i(a) = \begin{pmatrix} \cdot & \cdot \\ 2 & \cdot \end{pmatrix},\ \mu_i(b) = \begin{pmatrix} \cdot & 1 \\ \cdot & \cdot \end{pmatrix}, \mu_i(c) = \begin{pmatrix} 3 & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

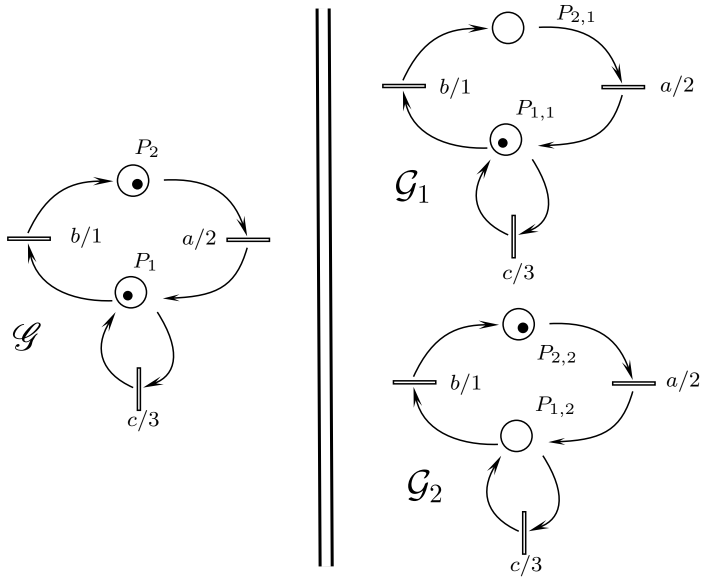

Fig. 9 A 2-bounded state graph $\mathscr{G}$ and associated safe state graphs $\mathcal{G}_1$, $\mathcal{G}_2$.

Their asynchronous composition $G_1 \not\parallel G_2$ is defined by

$$\delta = \begin{pmatrix} e & \cdot & \cdot & e \end{pmatrix},\ \nu(a) = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 2 & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix},\ \nu(b) = \begin{pmatrix} \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix},\ \nu(c) = \begin{pmatrix} \cdot & \cdot & 3 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix}.$$

For example, we obtain $x_{\mathsf{G}}(cacbba) = \begin{pmatrix} \cdot & 7 | 5 & \cdot \end{pmatrix}$.

According to decomposition (6), sequence '*cacbba*' corresponds to firing sequences '*ccb*' in $\mathcal{G}_1$ and '*aba*' in $\mathcal{G}_2$.

From Prop. 4, sub-vector $\begin{pmatrix} \cdot & 7 \end{pmatrix}$ (resp. $\begin{pmatrix} 5 & \cdot \end{pmatrix}$) of $x_{\mathsf{G}}(cacbba)$ is equal to $x_{\mathcal{G}_1}(ccb)$ (resp. $x_{\mathcal{G}_2}(aba)$) which contains the daters associated to $\mathcal{G}_1$ (resp. $\mathcal{G}_2$). It indicates that a token arrives at time instant 7 in $P_{2,1}$ (resp. at 5 in $P_{1,2}$) and no token is contained in $P_{1,1}$ (resp. in $P_{2,2}$) at the conclusion of firing sequence 'ccb' in $\mathcal{G}_1$ (resp. 'aba' in $\mathcal{G}_2$).

Acknowledgments

Fruitful discussion with Philippe Darondeau (INRIA Rennes) is gratefully acknowledged. The authors thank the anonymous referees for their helpful comments.

6 Conclusion

We have presented a new direct modeling approach that associates a (max,+) automaton with any safe timed Petri net. Two complementary types of synchronous product of (max,+) automata are proposed and their counterparts for timed Petri nets are mergings of subnets on shared places or shared transitions. We have also proposed asynchronous product of (max,+) automata that enables us to model bounded timed state graphs.

In the future work, we would like to investigate in the combinations of synchronous and asynchronous compositions in order to represent more general bounded timed Petri nets.

In addition, the observations in Remark 3 suggest an open question: depending on wether the approach in Section 3 or the modeling approach in [8] is used, does the determinization problem find positive answers for the same classes of systems? It has also been discussed in the paper that synchronous products of (max,+) automata yield nondeterministic (max,+) automata. Another open direction is to investigate determinization of these particular (max,+) automata.

References

1. F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity.* Wiley, 1992.
2. E. Badouel, A. Bouillard, P. Darondeau, and J. Komenda. Residuation of tropical series: rationality issues. In *joint 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'11)*, pages 3855–3861, 2011.
3. P. Buchholz and P. Kemper. Weak bisimulation for (max/+) automata and related models. *J. Autom. Lang. Comb.*, 8:187–218, April 2003.
4. C. G. Cassandras and S. Lafortune. *Introduction to DES.* Springer-Verlag New York, Inc., 2006.
5. R. David and H. Alla. *Discrete, continuous, and hybrid Petri Nets (2nd edition).* Springer, Paris, 2010.
6. S. Gaubert. Performance Evaluation of (max,+) Automata. *IEEE Transactions on Automatic Control*, 40(12):2014–2025, 1995.
7. S. Gaubert and J. Mairesse. Asymptotic analysis of heaps of pieces and application to timed petri nets. In *Petri nets and performance models (PNPM'99)*, pages 158 – 169, 1999.

8. S. Gaubert and J. Mairesse. Modeling and Analysis of Timed Petri Nets using Heaps of Pieces. *IEEE Transactions on Automatic Control*, 44(4):683–698, 1999.
9. L. Houssin. Cyclic jobshop problem and (max,plus) algebra. In *18th IFAC World Congress*, pages 2717–2721, Milan, Italy, 2011.
10. D. Kirsten. A burnside approach to the termination of mohri's algorithm for polynomially ambiguous min-plus-automata. *RAIRO - Theoretical Informatics and Applications*, 42(3):553–581, 2008.
11. I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoretical Computer Science*, pages 349–373, 2004.
12. J. Komenda, S. Lahaye, and J.-L. Boimond. Le produit synchrone des automates (max,+). *Journal Européen des Systèmes Automatisés*, 43(7), 2009.
13. J. Komenda, S. Lahaye, and J.-L. Boimond. Supervisory Control of (max,+) Automata: A Behavioral Approach. *Discrete Event Dynamic Systems*, 19(4):525–549, 2009.
14. S. Lahaye, J. Komenda, and J.-L. Boimond. Compositions of (max,+) automata. In *International Workshop on Discrete Event Systems (WODES 2012)*, Guadalajara, Mexico, 2012.
15. S. Lahaye, J. Komenda, and J.-L. Boimond. Modélisation modulaire à l'aide d'automates (max,+). 2012. Conférence Internationale Francophone d'Automatique (CIFA'2012).
16. S. Lombardy and J. Sakarovitch. Sequential ? *Theoretical Computer Science*, 359(1-2):224–244, 2006.
17. J. Mairesse and L. Vuillon. Asymptotic behavior in a heap model with two pieces. *Theoretical Computer Science*, 270(12):525 – 560, 2002.
18. M. Mohri. Weighted automata algorithms. In *Handbook of weighted automata*. Springer, 2011.
19. T. Murata. Petri Nets : Properties, Analysis and Applications. *IEEE Proceedings: Special issue on Discrete Event Systems*, 77:541–580, Jan. 1989.
20. C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. Ph.D. thesis, M.I.T., 1973.
21. J. Sakarovitch. *Éléments de théorie des automates*. Vuibert, 2003.
22. R. Su, J.H. van Schuppen, and J.E. Rooda. The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Transactions on Automatic Control*, 57(1):105–118, 2012.