# Compositions of (max,+) automata ⋆

**Sébastien Lahaye** * **Jan Komenda** ** **Jean-Louis Boimond** *

* *LUNAM Université, LISA, Angers, France ([sebastien.lahaye, jean-louis.boimond]@univ-angers.fr).*
** *Institute of Mathematics - Brno Branch, Czech Academy of Sciences, Czech Republic (komenda@ipm.cz).*

**Abstract:** Automata with weights (multiplicities) in the so called (max,+) semiring constitute a class of timed automata. Their modeling power has been studied in Gaubert and Mairesse (1999): at least timed safe Petri nets can be modeled by means of (max,+) automata.
In this contribution, we define compositions for (max,+) automata. The motivation is to be able to model a complex system by composing sub-models representing its elementary parts. In doing so we expect two benefits:

- The modeling activity should be eased and enhanced since the model can be obtained in a modular manner with a good understanding of phenomena.
- The modeling power of (max,+) automata can be refined.

*Keywords:* Discrete event systems, (max,+) automata, modeling, compositions

## 1. INTRODUCTION

Weighted automata with weights (multiplicities) in the idempotent semiring $(\mathbb{R} \cup \{-\infty\}, \max, +)$ can be viewed as an important class of timed automata. Their modeling power in terms of timed discrete-event systems (TDES) is studied in Gaubert and Mairesse (1999), where it is shown that the behavior of any safe timed Petri net can be expressed by a so-called heap automaton, which is a special type of (max,+)-automaton. Let us recall that safe timed Petri net are timed Petri nets, where the marking of any place is bounded by 1 (at most 1 token can be at a place in any time). Recently (max,+)-automata have been applied to performance evaluation Gaubert (1995); Su and Woeginger (2011), scheduling Houssin (2011) and control Komenda et al. (2009b) problems for a large class of TDES.

This paper focuses on the modeling of TDES using (max,+)-automata and we propose an alternative way of modeling some (but not all) $m$-bounded timed Petri nets using (max,+)-automata. The main advantage of our approach is that it is compositional, i.e. the (max,+)-automaton of the overall modularly structured and complex TDES is obtained as asynchronous and/or synchronous composition(s) of (max,+)-automata corresponding to smaller subsystems. In terms of Petri nets, these subsystems correspond to safe timed state graphs and it is first shown these can be equivalently modeled by deterministic (max,+) automata.

We then consider the case of asynchronous composition, where the local timed state graphs are composed in an asynchronous way. We compose several similar (max,+)-automata that have isomorphic state-transition structure (the same number of states and identical morphism matrices), i.e. they differ only by vector of initial states (final states are not considered here). This asynchronous product is helpful for modeling $m$-bounded timed state graphs (with $m > 1$) as asynchronous composition of safe timed state graphs.

Subsequently we present the construction of the synchronous product, where different local components are related *via* shared events on which they must synchronize. This is similar as composing (putting together) several timed state graphs using common (called synchronization) transitions. It is proven that the behavior of the product of local ($m$-bounded) timed state graphs given by merging the common (synchronization) transition is equal to the behavior of the synchronous product of (max,+)-automata corresponding to individual ($m$-bounded) timed state graphs. Note that the synchronous product of (max,+)-automata is a nondeterministic (max,+)-automaton defined over standard union alphabet and moreover, its number of states is smaller than the number of states in standard synchronous products of Boolean automata, cf. Cassandras and Lafortune (2006)) or in other approaches to synchronous product of $(\max, +)$-automata Buchholz and Kemper (2003); Su et al. (2012). 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 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 this result in Lahaye et al. (2012) with restrictions to safe timed state graphs and on the transition structure. Let us also recall that a synchronous composition of (max,+)-automata has been proposed in Komenda et al. (2009a), where the product au-

⋆ This work has been partially supported by research plan AV0Z10190503 and by project EU.ICT N.224498 (DISC).

tomaton 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 it application in decentralized supervisory control of (max,+)-automata, while the product proposed in this paper is better suited for verification and performance evaluation.

The paper is organised as follows. In the following section preliminaries necessary to understand the paper are briefly recalled. In Section 3 we study the modeling power of deterministic (max, +)-automata. Section 4 is dedicated to the definition of asynchronous product of (max, +)-automata. In section 5, synchronous product of (max, +)-automata is introduced and its properties are discussed. Finally, in section 6 concluding remarks with hints on future extensions and applications 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 Baccelli et al. (1992), Gaubert (1995) and David and Alla (2010).

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

*Example 1.* The set $(\mathbb{R} \cup \{-\infty\})$ with the maximum playing the role of addition and conventional addition playing the role of multiplication is a dioid, denoted $\mathbb{R}_{\max}$, 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 zero element for the addition is the matrix denoted $\varepsilon_n$ and exclusively composed of $\varepsilon$ $(= -\infty)$. We denote $I_n$ the zero element of the multiplication, which is the matrix with $e$ $(= 0)$ on the diagonal and $\varepsilon$ $(= -\infty)$ elsewhere. Note that any $1 \times n$ vector can be embedded in this dioid by adding $n - 1$ lines full of $\varepsilon$ but this construction is abusively omitted in the following, and the coefficients equal to $\varepsilon$ in the matrices will be replaced by '$\cdot$' to lighten the presentation.

*Example 2. Formal languages* over a finite alphabet $A$ are subsets of the free monoid $A^*$, which is composed of finite sequences of letters (called words) from $A$. 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 $(Pwr(A^*), \cup, .)$. The zero language is $0 = \{\}$, the unit language is denoted $1 = \{\epsilon\}$ where $\epsilon$ is the empty (zero length) string.

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

*Definition 2.* ((max,+) automaton). A *(max,+) automaton* $G$ is a quadruple $(Q, A, \alpha, \mu)$ where [1]

- $Q$ and $A$ are finite sets of states and of events ;
- $\alpha \in \mathbb{R}_{\max}^{1\times|Q|}$ is such that $\alpha_q \neq \varepsilon$ if $q$ is an initial state ;
- $\mu : A^* \to \mathbb{R}_{\max}^{|Q|\times|Q|}$ 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 $w = a_1 \dots a_n$, we have
$$\mu(w) = \mu(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n),$$
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 $q'$.

We restrict our attention to (max,+) automata in which the initial delays (that is the coefficients in $\alpha$ different from $\varepsilon$) are all equal to $e = 0$. The vector of final delays is not considered, hence all states can be thought of as final states (as is the case for heap automata). Consequently, the underlying languages of (max,+) automata (as supports of formal power series they recognize) will always be prefix-closed.

*Example 3.* Figure 1 is the typical graphical representation which can be associated with every (max,+) automaton:

- the nodes correspond to states $q \in Q$ ;
- an edge exists from state $q \in Q$ to state $q'$ if there exists an event $a \in A$ such that $[\mu(a)]_{qq'} \neq \varepsilon$ : it represents the state transition when event $a$ occurs and 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 $Q = \{I, II\}$, $A = \{a, b, c\}$, and
$\alpha = (\,\cdot\ e\,)$, $\mu(a) = \begin{pmatrix} \cdot & \cdot \\ 2 & \cdot \end{pmatrix}$, $\mu(b) = \begin{pmatrix} \cdot & 1 \\ \cdot & \cdot \end{pmatrix}$, $\mu(c) = \begin{pmatrix} 3 & \cdot \\ \cdot & \cdot \end{pmatrix}$.
The possible events sequences are the strings: $a$, $ac$, $ab$, $acc$, $acb$, $aba$, ..., i.e. the underlying language of the (max,+) automata is the prefix-closure of $L = (ac^*b)^*$.

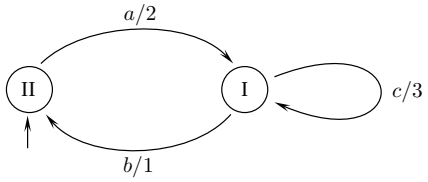


Fig. 1. A (max,+) automaton.

A (max,+) automaton is said to be *deterministic* (resp. *transition-deterministic*) if the following conditions are (resp. only the second condition is) satisfied:

(1) it has a unique initial state, namely, there is a unique $q \in Q$ such that $\alpha_q \neq \varepsilon$ ;
(2) from each state, the occurrence of an event cannot induce several possible state transitions, namely, if for all $a \in A$ each line of $\mu(a)$ contains at most one element not equal to $\varepsilon$.

Note that the term "transition-deterministic" is introduced for lack of having found an equivalent term in the literature. An automaton corresponding to a finite

[1] Without loss of generality and in order to make the presentation lighter, this definition omits to distinguish the marked states.

union of deterministic ones (called "finite union of sequential automata" in Klimann et al. (2004)) is a particular transition-deterministic automaton. Some less trivial transition-deterministic automata are introduced in section 4.

We define $x_G(w) \in \mathbb{R}_{\max}^{1\times|Q|}$ for $w \in A^*$ by

$$x_G(w) = \alpha\mu(w).$$

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

$$\begin{cases} x_G(\epsilon) = \alpha, \\ x_G(wa) = x_G(w)\mu(a). \end{cases} \tag{1}$$

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

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 $w = a_1a_2\dots a_n \in \mathcal{T}^*$ is a *firing sequence* starting from marking $M_0$ if there is a sequence of markings $M_1M_2\dots M_n$ 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. $m$ 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 consider T-timed Petri nets in which a firing finite duration $\tau$ is associated with each transition $a$: $\tau$ is the minimal time that must elapse, starting from the time at which $a$ is enabled, until this transition can fire.

Afterwards, several assumptions on the functioning of Petri nets are adopted:

- a token from the initial marking is supposed to have arrived in the Petri net at time instant 0;
- a transition is fired as soon as possible (*earliest functioning rule*) ;
- if a place has several output transitions, then for each token in this place it is required to decide on 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).

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

$$[x_\mathcal{G}(w)]_q = \begin{cases} \text{instant at which the last token arrived in } q \\ \text{(assuming that it is still contained in } q), \\ \varepsilon \text{ if } q \text{ does not contain any token.} \end{cases} \tag{2}$$

## 3. MODELING USING DETERMINISTIC (MAX,+) AUTOMATA

In the present contribution, deterministic $(max, +)$ automata will be used as elementary bricks for the modeling of TDES. The following proposition (also presented in Lahaye et al. (2012)) states their modeling power in terms of timed Petri nets.

*Proposition 1.* Deterministic $(\max, +)$ automata and safe timed state graphs have the same modeling power.

**Proof.** It is first shown that every safe timed state graph $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M)$ can be transformed into a deterministic $(\max, +)$ automaton $G = (Q, A, \alpha, \mu)$ with

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

for all $w$ corresponding respectively to a firing sequence in $\mathcal{G}$ and a state transition sequence in $G$. The transformation can be done as follows :

(1) a state in $Q$ is associated with each place in $\mathcal{P}$;
(2) the initial state (i.e., the unique $q \in Q$ such that $\alpha_q = e$) is associated with the only place containing a token (i.e., the unique $q \in \mathcal{P}$ such that $M(q) \neq 0$ and $M(q) = 1$) ;
(3) an event $a \in A$ is associated with each transition $a \in \mathcal{T}$. The morphism of $G$ is then defined by $\forall a \in A$, $\forall q, q' \in Q$,

$$[\mu(a)]_{qq'} = \begin{cases} \tau & \text{if } q = {}^\bullet a, q' = a^\bullet \\ & \text{and the firing duration of } a \text{ is equal to } \tau, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Since $\mathcal{G}$ is a state graph, ${}^\bullet a$ and $a^\bullet$ are singletons, and the morphism defined this way is such that for all $q$, there exists at most one $q'$ such that $[\mu(a)]_{qq'} \neq \varepsilon$. In other words, the derived automaton is deterministic.

For the empty string $w = \epsilon$ we have by construction

$$[x_G(\epsilon)]_q = \alpha_q = [x_\mathcal{G}(\epsilon)]_q = \begin{cases} 0 & \text{if } M(q) = 1, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Let us assume that equality

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

is true at the conclusion of firing sequence (resp. state transition sequence) $w$ in $\mathcal{G}$ (resp. in $G$), and let us check that $x_\mathcal{G}(wa) = x_G(wa)$, $\forall a$. Since $\mathcal{G}$ is a state graph, ${}^\bullet a$ and $a^\bullet$ are singletons and we denote $q = {}^\bullet a$ and $q' = a^\bullet$. Let $\tau$ be the firing duration associated with $a$, this transition is enabled from time instant $[x_\mathcal{G}(w)]_q$ and is fired at $\tau + [x_\mathcal{G}(wa)]_q$, hence

$$[x_\mathcal{G}(wa)]_{q'} = \tau + [x_\mathcal{G}(w)]_q.$$

Only place $q'$ contains a token at the conclusion of firing sequence $wa$ and then $[x_\mathcal{G}(wa)]_i = \varepsilon$ for $i \in \mathcal{P}$ and $i \neq q'$.

In automaton $G$ obtained from $\mathcal{G}$ as explained above, we have by construction $[\mu(a)]_{qq'} = \tau$ and it is the only coefficient in $\mu(a)$ different from $\varepsilon$. From (3), we have $x_G(wa) = x_G(w)\mu(a)$, hence $[x_G(wa)]_{q'} = \tau + [x_G(w)]_q$ and $[x_G(wa)]_i = \varepsilon$ for $i \in Q$ and $i \neq q'$.

In a symmetric way, a deterministic $(\max, +)$ automaton $G = (Q, A, \alpha, \mu)$ can be transformed into a safe timed state graph $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M)$, and it can be checked that

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

for all $w$.

*Example 4.* Figure 2 shows a safe timed state graph which is equivalent to the deterministic $(\max, +)$ automaton of figure 1.

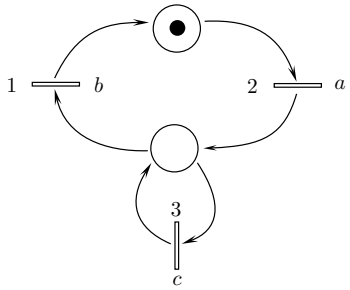


Fig. 2. A safe timed state graph.

## 4. ASYNCHRONOUS COMPOSITION

We first define a so-called *asynchronous composition* of several (max,+) automata admitting isomorphic state-transition structures $\mu$. This enables us to model similar (max,+) automata operating simultaneously and independently (asynchronously).

*Definition 4.* (Asynchronous composition).
Let $G_i = (Q_i, A, \alpha_i, \mu_i)$, $i = 1, \ldots, m$, be several (max,+) automata defined on the same set of events A but with disjoint sets of states. We assume that for each $i, j \in \{1, 2, \ldots, m\}$, $i \neq j$, we have $|Q_i| = |Q_j|$, and $\mu_i(a) = \mu_j(a)$ for all $a \in A$ (in other words, their representations possibly differ only through vectors $\alpha_i$, $i = 1, \ldots, m$), that we denote indifferently $|Q|$ and $\mu(a)$. Their asynchronous composition is the (max,+) automaton denoted $G^{G_1 \ldots G_m}$ defined on set of events $A$ with

- $Q_1 \cup Q_2 \cup \ldots \cup Q_m$ as set of states,
- $\alpha^{G_1 \ldots G_m} = (\, \alpha_1 \;\; \alpha_2 \;\; \ldots \;\; \alpha_m \,)$,
- for $a \in A$,

$$\mu^{G_1 \ldots G_m}(a) = \begin{pmatrix} \varepsilon_{|Q|} & \cdots & \cdots & \varepsilon_{|Q|} & \mu(a) \\ I_{|Q|} & \ddots & & & \varepsilon_{|Q|} \\ \varepsilon_{|Q|} & I_{|Q|} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \varepsilon_{|Q|} & \cdots & \varepsilon_{|Q|} & I_{|Q|} & \varepsilon_{|Q|} \end{pmatrix}.$$

Note that figure 5 aims at illustrating this definition.

*Proposition 2.* If (max,+) automata $G_i = (Q_i, A, \alpha_i, \mu_i)$, $i = 1, \ldots, m$, are transition-deterministic, then their asynchronous composition $G^{G_1 \ldots G_m}$ is also transition-deterministic.

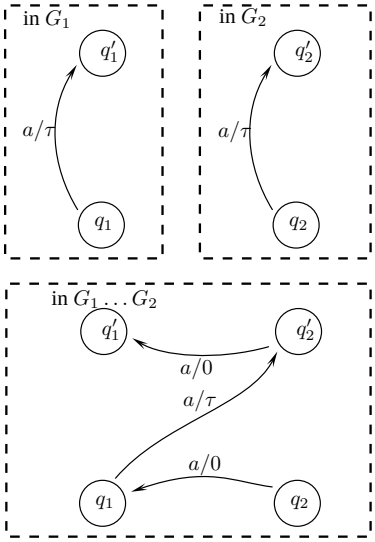


Fig. 3. Illustration of the definition of the asynchronous composition.

**Proof.** It is straightforward to check that for all $a \in A$ each line of $\mu^{G_1 \ldots G_m}(a)$ contains at most one element different from $\varepsilon$ if $\mu_i(a)$ satisfies the same property and, in particular, if $G_i = (Q_i, A, \alpha_i, \mu_i)$, $i = 1, \ldots, m$, are transition-deterministic. Note that there are at least $m$ states $q \in Q_1 \cup Q_2 \cup \ldots \cup Q_m$ such that $\alpha_q^{G_1 \ldots G_m} \neq \varepsilon$.

Let us consider a $m$-bounded timed state-graph $\mathcal{G}$. We associate $m$ safe state-graphs $\mathcal{G}_i$ to $\mathcal{G}$, each corresponding to $\mathcal{G}$ 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 $\mathcal{G}_i$, $i = 1, \ldots, m$. According to proposition 1, these state graphs admit equivalent deterministic (max,+) automata $G_i = (Q_i, A, \alpha_i, \mu_i)$, $i = 1, \ldots, m$, defined on the same set of events, with disjoint sets of states and such that for each $i, j \in \{1, 2, \ldots, m\}$, $i \neq j$, $|Q_i| = |Q_j|$ and $\mu_i(a) = \mu_j(a)$ for all $a \in A$.

The next proposition states that the asynchronous product (cf. definition 4) of these equivalent deterministic automata $G_i$ can be used to evaluate the daters associated with safe state graphs $\mathcal{G}_i$.

*Proposition 3.* Let $x_{G^{G_1 \ldots G_m}}$ be the vector of generalized defined for (max,+) automaton $G^{G_1 \ldots G_m}$ and satisfying the underlying recurrence

$$\begin{cases} x_{G^{G_1 \ldots G_m}}(\epsilon) = \alpha^{G_1 \ldots G_m}, \\ x_{G^{G_1 \ldots G_m}}(wa) = x_{G^{G_1 \ldots G_m}}(w)\mu^{G_1 \ldots G_m}(a). \end{cases} \tag{3}$$

Then $x_{G^{G_1 \ldots G_m}}$ and vector of generalized daters associated to state graphs $\mathcal{G}_i, i = 1, \ldots, m$ are related as follows. For a string

$$\begin{gathered} w = a_{1,1}a_{1,2} \ldots a_{1,m}a_{2,1} \ldots a_{2,m} \ldots a_{k,1} \ldots a_{k,j} \\ \text{with } a_{1,i}a_{2,i} \ldots a_{k,i} \in L_i \text{ for } i = 1, \ldots, m, \end{gathered} \tag{4}$$

we have

$$x_{G^{G_1 \ldots G_m}}(w) = \begin{pmatrix} x_{\mathcal{G}_{j+1}}(a_{1,j+1}a_{2,j+1} \ldots a_{k-1,j+1}) \\ \vdots \\ x_{\mathcal{G}_m}(a_{1,m}a_{2,m} \ldots a_{k-1,m}) \\ x_{\mathcal{G}_1}(a_{1,1}a_{2,1} \ldots a_{k,1}) \\ \vdots \\ x_{\mathcal{G}_j}(a_{1,j}a_{2,j} \ldots a_{k,j}) \end{pmatrix}^T. \tag{5}$$

**Proof.** Vector $x_{G^{G_1\dots G_m}}(w)$ is derived from definition 4 of $G^{G_1\dots G_m}$. We then show by induction that for all $j,k$ we have for $w$ defined according to (4)

$$x_{G^{G_1\dots G_m}}(w) = \begin{pmatrix} \alpha_{j+1}\mu(a_{1,j+1}a_{2,j+1}\dots a_{k-1,j+1}) \\ \vdots \\ \alpha_m\mu(a_{1,m}a_{2,m}\dots a_{k-1,m}) \\ \alpha_1\mu(a_{1,1}a_{2,1}\dots a_{k,1}) \\ \vdots \\ \alpha_j\mu(a_{1,j}a_{2,j}\dots a_{k,j}) \end{pmatrix}^T . \quad (6)$$

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

$$\begin{aligned} x_{G^{G_1\dots G_m}}(a_{1,1}) &= \alpha^{G_1\dots G_m}\mu^{G_1\dots G_m}(a_{1,1}) \\ &= (\ \alpha_2\ \dots\ \alpha_m\ \alpha_1\mu(a_{1,1})\ ). \end{aligned}$$

Let us assume that (6) is satisfied for $w$ given by (4), we then check that (6) is also satisfied for $wa_{k,j+1}$:

$$\begin{aligned} & x_{G^{G_1\dots G_m}}(wa_{k,j+1}) \\ &= x_{G^{G_1\dots G_m}}(w)\mu^{G_1\dots G_m}(a_{k,j+1}) \\ &= \begin{pmatrix} \alpha_{j+2}\mu(a_{1,j+2}a_{2,j+2}\dots a_{k-1,j+2}) \\ \vdots \\ \alpha_m\mu(a_{1,m}a_{2,m}\dots a_{k-1,m}) \\ \alpha_1\mu(a_{1,1}a_{2,1}\dots a_{k,1}) \\ \vdots \\ \alpha_{j+1}\mu(a_{1,j+1}a_{2,j+1}\dots a_{k,j+1}) \end{pmatrix}^T . \end{aligned}$$

In (6), each term $\alpha_j\mu(a_{1,j}a_{2,j}\dots a_{k,j})$ 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 $G_j$ is equivalent to 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 1.*

- Each vector of daters $x_{\mathcal{G}_j}$ in (5) describes the evolution "in isolation" of one of the $m$ tokens in $m$-bounded state graph $\mathcal{G}$. With the considered assumptions on Petri nets, the $m$ tokens evolve independently in $\mathcal{G}$ and equation (5) then allows us to model properly the behavior of this $m$-bounded timed Petri net.
- Note that only strings corresponding to interlacements of words of $L_i$ (as defined in (4)) are considered in (5) 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 $L_1,\ L_2\dots L_m$ (as defined for example in (Sakarovitch, 2003, sec. 6.1)).

*Example 5.* Let us consider 2-bounded state graph $\mathcal{G}$ represented in figure 4. Associated safe state graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are also represented in the figure, and their equivalent deterministic (max,+) automata $G_1$ and $G_2$ are defined by

$$\alpha_1 = (\ e\ \cdot\ ),\ \alpha_2 = (\ \cdot\ e\ ),$$

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

for $i=1,2$.

Their asynchronous composition $G^{G_1\dots G_2}$ is defined by

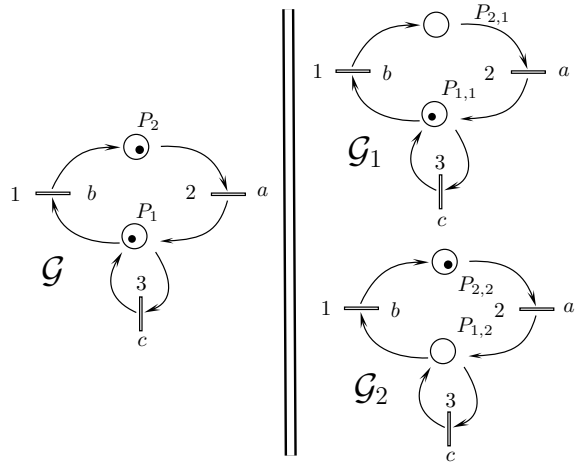


Fig. 4. A 2-bounded state graph $\mathcal{G}$ and associated safe state graphs $\mathcal{G}_1$, $\mathcal{G}_2$.

$$\alpha^{G_1\dots G_2} = (\ e\ \cdot\ \cdot\ e\ ),\quad \mu^{G_1\dots G_2}(a) = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 2 & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix},$$

$$\mu^{G_1\dots G_2}(b) = \begin{pmatrix} \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix},\quad \mu^{G_1\dots G_2}(c) = \begin{pmatrix} \cdot & \cdot & 3 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot \end{pmatrix}.$$

For example, we obtain $x_{G^{G_1\dots G_2}}(cacbba) = (\ \cdot\ 7\,|\,5\ \cdot\ )$. Sub-vector $(\ \cdot\ 7\ )$ (resp. $(\ 5\ \cdot\ )$) corresponds to $x_{\mathcal{G}_1}(ccb)$ (resp. $x_{\mathcal{G}_2}(aba)$) which is the dater associated to $\mathcal{G}_1$ (resp. $\mathcal{G}_2$), and 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 $\mathcal{G}_2$).

## 5. SYNCHRONOUS COMPOSITION OF TRANSITION-DETERMINISTIC (MAX, +) AUTOMATA

Synchronous composition of transition-deterministic (max,+) automata is now defined. To have a synthetic presentation, we consider only two (max,+) automata $G_1 = (Q_1, A_1, \alpha_1, \mu_1)$ and $G_2 = (Q_2, A_2, \alpha_2, \mu_2)$ in this section, knowing that all the definitions and results can easily be extended to the composition of more than two automata.

*Definition 5.* We denote $G_1||G_2 = (Q, A, \alpha, \mu)$ the automaton resulting from the synchronous product of $G_1$ and $G_2$ defined by:

$$Q = Q_1 \cup Q_2, A = A_1 \cup A_2, \alpha = (\ \alpha_1\ \alpha_2\ ),$$

$$\mu(a) = \left(\begin{array}{c|c} \mu_{11} & \mu_{12} \\ \hline \mu_{21} & \mu_{22} \end{array}\right),$$

in which each block $\mu_{ij}$ is a $|Q_i| \times |Q_j|$ matrix defined as follows:

- for $i = j$ :

$$\mu_{ii} = \begin{cases} \mu_i(a), & \text{if } a \in A_i; \\ I_{|Q_i|}, & \text{otherwise.} \end{cases}$$

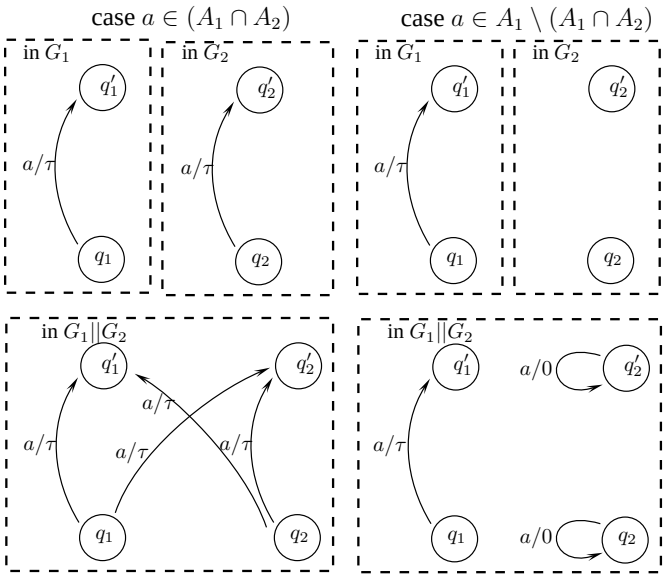


Fig. 5. Illustration of the definition of the synchronous composition.

- for $i \neq j$ :

$$[\mu_{ij}]_{kl} = \begin{cases} \tau, \text{ if } a \in A_1 \cap A_2 \text{ and } \exists q'_k \in Q_i, q'_l \in Q_j \\ \quad \text{s.t. } [\mu_i(a)]_{q_k,q'_k} = [\mu_j(a)]_{q'_l,q_l} = \tau (\neq \varepsilon) \\ \quad \text{with } \tau \neq 0; \\ \varepsilon, \text{ otherwise.} \end{cases}$$

Note that figure 5 aims at illustrating this definition.

The next proposition states the modeling power obtained by means of synchronous product of transition deterministic (max,+) automata. It is expressed as a Petri net resulting from the merging of common transitions in two sub-nets.

*Proposition 4.* Let $G_1$ and $G_2$ be the two transition-deterministic automata obtained to model two bounded state graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ (cf. definition 4 and proposition 3). We denote $\mathcal{G}_1||\mathcal{G}_2$ the Petri net obtained from $\mathcal{G}_1$ and $\mathcal{G}_2$ by merging the transitions corresponding to common events $a \in A_1 \cap A_2$ of $G_1$ and $G_2$. We have :

$$x_{\mathcal{G}_1||\mathcal{G}_2}(w) = x_{G_1||G_2}(w),$$

for all $w \in A^*$ corresponding to a firing sequence in $\mathcal{G}_1||\mathcal{G}_2$ such that $P_1(w)$ ($P_1$ is the projection on the language defined on $A_1$ [2]) and $P_2(w)$ are possible firing sequences in $\mathcal{G}_1$ and $\mathcal{G}_2$, and in particular if $G_1$ results from the asynchronous composition of deterministic automata $G_{1,1}$, ..., $G_{1,m}$ then $P_1(w)$ must satisfy the decomposition (4) on languages $L_{G_{1,i}}$, $i = 1, \ldots, m$.

**Proof.** We have by construction $x_{\mathcal{G}_1||\mathcal{G}_2}(\epsilon) = x_{G_1||G_2}(\epsilon)$. Let us assume that equality $x_{\mathcal{G}_1||\mathcal{G}_2}(w) = x_{G_1||G_2}(w)$ is true at the conclusion of event sequence (resp. firing sequence) $w$ in $\mathcal{G}_1||\mathcal{G}_2$ (resp. in $G_1||G_2$), and let us check that $x_{\mathcal{G}_1||\mathcal{G}_2}(wa) = x_{G_1||G_2}(wa)$, $\forall a$.

[2] The natural projection, from $A$ to $A_1 \subseteq A$, is such that:

$$P_1(a) = \begin{cases} a \text{ if } a \in A_1, \\ \varepsilon \text{ if } a \in A \setminus A_1, \end{cases}$$

$P_1$ preserves concatenation and can be extended to words (for all word, $P_1$ removes events (letters) in $A \setminus A_1$).

Let us consider first that $a \in A_1 \cap A_2$ corresponds to a shared event in $G_1||G_2$, that is a common transition $a \in \mathcal{T}$ in $\mathcal{G}_1||\mathcal{G}_2$. As $G_1$ and $G_2$ are transition-deterministic, there exist $\{q_1, q'_1\} \in Q_1$ and $\{q_2, q'_2\} \in Q_2$ such that

$$[\mu(a)]_{q_1 q'_1} = [\mu(a)]_{q_1 q'_2} = [\mu(a)]_{q_2 q'_1} = [\mu(a)]_{q_2 q'_2} = \tau,$$

with $\tau \neq \varepsilon$, $\tau \neq 0$ and all other coefficients of lines $q_1$ and $q_2$ are equal to $\varepsilon$ (see the left part of fig. 5 for an illustration). We then have

$$\begin{aligned} [x_{G_1||G_2}(wa)]_{q'_1} &= \max\left(\tau + [x_{G_1||G_2}(w)]_{q_1}, \tau + [x_{G_1||G_2}(w)]_{q_2}\right), \\ &= \max\left(\tau + [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_1}, \tau + [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_2}\right), \end{aligned}$$

$$\begin{aligned} [x_{G_1||G_2}(wa)]_{q'_2} &= \max\left(\tau + [x_{G_1||G_2}(w)]_{q_1}, \tau + [x_{G_1||G_2}(w)]_{q_2}\right), \\ &= \max\left(\tau + [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_1}, \tau + [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_2}\right). \end{aligned}$$

This enables us to properly model the synchronization operating when transition $a$ is fired in $\mathcal{G}_1||\mathcal{G}_2$ (that is $x_{G_1||G_2}(wa) = x_{\mathcal{G}_1||\mathcal{G}_2}(wa)$).

Let us now consider that $a$ is a private event, say $a \in A_1 \setminus (A_1 \cap A_2)$ (see the right part of fig. 5 for an illustration). We then have

$$[x_{G_1||G_2}(wa)]_{q'_1} = \tau + [x_{G_1||G_2}(w)]_{q_1} = \tau + [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_1}, \quad (7)$$

for a $q'_1 \in Q_1$ and $q'_1 \in Q_1$, and

$$[x_{G_1||G_2}(wa)]_{q_2} = [x_{G_1||G_2}(w)]_{q_2} = [x_{\mathcal{G}_1||\mathcal{G}_2}(w)]_{q_2}, \quad (8)$$

for all $q_2 \in Q_2$. This enables us to properly model that when transition $a$ is fired in $\mathcal{G}_1||\mathcal{G}_2$ then the corresponding dater in $\mathcal{G}_1$ is shifted due to the firing duration and that daters in $\mathcal{G}_2$ are not affected.

We have the following property for sequences composed exclusively of private events.

*Proposition 5.* For all $u, v \in (A \setminus (A_1 \cap A_2))^*$, if $P_1(u) = P_1(v)$ and $P_2(u) = P_2(v)$ we then have $\mu(u) = \mu(v)$.

**Proof.** Due to the block-diagonal (with an identity block) structure of $\mu$ for private events, we have $\forall a \in A_1 \setminus (A_1 \cap A_2)$, $\forall b \in A_2 \setminus (A_1 \cap A_2)$, $\mu(ab) = \mu(ba)$. This observation can be easily extended to show the statement of the proposition.

Now, any $w \in A^*$ can be decomposed as $w = v_0 a_1 v_1 \ldots a_n v_n$ where $a_i \in A_1 \cap A_2$, $i = 1, \ldots, n$ are common events and $v_i \in (A \setminus (A_1 \cap A_2))^*$ are sequences of private events. Then, it should be clear that for any other word $w' = v'_0 a_1 v'_1 \ldots a_n v'_n$ such that $P_1(v_i) = P_1(v'_i)$ and $P_2(v_i) = P_2(v'_i)$ we have $\mu(w) = \mu(w')$ enabling us to model that subsequences of private events are executed simultaneously and asynchronously in each components.

*Example 6.* Let the Petri net depicted in figure 6 be studied as the merging of two bounded state graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ having a common transition $a$. The 2-bounded state graph, denoted here $\mathcal{G}_1$, has been described by means of a (max,+) automaton resulting from an asynchronous composition in example 5. In a similar manner, 3-bounded state graph $\mathcal{G}_2$ can be modeled by (max,+) automaton $(Q_2, A_2, \alpha_2, \mu_2)$ with

$$\alpha_2 = (\, e \;\; e \;\; e \,), \quad \mu_2(a) = \begin{pmatrix} \cdot & \cdot & 2 \\ e & \cdot & \cdot \\ \cdot & e & \cdot \end{pmatrix}, \quad \mu_2(d) = \begin{pmatrix} \cdot & \cdot & 4 \\ e & \cdot & \cdot \\ \cdot & e & \cdot \end{pmatrix}.$$

According to definition 5, the synchronous composition of these (max,+) automata (a graphical representation is fig. 7) is

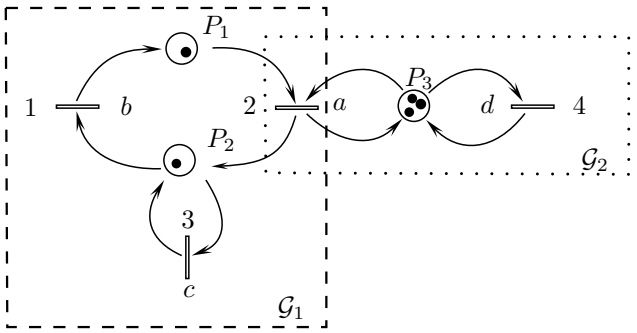


Fig. 6. A bounded timed Petri net.

$$\alpha = (\, e \cdot\cdot e \,|\, e\ e\ e\,), \quad \mu(b) = \left(\begin{array}{ccc|ccc} \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & e \end{array}\right),$$

$$\mu(c) = \left(\begin{array}{ccc|ccc} \cdot & \cdot & 3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & e \end{array}\right), \mu(d) = \left(\begin{array}{ccc|ccc} e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & e & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & 4 \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & e \end{array}\right),$$

$$\mu(a) = \left(\begin{array}{ccc|ccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 2 & \cdot & \cdot & 2 \\ e & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & e & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & 2 & \cdot & \cdot & 2 \\ \cdot & \cdot & \cdot & e & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & e & \cdot \end{array}\right).$$

For example, we have

$$\alpha\mu(cdab) = \alpha\mu(dcab) = (\, 2 \cdot\cdot 4\ 0\ 4\ 2\,).$$

Sub-vector $(\, 2 \cdot\,)$ (resp $(\, \cdot\ 4\,)$) indicates that a token arrives at time instant 2 (resp. 4) in $P_2$ (resp. in $P_1$) at the conclusion of the firing of transition $a$ (resp. the firing sequence $cb$) . Sub-vector $(\, 0\ 4\ 2\,)$ gives the last arrival dates of the three tokens in $P_3$ (the first one has not contributed to a firing, the second has contributed to a firing of $d$ and the third has contributed to a firing of $a$).

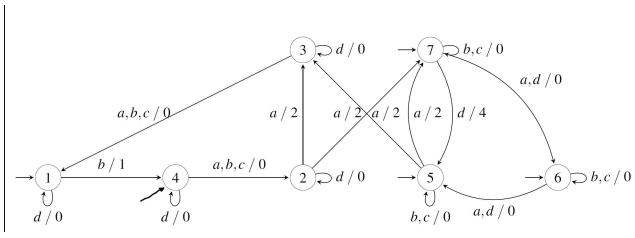


Fig. 7. Automaton associated with Petri net of fig. 6

## 6. CONCLUSION

Both asynchronous and synchronous products of (max, +)-automata are proposed and their modeling power in terms of classes of timed Petri nets has been discussed. When they are used in a joint way, timed Petri nets obtained by merging common transitions of bounded timed state graphs can be studied as illustrated in example 6. Future investigations should make clear the whole methodology using our compositions for the modeling a large class of TDES (order in which the compositions should be used,...). The proposed compositions also open the way to the compositional verification and performance evaluation of large TDES.

## REFERENCES


Baccelli, F., Cohen, G., Olsder, G.J., and Quadrat, J.P. (1992). *Synchronization and Linearity.* Wiley.

Buchholz, P. and Kemper, P. (2003). Weak bisimulation for (max/+) automata and related models. *J. Autom. Lang. Comb.*, 8, 187–218.

Cassandras, C.G. and Lafortune, S. (2006). *Introduction to Discrete Event Systems.* Springer-Verlag New York, Inc.

David, R. and Alla, H. (2010). *Discrete, continuous, and hybrid Petri Nets (2nd edition).* Springer, Paris.

Gaubert, S. (1995). Performance Evaluation of (max,+) Automata. *IEEE Transaction on Automatic Control*, vol. 40(12), 2014–2025.

Gaubert, S. and Mairesse, J. (1999). Modeling and Analysis of Timed Petri Nets using Heaps of Pieces. *IEEE Transaction on Automatic Control*, vol. 44(4), 683–698.

Houssin, L. (2011). Cyclic jobshop problem and (max,plus) algebra. In *18th IFAC World Congress*, 2717–2721. Milan, Italy.

Klimann, I., Lombardy, S., Mairesse, J., and Prieur, C. (2004). Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327.

Komenda, J., Lahaye, S., and Boimond, J.L. (2009a). Le produit synchrone des automates (max,+). *Special issue of Journal Européen des Systèmes Automatisés (JESA)*, vol. 43(7), 1033–1047.

Komenda, J., Lahaye, S., and Boimond, J.L. (2009b). Supervisory Control of (max,+) Automata: A Behavioral Approach. *Discrete Event Dynamic Systems*, vol. 19(4), 525–549.

Lahaye, S., Komenda, J., and Boimond, J.L. (2012). Modélisation modulaire à l'aide d'automates (max,+). Accepted at CIFA 2012 (Grenoble, France).

Sakarovitch, J. (2003). *Éléments de théorie des automates.* Vuibert.

Su, R., van Schuppen, J., and Rooda, J. (2012). The synthesis of time optimal supervisors by using heaps-of-pieces. *IEEE Transaction on Automatic Control*, vol. 57(1), 105–118.

Su, R. and Woeginger, G.J. (2011). String execution time for finite languages: Max is easy, min is hard. *Automatica*, 47(10), 2326–2329.