MODELS COMBINATION IN (MAX,+) ALGEBRA FOR THE IMPLEMENTATION OF A SIMULATION AND ANALYSIS SOFTWARE

Sébastien Lahaye, Laurent Hardouin and Jean-Louis Boimond

This paper presents a modeling methodology in (max, +) algebra which has been developed in order to implement a modulary software for the simulation and the analysis of electronic cards production lines. More generally, this approach may be applied to hybrid flowshop type manufacturing systems.

1 INTRODUCTION

The first results about (max,+) algebra appeared [4] more than 20 years ago. In particular this approach allows apprehending more easily the modeling of synchronization and delay phenomena. Many algebraic tools have been developed in order to work out a linear theory for a certain class of discrete events systems. The main class corresponds to the systems which can be represented by timed event graphs (TEGs). A TEG is a timed Petri net [11] for which each place admits only one upstream transition and one downstream transition. This means that all potential conflicts in using tokens in places have already been arbitrated by some predefined policy. These limitations are certainly restrictive for most applications, nevertheless they can generally be satisfied by making some design and scheduling decisions at an upper hierarchical level. The theory for these systems in (max,+) or (min,+) algebra presents strong analogies with the conventional linear system theory. Indeed concepts such as state representation, transfer matrices, optimal control, correctors synthesis and identification theory have been introduced [1], [2], [3], [10].
This theory finds applications in particular for the study of manufacturing systems. This paper presents an application of these theoretical results developed in collaboration with an industrial partner. Indeed, we have developed a modulary software package dedicated to the simulation and the analysis of electronic cards production lines. We present the modeling methodology in (max, +) algebra adopted in order to implement this modulary software. More generally, this approach may be applied to hybrid flowshop type manufacturing systems where products cross successive stages arranged in a linear manner, and where several products can be processed simultaneously at a given stage.

In the first place, we propose to model each stage of the assembly line by a timed event graph with variable holding times. In order to ease this first modeling step, a functioning rule for Petri nets, which forces places to operate as FIFO channels, is introduced. From this graphical model, representations in $(\max, +)$ algebra are then deduced.
The second step consists in combining the different models obtained for the successive stages of the line. A simple combination in series is not sufficient to accurately model the dynamic behavior of the line. A method for combination, called concatenation, is then proposed to take into account the substantial correlations between the assembled elementary systems. It consists in splitting each elementary system into two subsystems in order to model, on the one hand, the parts flow from input to output, and on the other hand, the information flow from output to input. This approach allows taking into account, not only the dynamics of each stage, but also its loading effects on the assembly line. A simulation algorithm can directly be deduced from the resulting model. Standard representations in $(\max, +)$ algebra can also be established for the analysis of the system. The outline of the paper is as follows. In §2, we recall basic elements of $(\max, +)$ algebra. In §3, the modeling technique in $(\max, +)$ algebra is presented. Section 4 is devoted to the combination of systems. More precisely, the concatenation of elementary systems is presented for the modeling of assembly lines.

2 PRELIMINARIES

We consider the semi-field $(\mathbb{R} \cup \{-\infty\}, \oplus, \otimes)$ in which the law $\oplus$ is max, and $\otimes$ is the usual addition. We denote respectively $\varepsilon = -\infty$ and $e = 0$ the neutral elements of $\oplus$ and $\otimes$. The element ε is absorbing for $\otimes$. The law $\oplus$ is idempotent, *i.e.*, $a \oplus a = a$. $(\mathbb{R} \cup \{-\infty\}, \oplus, \otimes)$ is an idempotent semi-ring or *dioid* [1], [2], and is usually referred to as *(max,+) algebra*. We shall denote it by $\mathbb{R}_{\max}$.
In the following, we shall consider vectors and matrices with entries in $\mathbb{R}_{\max}$. The product of a vector $u \in \mathbb{R}^n_{\max}$ by a scalar $a \in \mathbb{R}_{\max}$ is defined as

$$(a \otimes u)_i = a \otimes u_i = a + u_i.$$

The sum and product of matrices are defined conventionally, replacing $+$ and $\times$ by $\oplus$ and $\otimes$, respectively. Let $A, B \in \mathbb{R}^{n \times n}_{\max}$,

$$\begin{aligned}(A \oplus B)_{ij} &= A_{ij} \oplus B_{ij} \\ (A \otimes B)_{ij} &= \bigoplus_{l=1}^{n} A_{il} \otimes B_{lj} = \max_{1 \le l \le n} (A_{il} + B_{lj}).\end{aligned}$$

The matrix-vector product is defined in a similar way.
Most of the time, the symbol '$\otimes$' is omitted as is the case in conventional algebra.
A system $\mathcal{S}$ is a mapping from the set of admissible input signals to the set of admissible output signals. In this paper, the signals of interest are non-decreasing functions: $\mathbb{Z} \mapsto \mathbb{R}_{\max}$.

3 MODELING IN THE (MAX,+) ALGEBRA

In this section, we recall how systems involving synchronization phenomena can be modeled by linear equations in (max,+) algebra. More particularly, the focus will be on assembly lines and an example of electronic cards production line. In a first place, successive stages of assembly lines (machines, conveyors, etc) are modeled by TEGs with variable holding times. With the aim of easing this modeling, we introduce a functioning rule for Petri nets which forces places to operate as FIFO channels. Starting from this graphical model, representations in the $(\max, +)$ algebra are presented in a second place.

3.1 Petri net model

We denote by $\mathcal{P}$ (respectively, $\mathcal{T}$) the finite set of places (respectively, transitions[1]) of a TEG, and $M_p \in \mathbb{N}$ the number of tokens being initially in place $p \in \mathcal{P}$; $p^\bullet$ (respectively, ${}^\bullet p$) refers to the output transition (respectively, input transition) of p. We define similarly the sets $t^\bullet$, ${}^\bullet t$ as the set of output places, and the set of input places, of transition $t \in \mathcal{T}$.
We call *holding time* the minimum amount of time tokens have to stay in a place: the token indexed k in place p incurs the holding time denoted $\tau_p(k)$.

Definition 1 *We define the earliest FIFO functioning rule of a TEG as follows.*

1. *A transition t fires as soon as each place upstream t contains at least one available token.*
2. *We denote $t(n)$ the date at which transition t fires for the $(n+1)$-st time. This firing consumes one token in each upstream place and produces one token in each downstream place. A token added in place $p \in t^\bullet$ at time $t(n)$ is indexed k with $k = n + M_p$ and becomes available for transition $p^\bullet$ from instant $\max\limits_{0 \le i \le n} \{t(i) + \tau_p(i + M_p)\}$.*

The only originality in the proposed functioning rule concerns the tokens availability. Classically, the token indexed k in place p is said to be available as soon as its holding time $\tau_p(k)$ is over [1, ch. 2]. The above definition of availability ensures besides that tokens cannot overtake one another when traversing places (places operate as FIFO channels). It notably enables to easily model elements (machines, conveyor,...) of mixed-model assembly lines (which are intrinsically overtake free) on which several parts can be simultaneously handled.
With the aim of corroborating this point, let us consider for example a machine of insertion of electronic components modeled by the TEG functioning according to the FIFO rule represented on figure 1. Transition u_1 represents the arrival of electronic cards in the upstream storage area, x_1 the arrival of unprocessed cards in the machine, transition v_1 the arrival of components in machine, x_2 the beginning

[1]In the following, we classically partition the set of transitions $\mathcal{T} = \mathcal{U} \cup \mathcal{X} \cup \mathcal{Y}$, where $\mathcal{U}$ is the set of transitions with no predecessors (input transitions), $\mathcal{Y}$ is the set of transitions with no successors (output transitions), and $\mathcal{X} = \mathcal{T} \setminus (\mathcal{U} \cup \mathcal{Y})$ (internal transitions).

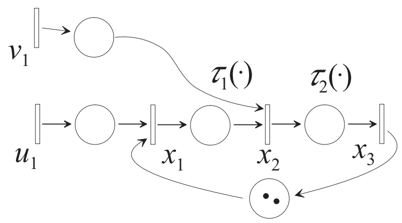

Figure 1: TEG example

of the insertion process which requires a card and the presence of components, x_3 the output of processed cards in the downstream area. The sequence of holding time $\tau_1(\cdot)$ correspond to the preparation times of the successive cards and $\tau_2(\cdot)$ is equal to their processing times. The two tokens indicate that the machine can process two cards simultaneously. Two cards, processed simultaneously, cannot overtake one another in the machine even if their processing times are different, and the FIFO functioning rule enables to naturally model this behavior.

3.2 Representations in the (max,+) algebra

The modeling methodology for TEGs functioning according to the FIFO rule is similar to the one presented in [1, chap. 2] for TEGs with the classical rule.
With each transition $t \in T$ we associate a *dater variable* also denoted t: $t(k)$ denotes the date of the $(k+1)$-st firing of transition t. By convention we have $t(k) = +\infty$ if t fires less than $(k+1)$ times, and, $t(k) = \varepsilon$ for all $k \leq 0$. The sequences of holding times $\tau_p(k)$, $p \in \mathcal{P}$, $k \in \mathbb{Z}$ are assumed to be given nonnegative and finite integers.
For sake of briefness, we suppose here that initial tokens are available from instant $-\infty$. Towards the modeling of TEGs in the $(\max, +)$ algebra, this comes down considering canonical initial conditions (see in [1, §2.5] how to deal with 'non-zero' initial conditions).

Assertion 1 *The dater variables of a TEG functioning according to the FIFO rule satisfy the following equation:*

$$t(k) = \bigoplus_{\{t' = {}^\bullet p | p \in {}^\bullet t\}} \bigoplus_{i \leq k} \left[\tau_p(i) \otimes t'(i - M_p)\right], \; k \in \mathbb{Z},$$

or equivalently

$$t(k) \quad = \bigoplus_{\{t' = {}^\bullet p | p \in {}^\bullet t\}} \left[\tau_p(k) \otimes t'(k - M_p)\right] \oplus \; t(k-1) \; , \; k \in \mathbb{Z}.$$

Remark 1: Eq. (1) does not model the behavior of a TEG (with the classical functioning rule) for which each transition is recycled[2]. Consider for example the simple

[2]A transition is said to be recycled if $\{p \in \mathcal{P} \mid p \in {}^\bullet t, p \in t^\bullet, M_p = 1\} \neq \varnothing$

Figure 2: Illustration of remark 1

TEG functioning according to the FIFO rule represented on figure 2.(a). Its dynamic behavior is described by equation

$$t_2(k) = \tau(k)t_1(k) \oplus t_2(k-1) \quad (1)$$

and with

k	$t_1(k)$	$\tau(k)$
0	1	3
1	2	1
$\vdots$	$\vdots$	$\vdots$

we have $t_2(0) = 4$ and $t_2(1) = 4$. Eq. 1 does not model the dynamic behavior of the TEG represented on figure 2.(b) functioning according to the usual rule. Indeed, even if transition t_2 is recycled, with the classical rule, tokens may overtake one another in the timed place, and with the previous scenario, the first and second firing of transition t_2 occur respectively at time 3 and 4.

3.2.1 State model

We denote by u (respectively x, y) the vector of input (respectively, state, output) daters t, $t \in \mathcal{U}$ (respectively, $\mathcal{X}$, $\mathcal{Y}$). As for TEGs, one can obtain after several manipulations the following *standard state model* [7]:

$$\left\{ \begin{array}{lcl} x(k) & = & A(k-1)x(k-1) \oplus B(k)u(k) \\ y(k) & = & C(k)x(k) \end{array} \right. \quad (2)$$

where $A(k)$, $B(k)$, $C(k)$, $k \in \mathbb{Z}$, are matrices of respective dimensions $n \times n$, $n \times p$, $q \times n$ [3] with entries in $\mathbb{R}_{\max}$. In particular, an entry $[A(k)]_{ij}$ is equal to ε if, and only if, transition labeled x_j does not belong to the set $^\bullet(^\bullet x_i)$.

3.2.2 Input-output relationship

The first recursive equation of (2) can also be written

$$x(k) = \Phi(k,k_0)x(k_0) \oplus \bigoplus_{j=k_0+1}^{k} \Phi(k,j)B(j)u(j),\ k \geq k_0, \quad (3)$$

[3] We have $q = \mid \mathcal{Y} \mid$, but n (resp. p) may be greater than $\mid \mathcal{X} \mid$ (resp. $\mid \mathcal{U} \mid$) because the state vector x (resp. input vector u) may have been extended to obtain the standard Eqs. (2).

where the *state-transition matrix* $\Phi(k,i)$ is given by

$$\Phi(k,i) = \begin{cases} \text{not defined} & ,i > k \\ e \quad (\text{identity element of } \overline{\mathbb{Z}}_{\max}{}^{n\times n}) & ,i = k \\ A(k-1)\otimes A(k-2)\otimes \cdots \otimes A(i) & ,i < k \end{cases}$$

Remark 2: The state-transition matrix satisfies the composition property

$$\Phi(k,i) = \Phi(k,l)\otimes\Phi(l,i), \text{ where } k \geq l \geq i,$$

and in particular for $k > i$:

$$\Phi(k,i) = A(k-1)\Phi(k-1,i)\ ,\ \Phi(k,i) = \Phi(k,i+1)A(i).$$

The *input-output relationship* is deduced from Eq. (3) with $x(k_0) = u(k_0) = \varepsilon$ for $k_0 \leq 0$, and is given by:

$$\forall k \in \mathbb{Z}, \quad y(k) = \bigoplus_{j\in\mathbb{Z}} h(k,j)u(j) \tag{4}$$

where h is called the *impulse response* and is defined by:

$$h(k,j) = \begin{cases} C(k)\Phi(k,j)B(j) & ,k \geq j, \\ \varepsilon \ \ (q \times p \text{ matrix of } \varepsilon) & ,k < j, \end{cases} \tag{5}$$

Remark 3: For conventional discrete-time linear time-varying systems [6], the input-output relationship is given by: $y(k) = \sum_{j=-\infty}^{k} h(k,j)u(j)$. The analogy with formula (4) should be clear.

Example 1: Let us consider again a machine of insertion of components modeled by the TEG represented on figure 1. Its dynamic behavior is described by the state equation[4] (2) with:

$$A(k-1) = \begin{pmatrix} e & \varepsilon & \varepsilon & e \\ \tau_1(k) & e & \varepsilon & \tau_1(k) \\ \tau_1(k)\tau_2(k) & \tau_2(k) & e & \tau_1(k)\tau_2(k) \\ \varepsilon & \varepsilon & e & \varepsilon \end{pmatrix},$$

$$B(k) = \begin{pmatrix} e & \varepsilon \\ \tau_1(k) & e \\ \tau_1(k)\tau_2(k) & \tau_1(k) \\ \varepsilon & \varepsilon \end{pmatrix},$$

$$\text{and } x(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_3(k-1) \end{pmatrix},\ u(k) = \begin{pmatrix} u_1(k) \\ v_1(k) \end{pmatrix}$$

[4]An output equation is not required since the TEG has no output transition.

4 MODEL FOR ASSEMBLY LINES

Assembly lines, in particular electronic cards production lines, are composed of several stages (machines, such as the components insertion machine presented in the previous section, conveyors, etc) arranged in a linear manner, and each card crosses these successive stages of the line in the same order (manufacturing system of flow-shop type). A whole model for these systems can be obtained by establishing a TEG representing the whole line, and by obtaining its representations in the $(\max, +)$ algebra. The modeling approach presented in this section rather consists in modeling each stage separately, and besides in connecting these elementary models in a correct manner. Such a broken down modeling approach is appropriate if one aims to build a *modulary* simulation software for these systems. A library of elementary models (representing different machines, conveyors, etc) can then be defined, and a simulation of the line is then obtained by simply connecting these models according to any order.

4.1 Elementary systems in series

The simplest principle for considering the assembly of models is to put in series the elementary systems as represented on figure 3.

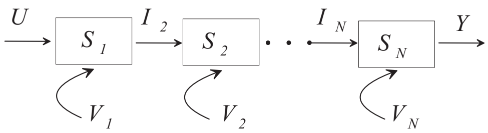

Figure 3: Series combination of elementary systems

This principle assumes that the subsystem inputs do not depend on inputs in downstream systems.
As in conventional systems theory, starting from the state model (or impulse response) of each elementary system, the representation (state model or impulse response) resulting from their connection in series can be established [9].
Regarding an assembly line, cascading elementary systems comes down to assuming that there is a storage area with an infinite capacity between each subsystem.
Example 2: Figure 6 presents the resulting series combination of two TEGs. The first system (system 1) represents a machine of insertion of electronic components previously described in section 3.1. The second TEG (system 2) represents a furnace allowing the welding of components on the cards. A card is transported, through furnace, on a conveyor advancing at constant speed. The card is then conveyed towards a buffer ventilated in order to be cooled. Holding time τ_3 is the minimum interval between the entry of two cards in the furnace, τ_4 is the transportation time in the furnace, τ_5 is the transfer time towards the cooling buffer, and $\tau_6(\cdot)$ represents the minimum sojourn time in the cooling buffer. The number of tokens ν_3 corresponds to the maximum number of cards authorized between the input of

the furnace and the output of the buffer. ν_2 represents the capacity of the conveyor between furnace and buffer. The bold place represents the storage area with an infinite capacity between the elementary systems.

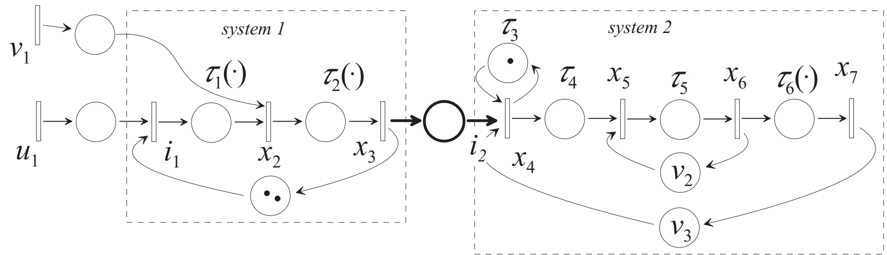

Figure 4: TEG modeling two elementary systems in series

In general, the model obtained by cascading elementary systems is not sufficiently accurate for assembly lines. In fact, the assumption of infinite capacity stocks between elementary systems, which corresponds to a perfect impedance matching when electrical or acoustical systems are combined in series, is not realistic. Besides, in pull-flow systems, such internal stocks are usually restricted as much as possible in order to limit works in process. In the following section, we propose another principle for the assembly of models which allows taking into account the possible correlations between the assembled elementary systems.

4.2 Concatenation of elementary systems

To consider the coupling between elementary systems, more exactly, to consider the realistic case where the inputs of the systems are constrained by the inputs in the downstream systems, we assume that each elementary system $\mathcal{S}_n$ is split into two subsystems $\mathcal{S}_n^a$ and $\mathcal{S}_n^b$ which are connected as represented on figure 5. The subsystem $\mathcal{S}_n^b$ describes the information flow from upstream to downstream, and conversely $\mathcal{S}_n^a$ describes the information flow from downstream to upstream.

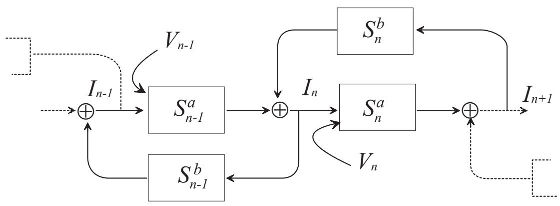

Figure 5: Model concatenation

The input of each subsystem $\mathcal{S}_n^a$ corresponds to the vector of signals

$$U_n = \begin{pmatrix} I_n \\ V_n \end{pmatrix}$$

in which V_n corresponds to the inputs acting directly on the system and I_n the inputs depending both on outputs of upstream subsystem and on input of downstream subsystem:

$$I_n = \mathcal{S}_n^b(I_{n+1}) \oplus \mathcal{S}_{n-1}^a(U_{n-1}) \, .$$

By convention, we set: $I_1 = \mathcal{S}_1^b(I_2) \oplus u$. The output y of the last subsystem numbered N is simply equal to $y = I_{N+1} = \mathcal{S}_N^a(I_N)$.

This model combination leads easily to an algorithm for the simulation of the production line. Indeed, assuming that each stage of the line is initially idle (no parts remain in the line from a past functioning), in other words, that the TEG modeling the system is in canonical initial conditions (see [1]), the state equation (2) of each subsystem $\mathcal{S}_j^b$ can also be written

$x_j^b(k) = A_j^b(k-1)x_j^b(k-1) \oplus B_j^b(k)I_{j+1}(k-\nu_j)$

in which $\nu_j > 0$ corresponds to the number of parts which can be processed simultaneously at stage j. This then leads to explicit recurrences in the following algorithm used for simulation.

$\forall j \in [1, n]$, $x_j^a(-1) = x_j^b(-1) = \varepsilon$, $I(-\nu_j) = I(-\nu_j + 1) = \ldots = I(-1) = \varepsilon$
For k from 0 to k_f
  For j from 1 to N
    $x_j^b(k) = A_j^b(k-1)x_j^b(k-1) \oplus B_j^b(k)I_{j+1}(k-\nu_j)$
    $x_{j-1}^a(k) = A_{j-1}^a(k-1)x_{j-1}^a(k-1) \oplus B_j^a(k)U_{j-1}(k)$
    $I_j(k) = C_j^b(k)x_j^b(k) \oplus C_{j-1}^a(k)x_{j-1}^a(k)$
    $U_j(k) = (I_j(k)^\top \, , \, V_j(k)^\top)^\top$

The use of this algorithm (instead of a recurrence on the global state model) has several advantages for the simulation:

- *modularity*: the addition or deletion of elementary systems only requires to modify a portion of the algorithm (instead of re-establishing the whole state model).
- *optimization of the calculating times*: in assertion 2, an example of construction of matrices for the global state model is given. The resulting matrices contain several null blocks (whose entries are equal to ε). The proposed algorithm avoids handling needlessly these blocks for the simulation.

The representations (state-space representation or impulse response) of the resulting system can be established. Because of the lack of space, only a state-representation for the system resulting from the concatenation of two elementary subsystems is given in the following assertion.

Assertion 2 *Let $\left(A_i^a(k),\, B_i^a(k),\, C_i^a(k)\right)$ and $\left(A_i^b(k),\, B_i^b(k),\, C_i^b(k)\right)$, $k \in \mathbb{Z}$, be the state-space realizations of $\mathcal{S}_i^a$ and $\mathcal{S}_i^b$, $i = 1, 2$. A state-space realization of $\mathcal{S}$ resulting from the concatenation of $\mathcal{S}_1$ and $\mathcal{S}_2$ is given by:*

$$A(k) = \begin{pmatrix} A_1^a(k) & B_1^a(k)C_1^b(k) & \cdot & \cdot \\ B_1^b(k)C_1^a(k) & A_1^b(k) & \cdot & B_1^b(k)C_2^a(k) \\ B_2^a(k)C_1^a(k) & \cdot & A_2^a(k) & B_2^a(k)C_2^b(k) \\ \cdot & \cdot & B_2^b(k)C_2^a(k) & A_2^b(k) \end{pmatrix}$$

$$B(k) = \begin{pmatrix} B_1^a(k) \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}, \; C(k) = \begin{pmatrix} \cdot & \cdot & C_2^a(k) & \cdot \end{pmatrix}$$

in which entries denoted "·" are blocks of ε*.*

This state-space representation can be used to establish a just-in-time control for the manufacturing system [7], and/or to compute its cycle time when the production is repetitive [8]. It also makes it possible to simulate the system, but, as pointed out previously, the proposed modular approach is preferable for simulation.
Example 3: An adequate model for the portion of production line considered previously, is the concatenation of the TEGs modeling the machine of insertion of components and the furnace represented on figure 6. More precisely, the transition denoted i_2 on figure 6 is the merge of transitions x_3 and x_4 on figure 6. The bold line parts of the TEGs correspond to the subsystems $\mathcal{S}_1^b$ and $\mathcal{S}_2^b$, while the thin line parts represent the subsystems $\mathcal{S}_1^a$ and $\mathcal{S}_2^a$. Let us assume that $\nu_2 = 1$ and $\nu_3 = 2$.

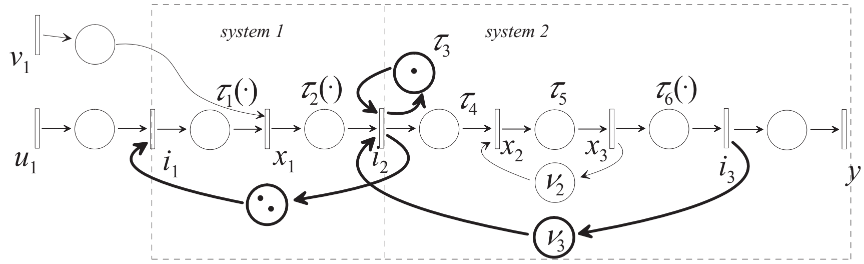

Figure 6: TEG modeling two elementary systems concatenated

The state model of this system is defined in assertion 2 with:

$$x_1^a(k) = \begin{pmatrix} i_1(k) \\ x_1(k) \end{pmatrix}, u_1^a(k) = \begin{pmatrix} u(k) \\ v_1(k) \end{pmatrix},$$

$$A_1^a(k) = \begin{pmatrix} \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{pmatrix}, B_1^a(k) = \begin{pmatrix} e & \varepsilon \\ \tau_1(k) & e \end{pmatrix}, C_1^a(k) = \begin{pmatrix} \varepsilon & \tau_2(k) \end{pmatrix},$$

$$x_1^b(k) = \begin{pmatrix} i_2(k) \\ i_2(k-1) \\ i_2(k-2) \end{pmatrix}, u_1^b(k) = i_2(k),$$

$$A_1^b(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon \end{pmatrix}, B_1^b(k) = \begin{pmatrix} e \\ \varepsilon \\ \varepsilon \end{pmatrix}, C_1^b(k) = \begin{pmatrix} \varepsilon & \varepsilon & e \end{pmatrix},$$

$$x_2^a(k) = \begin{pmatrix} x_2(k) \\ x_3(k) \\ x_3(k-1) \end{pmatrix}, u_2^a(k) = i_2(k),$$

$$A_2^a(k) = \begin{pmatrix} \tau_5 & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon \end{pmatrix}, B_2^a(k) = \begin{pmatrix} \tau_4 \\ \varepsilon \\ \varepsilon \end{pmatrix}, C_2^a(k) = \begin{pmatrix} \varepsilon & \tau_6(k) & e \end{pmatrix},$$

$$x_2^b(k) = \begin{pmatrix} i_3(k) \\ i_3(k-1) \\ i_2(k) \end{pmatrix}, u_2^b(k) = i_3(k),$$

$$A_2^b(k) = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon \\ \varepsilon & e & \tau_3 \end{pmatrix}, B_2^b(k) = \begin{pmatrix} e \\ \varepsilon \\ \varepsilon \end{pmatrix}, C_2^a(k) = \begin{pmatrix} \varepsilon & \varepsilon & e \end{pmatrix}.$$

From these representations, the proposed algorithm can be used to compute evolutions of variables $I_j(k)$, $j = 1 \ldots N$.
With a view to implement a simulation software, these representations can be established for each kind of system composing the line in order to build a resource library.

5 CONCLUSION

A modeling methodology in $(\max, +)$ algebra has been presented for assembly lines. This approach has been implemented in a software package called MAISTeR [5] developed in collaboration with industry in order to simulate and evaluate performance of electronic cards production lines. The modularity of the approach has made it possible that the software user can simply build the model of the line by connecting machines models available in a library. For the simulation, the user must besides inform a schedule as well as an inventory position. Thanks to analysis modules, the resulting data are used by engineers or operators for several purposes: detection of bottle necks, prediction of requirements in human resources and component supply. They are also used to evaluate relevance of modifications on the line. We are currently aiming at implementing an additional control module, using available control results in $(\max, +)$, which would allow improving the production management.

References

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and Linearity.* Wiley, 1992.

[2] G. Cohen, P. Moller, J. P. Quadrat, and M. Viot. Algebraic tools for the performance evaluation of discrete event systems. *IEEE Proceedings: Special issue on Discrete Event Systems*, 77(1), Jan. 1989.

[3] B. Cottenceau. *Contribution à la commande de systèmes événements discrets : synthèse de correcteurs pour les graphes d'événements temporisés dans les dioïdes.* Ph.D Thesis, ISTIA - Université d'Angers, Oct. 1999.

[4] R.A. Cuninghame-Green. *Minimax Algebra.* Number 166 in Lecture notes in Economics and Mathematical Systems. Springer, 1979.

[5] J. L. Ferrier, S. Lahaye, L. Hardouin, and J. L. Boimond. MAISTeR : a user-friendly software package for performance analysis and decision, based on $(\max, +)$. In *Proceedings of the SMC'2000, IEEE conference on Systems Man and Cybernetics*, Nashville, USA, 2000.

[6] E. W. Kamen. Fundamentals of linear time-varying systems. In W. S. Levine, editor, *The Control Handbook.* IEEE Press and CRC Press, 1996.

[7] S. Lahaye. *Contribution à l'étude des systèmes linéaires non stationnaires dans l'algèbre des dioïdes*. Ph.D Thesis, ISTIA - Université d'Angers, Oct. 2000.

[8] S. Lahaye, J. L. Boimond, and L. Hardouin. Analysis of periodic discrete event systems in (max,+) algebra. In *Proceedings of WODES'2000*, Ghent, Belgium, 2000.

[9] Max-Plus. A linear system theory for systems subject to synchronization and saturation constraints. In *Proceedings of the first European Control Conference*, Grenoble, July 1991.

[10] E. Menguy, J. L. Boimond, L. Hardouin, and J. L. Ferrier. Just in time control of linear systems in dioid: cases of an update of reference input and uncontrollable input. *IEEE Trans. on Automatic Control*, 2000. to appear.

[11] T. Murata. Petri Nets : Properties, Analysis and Applications. *IEEE Proceedings: Special issue on Discrete Event Systems*, 77:541–580, Jan. 1989.

S. LAHAYE, J.-L. BOIMOND, L. HARDOUIN
Laboratoire d'Ingńierie des Systèmes automatisés, 62, Ave Notre Dame du Lac, 49000 Angers, France.
e-mail: {lahaye,hardouin,boimond}@istia.univ-angers.fr