Heap Models, Composition and Control

Jan Komenda, Sébastien Lahaye* & Jean-Louis Boimond*
Institute of Mathematics, Czech Academy of Sciences, Zizkova 22, Brno
Czech Republic
**LISA, Université d' Angers, 62, Av. Notre Dame du Lac, Angers*
France

1. Introduction

This chapter deals with modelling and control of discrete event systems. Discrete event systems (DES) are event driven man made systems whose evolution is guided by occurrences of asynchronous events as opposed to classical time driven discrete or continuous systems. DES are modelled by tools from computer science like automata, Petri nets and process algebras. There are two major streams in control theory of DES. The first stream, known as supervisory control theory, has been introduced by Wonham and Ramadge for logical automata, e.g. (Ramadge & Wonham,1989).

The second stream, more specialized, which uses the class of timed Petri nets, called timed event graphs is based on linear representation in the (max,+) algebra. It is of large interest to make a bridge between both approaches while generalizing both of them at the same time. Being inspired by papers on (max,+) automata (e.g. (Gaubert & Mairesse, 1999); (Gaubert, 1995), which generalize both logical automata and (max,+)-linear systems, it is interesting to develop a control method for (max,+) automata by considering supervisory control approach.

The time semantics of the parallel composition operation (called supervised product) we have proposed for control of (max,+) automata in (Komenda et al, 2007) are different from the standard time semantics for timed automata or timed Petri nets. One has to increase the number of clocks in order to define a synchronous product of (max,+) automata viewed as 1-clock timed automata. This goes in general beyond the class of (max,+) automata and makes powerful algebraic results for (max,+) automata difficult to use.

The results of (Gaubert & Mairesse, 1999) suggest however an alternative for the subclass of (max,+) automata corresponding to safe timed Petri nets, where synchronous product is standard composition of subnets through shared (synchronization) transitions. The intermediate formalism of heap models enables a letter driven (max,+)-linear representation of 1-safe timed Petri nets. Therefore it is interesting to work with heaps of pieces instead of (max,+)-automata and introduce a synchronous composition of heap models that yields essentially reduced nondeterministic (max,+)- automata representation of synchronous composition of corresponding (max,+)-automata. This way we obtain representations allowing for use of powerful dioid algebras techniques and the reduced dimension of concurrent systems at the same time: the dimension of synchronous product of two heap models is the sum of each models dimensions, while the dimension of supervised product of (max,+)-automata is the product of the individual dimensions, which causes an exponential blow up of the number of states in the number of components.

The extension of supervisory control to timed DES represented by timed automata is mostly based on abstraction methods (for instance region construction turning a timed automaton into a logical one). On the other hand abstraction methods are not suitable for (max,+) or heap automata, because their timed semantics (when weights of transitions are interpreted as their minimal durations) are based on the earliest possible behavior similarly as for timed Petri nets. The method we propose avoids any abstraction and works with timed DES (TDES) represented by heap models.

Similarly as for logical DES our approach to supervisory control is based on the parallel composition (synchronous product) of the system with the supervisor (another heap model). Our research is motivated by applying supervisory control on heap models, which are appropriate to model 1-safe timed Petri nets. This is realized by using the synchronous product: the controlled system is the synchronous product of the system with its controller (another heap model). The algebraization of the synchronous product that is translated into idempotent sum of suitable block matrices together with a linear representation of composed heap models using its decomposed morphism matrix is then applied to the control problem for heap models.

This research work is organized as follows. Algebraic preliminaries needed throughout this chapter are recalled in Section 2. In Section 3 are introduced heap models together with their synchronous product and their modelling by fixed-point equations in the dioid of formal power series. Section 4 is devoted to the study of properties of synchronous product of heap models that will be applied in Section 5 to supervisory control of heap models using residuation theory. Section 6 is devoted to an example illustrating the proposed approach. The conclusion is given and future extensions of our approach are discussed in Section 7.

2. Preliminaries on dioid algebras.

In this section we recall some fundamental algebraic notions needed in the next sections. An idempotent semigroup is a set M equipped with a commutative, associative operation $\oplus$ that has a unit element ε that satisfies the idempotency condition $a \oplus a = a$ for each $a \in M$. There is a naturally defined partial order $\preceq$ on any idempotent semigroup, namely, $a \preceq b$ if and only if $a \oplus b = b$.

An idempotent semigroup is called idempotent semiring or dioid if it is equipped with another associative operation $\otimes$ that has a unit element *e*, distributes over $\oplus$ on the left and on the right.

An idempotent semiring is said to be commutative if the multiplication is commutative. A dioid D is called to be complete if any nonempty subset has an infimum and a supremum with respect to the order generated by $\oplus$ and the distributivity axioms extend to infinite sums. In the context of complete dioids, the star operation has been introduced by $a^* = \overset{\infty}{\underset{i=0}{\oplus}} a^i$, where by convention $a^0 = e$. Let us recall the notation $a^+ = \overset{\infty}{\underset{i=1}{\oplus}} a^i$, *i.e.* $e \oplus a^+ = a^*$ and the well know properties $aa^* = a^+$ and $a^* \oplus a^+ = a^*$ Moreover, the infimum of each non empty subset A of D always exists and it is denoted by $a = \bigwedge_{x \in A} x$.

Elementary examples of dioids are number dioids such as the semiring $R_{\max} = (\mathrm{R} \cup \{-\infty\}, \max, +)$ endowed with maximum (denoted additively) and the usual addition (denoted multiplicatively). The zero element of $R_{\max}$ is $\varepsilon = -\infty$, the unity is e = 0. The complete version of $R_{\max}$ is denoted by $\overline{R}_{\max} = (R \cup \{\pm\infty\}, \max, +)$.

2.1 Fixed point equations and residuation

Two powerful tools have been developed for complete dioids: fixed-point theorems and the residuation theory. Residuation theory generalizes the concept of inversion for mappings that do not necessarily admit an inversion, in particular those among ordered sets. If $f : C \to D$ is a mapping between two dioids, in most cases there does not exist a solution to the equation $f(x) = b$. Instead of solutions to this equation, the greatest solution to the inequality $f(x) \leq b$ or the least solution to the inequality $f(x) \geq b$ are considered. In the case these exist for all $b \in D$, the mapping f is called residuated, and dually residuated, respectively. The notation proposed in (Baccelli et al., 1992) is used in this paper.

Concretely, the residual mapping of a residuated mapping f will be denoted by $f^{\#}$. It is defined by the formula: $f^{\#}(y) = \max\{x : \mathrm{f(x)} \preceq \mathrm{y}\}$. The residual of the right multiplication by an element $a \in D$ in a commutative dioid will be denoted through x / a, i.e. $x / a = \max_{y \in D}(y \otimes a \leq x)$. Similarly, the residual of the left multiplication is denoted by $a \setminus x = \max_{y \in D}(a \otimes y \leq x)$.

Finally, residuation of matrix multiplication will be needed. In this paper only residuated mappings of matrix multiplication are used. The residuated mapping of the left matrix multiplication, i.e. the greatest solution to the inequality $AX \leq B$ is denoted by $A \setminus B$. Similarly, the residuated mapping of the right matrix multiplication, i.e. the greatest solution to the inequality $XA \leq B$ is denoted by B / A. Recall from (Gaubert 1992) that for matrices $A \in D^{m \times n}$, $B \in D^{m \times p}$, and $C \in D^{n \times p}$ over a complete dioid D $A \setminus B \in D^{\mathrm{n \times p}}$ is given by $(A \setminus \mathrm{B})_{ij} = \overset{m}{\underset{l=1}{\wedge}} A_{li} \setminus \mathrm{B_{lj}}$ and $B / C \in D^{\mathrm{m \times n}}$ is given by $(B / \mathrm{C})_{ij} = \overset{p}{\underset{k=1}{\wedge}} B_{ik} / \mathrm{C_{jk}}$.

Now fixed point equations in complete dioids are recalled (see Baccelli et al. 1992).

Theorem 1. Let D be a complete dioid, and $x = a \otimes x \oplus b$ be a linear equation of the fixed-point type. The least solution to this equation exists and is given by $x = a^* \otimes b$.

The following Lemma will be needed.
Lemma 1. Let D be a complete dioid, $a, b \in D$. Then

$$(a \oplus b)^* = (a^* b)^* a^* = (b^* a)^* b^* = a^* (b a^*)^* = b^* (a b^*)^* \quad . \tag{1}$$

2.2 Formal power series and their properties

Now we recall formal power series, which include both formal languages and dater functions from R to R_{max} and their properties. Formal power series in the noncommutative variables from and coefficients from R_{max} form a dioid called dioid of formal power series. The standard notation A^* is used for the free monoid of finite sequences (words) from A. The empty word is denoted by *1*.

Formal power series form a dioid denoted $R_{max}(A)$, where addition and (Cauchy or convolution) multiplication are defined as follows.
For any $s, s' \in R_{max}(A)$

$$\begin{aligned}(s \oplus s')(w) &= s(w) \oplus s'(w)\\ (s \otimes s')(w) &= \bigoplus_{uv = w} s(u) \oplus s'(v)\end{aligned} \tag{2}$$

This dioid is isomorphic to the dioid of generalized dater functions from A^* to R_{max} similarly as the dioid $Z_{max}(\gamma)$ used to study Timed Event Graphs (TEG), is isomorphic to the dioid of daters Z to Z_{max}. The isomorphism associates to any
$y : A^* \to R_{max}$ the formal power series $\bigoplus_{w \in A^*} y(w) w \in R_{max}(A)$.
The zero and identity series are respectively denoted ε and e. It will always be clear from the context whether these elements are meant for a number dioid or a diod of formal power series.
Let us recall $\forall w \in A^*,\ \varepsilon(w) = -\infty$ and $e(w) = \begin{cases} 0 & w = 1 \\ -\infty & w \neq 1 \end{cases}$.
We consider in the sequel the complete version of $R_{max}(A)$ with coefficients in $\overline{R}_{max}$.

An order relation on $R_{max}(A)$ will be needed for introduction and study of control problems in Section 5. Let us recall the natural order relation on formal power series from $R_{max}(A)$. For $s, s' \in R_{max}(A)$ we put $s \le s'$ iff $\forall w \in A^* : s(w) \le s'(w)$, where in the latter inequality usual order on R_{max} that coincides with the natural order is used.

3. Heap models and their synchronous products

In this section heap models together with their basic poperties are first recalled. Then synchronous product of heap models is proposed as a mechanism to build large heap models out of smaller ones.

3.1 Basic properties of heap models

Let us first recall the definition of time extension of heap models, also called task-resource systems (Gaubert & Mairesse, 1999), which model an important class of TDES exhibiting both synchronization and resource sharing phenomena.

Definition 1. (Heap model)

A heap model is the structure $\Re = (A, P, r, l, u)$, where

- A is a finite set of pieces (also called tasks)
- P is a finite set of slots (also called resources)
- $r : \mathrm{A} \to \mathrm{Pwr(P)}$ determines the subset of resources required by a task. It is always assumed that $\forall \mathrm{a} \in \mathrm{A} : \mathrm{r(a)} \neq \varnothing$.
- $l : \mathrm{A} \times \mathrm{P} \to \mathrm{R}_{\max}$ is a function such that $l(a,p)$ determines the height of the lower contour of piece *a* at the slot p.
- $u : \mathrm{A} \times \mathrm{P} \to \mathrm{R}_{\max}$ is function such that $u(a,p)$ determines the height of the upper contour of piece *a* at the slot p.

By convention, $u \leq l$, $l(a,p) = u(a,p) = -\infty$ if $p \notin r(a)$, and $\min_{p \in r(a)} l(a,p) = 0$.

Any sequence of pieces (tasks) is called a heap, i.e. heaps are just words of the free monoid A^*. There is a nice geometrical interpretation of heaps.

The dynamics of heap models is described by row vectors of generalized dater function $x(w)_p ; p \in P$ corresponding to the height of the heap $w = a_1 \ldots a_n$ on individual slots $p \in P$. It has been shown in (Gaubert & Mairesse, 1999) that the upper contour of a heap w, denoted by $x(w)$, and the overall height of the heap w, denoted $y(w)$, are given by the following letter driven (max,+)-linear equations in terms of generalized dater functions x and y.

$$\begin{aligned} x(1) &= (0 \ldots 0) \\ x(wa) &= x(w)\mu(a) \\ y(w) &= x(w)(0 \ldots 0)^T \end{aligned}$$

with the so called morphism matrix associated to the heap model and given by:

$$[\mu(a)]_{pq} = \begin{cases} 0 & \text{if} \quad p = q \notin r(a) \\ u(a,q) - l(a,s) & \text{if} \quad p \in r(a) \, and \;\; q \in r(a) \\ -\infty & \text{otherwise} \end{cases} \tag{3}$$

It is called the morphism matrix associated to the heap model $\Re = (A, P, r, l, u)$.

In fact; it is shown in (Gaubert & Mairesse, 1999) that heap models are special (max,+)-automata with input and output functions as row, resp. column vectors of zeros, i.e. (max,+) identity elements, and the morphism matrix defined above. The (max,+)-automaton $A(\Re) = (\alpha, \mu, \beta)$ given by the triple input function $\alpha = (0 \dots 0)$, output function $\beta = (0 \dots 0)^T$, and the morphism matrix μ recalled above is then called heap automaton. Therefore one may view heap models as special (max,+)-automata called heap automata.
On one hand the upper contour (height) of heap models is recognized by (max,+) automata and are considered as a subclass of (max,+) automata , but on the other hand it is known from (Gaubert & Mairesse, 1999) that heap models have a strong expressive power in terms of Timed Petri Nets (TPN), a very important tool in the study of Timed DES. Let us recall at this point few basic facts about TPNs.

Definition 2. (Timed Petri Net)
A Timed Petri Net (TPN) is a valued bipartite graph represented by a 5-tuple $N = (P, T, W, M, t_p)$. The finite sets P and T are called the set of places and the set of transitions, respectively. $W \in N^{|P| \times |T|}$ is the Boolean incidence matrix, $W_{ij} = W_{ij}^{+} - W_{ij}^{-}$, where W_{ij}^{+} equals 1 iff there exists an edge going from the transition T_j to the place P_i and similarly W_{ij}^{+} equals 1 iff there exists an edge going from P_i to T_j. Otherwise the corresponding values of W_{ij}^{+} and W_{ij}^{-} are 0. The vector $M \in N^P$ denotes the distribution of the initial marking in the places. The vector $t_P \in N^P$ is formed by the sojourn times associated to the places of the net.

A place may contain tokens (called marking), which move from place to place according to the following (earliest functioning) firing rule. A transition T_j fires as soon as all the places P_i upstream T_j contain at least one available token. A token entering a place P_i is considered available after having sojourned t_{P_i} units of time. A Timed Event Graph (TEG) is a TPN such that each place has exactly one upstream and one downstream transition.

Although heap models similarly as Petri nets already support the concurrency phenomenon, it is useful for control purposes to introduce explicit concurrency by defining synchronous product of heap models. DES are typically composed of large number of components and the compositionality, i.e. building of large system out of smaller ones and inversely analysis or control synthesis based on decomposition into elementary components are very efficient techniques.
So we will consider two levels of concurrency: "local" (or implicit) concurrency that is inside each heap model given by "local" tasks that do not share any resource and the "global" (or explicit) concurrency between tasks of two component heap models that are not common to both heaps, but belong only to one component. This is is given by the distribution of tasks in the definition of synchronous composition below. We then speak about private and shared tasks of two heap models.

The situation is similar to logical automata, where there can be two levels of concurrency (an explicit one using synchronous composition and an implicit one inside individual automata that may have "hidden concurrency").

We assume in the definition of synchronous product below that there are no shared resources between two heap models. Otherwise stated: resources are shared only by tasks within individual heap models. This requirement is best understood if one considers safe timed Petri nets (which can be viewed (Gaubert & Mairesse, 1999) as particular heap models), where synchronous compositions of subnets is realized by synchronizing shared transitions (in heap models tasks), while the set of places (in heap models resources) of the individual subnets are disjoint.

Definition 3. (Synchronous product of heap models)
Let $\Re_i = (A_i, P_i, r_i, l_i, u_i), i = 1,2$ be two heap models with $P_1 \cap P_2 = \varnothing$. Their *synchronous product* is the heap model $\Re_1 \,||\, \Re_2 = (A_1 \cup A_2, P_1 \cup P_2, r, l, u)$, where

$$r(a) = \begin{cases} r_1(a) \cup r_2(a) & \text{if } a \in A_1 \cap A_2 \\ r_1(a) & \text{if } a \in A_1 \setminus A_2 \\ r_2(a) & \text{if } a \in A_2 \setminus A_1 \end{cases},$$

$$l(a,p) = \begin{cases} l_1(a,p) & \text{if } p \in P_1 \\ l_2(a,p) & \text{if } p \in P_2 \end{cases}, \text{ and } u(a,p) = \begin{cases} u_1(a,p) & \text{if } p \in P_1 \\ u_2(a,p) & \text{if } p \in P_2 \end{cases}.$$

Since the slots (resources) of component heaps are disjoint, l and u are well defined: even though in $A_1 \cap A_2 = \varnothing$ for any $p \in P$ there is only one $i \in \{1,2\}$, namely i such that $p \in P_i$, with $l_i(a,p)$ and $u_i(a,p)$ being defined.

Similarly as in the supervisory control of (logical) automata the purpose of synchronous product is twofold. Firstly, explicitly concurrent heap models (*cf.* concurrent or modular automata) are heap models built by the synchronous product of "local" heap models, whence the interest in studying the properties of synchronous composition of heap models. Secondly, synchronous product is used to describe the action of the supervisor, *i.e.* interaction of the supervisor with the system *cf.* (Kumar & Heymann, 2000).

Let us remark that if the above definition is used for control purposes, it is symmetric with respect to both the plant heap (say $\Re_1$) and the controller (say $\Re_2$). We then implicitly assume in the above definition that the supervisor is complete, *i.e.* that it never tries to disable an uncontrollable task. This is always true in the special case, where all tasks are controllable.

Now (max,+)-linear representation of heap models will be used for the study of the morphism matrix of the synchronous product of two heap models. An approach for just in time control of flexible manufacturing systems based on Petri net and heap models, that builds upon the approach of (Menguy, 1997), has been developped in (Al Saba et al., 2006).

Our aim is to develop the control theory directly for heap models using synchronous composition of a heap model with its controller (another heap model).

Let us consider the following flexible manufacturing system modelled by Petri net displayed below in Figure 1.

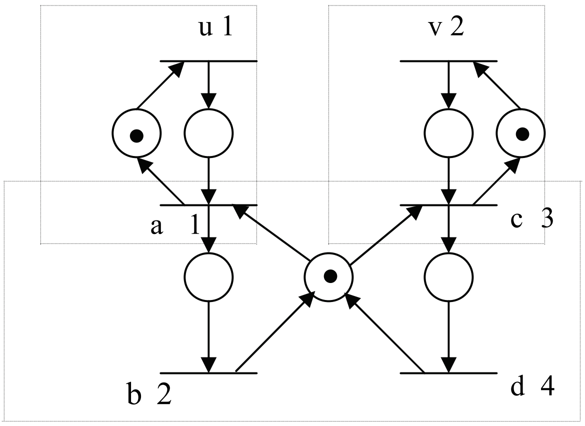

Fig. 1. Example of a Timed Petri Net composed of three modules

Note that this 1-safe Petri net is T-timed (timing is associated to transitions) and it can be decomposed into three parts, which are synchronized using shared (synchronization) transitions a and c. Equivalently, each component of this Petri net can be viewed as a separate heap model. The "global" heap model corresponding to the whole timed Petri net is the synchronous product of "local" heap models.

Similarly as a TEG admits a linear representation in the dioid of formal power series $Z_{max}(\gamma)$, see (Bacceli et al., 1992), a heap model admits linear representation in the dioid of formal power series with noncommutative variables from A that we denote by $R_{max}(A)$. As an example let us consider the following heap automaton $\Re$ corresponding to the TPN depicted on figure 1.

The set of tasks is $A=\{a,b,c,d\}$. The set of resources is $P=\{p_1,p_2,p_3\}$. Let $r(a)=r(b)=\{p_1,p_2\}$ and $r(c)=r(d)=\{p_1,p_3\}$. The lower and upper contours of tasks are given by

$$\begin{aligned}
&l(a,.)=\begin{pmatrix}0 & 0 & -\infty\end{pmatrix}; l(b,.)=\begin{pmatrix}0 & 0 & -\infty\end{pmatrix}\\
&l(c,.)=\begin{pmatrix}0 & -\infty & 0\end{pmatrix}; l(b,.)=\begin{pmatrix}0 & -\infty & 0\end{pmatrix}\\
&u(a,.)=\begin{pmatrix}0 & 1 & -\infty\end{pmatrix}; l(b,.)=\begin{pmatrix}2 & 0 & -\infty\end{pmatrix}\\
&u(c,.)=\begin{pmatrix}0 & -\infty & 3\end{pmatrix}; u(b,.)=\begin{pmatrix}4 & -\infty & 0\end{pmatrix}
\end{aligned}$$

It has been shown in (Gaubert & Mairesse, 1999) that any heap model is a special (max,+)-automaton with the morphism matrix defined in equation (3). The graphical interpretation of the morphism matrix is given in terms of transition weights:
$[\mu(a)]_{ij} = k$ means that there is a transition labelled by $a \in A$ from state i to state j with weight k provided $k \neq -\infty$, while in the case $k = -\infty$ there is no transition from state i to state j. Note that the morphism matrix μ of a heap model can be also considered as element of $R_{max}(A)^{|R|\times|R|}$, and by extending the definition of μ from letters $a \in A$ to sequences (words) $w \in A^*$ using the morphism property $\mu(a_1 \ldots a_n) = \mu(a_1)\ldots\mu(a_n)$ and hence we can even write $\mu = \bigoplus_{w\in A^*} \mu(w)w$.
However, μ has an important property of being finitely generated, because it is completely determined by its values on A: $\mu(a), a \in A$. For this reason we have in fact $\mu^* = \left(\bigoplus_{a\in A} \mu(a)a\right)^*$. Since we are interested in behaviors of heap models that are given in terms of μ^* (more precisely by $\alpha\mu^*\beta$) we abuse the notation and write simply $\mu = \bigoplus_{a\in A} \mu(a)a$.

The corresponding heap automaton is in Figure 2 below.

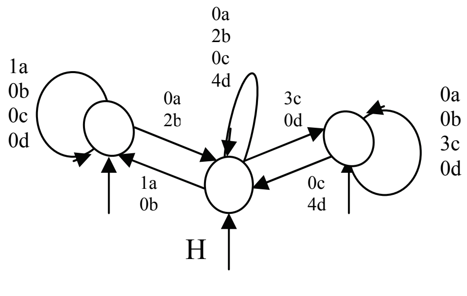

Fig. 2. Heap automaton corresponding to the Timed Petri Net

The state vector is associated to resources of $\mathfrak{R}$. We denote the component variables (formal power series) $x_2, x_1, x_3 \in R_{max}(A)$ from left to right. We obtain the following equations in dioid $R_{max}(A)$:

$$\begin{aligned}
x_1 &= x_1(0a \oplus 2b \oplus 0c \oplus 4d) \oplus x_2(0a \oplus 2b) \oplus x_3(0c \oplus 4d) \oplus e \\
x_2 &= x_1(1a \oplus 0b) \oplus x_2(1a \oplus 0b \oplus 0c \oplus 0d) \oplus e \\
x_3 &= x_1(3c \oplus 0d) \oplus x_3(0a \oplus 0b \oplus 3c \oplus 0d) \oplus e \\
y &= x_1 \oplus x_2 \oplus x_3
\end{aligned} \tag{4}$$

The corresponding matrix form of system of equations (4) is

$$\begin{array}{l} x = x\mu \oplus \alpha \\ \mathrm{y} = x\beta \end{array} \quad \text{with} \quad x = (x_1\, x_2\, x_3), \quad \alpha = (0\,0\,0), , \quad \beta = (0\,0\,0)^T, \text{ and finally}$$

$$\mu = \begin{pmatrix} 0a \oplus 2b \oplus 0c \oplus 4d & 1a \oplus 0b & 3c \oplus 0d \\ 0a \oplus 2b & 1a \oplus 0b \oplus 0c \oplus 0d & \varepsilon \\ 0c \oplus 4d & \varepsilon & 0a \oplus 0b \oplus 3c \oplus 4d \end{pmatrix}. \tag{5}$$

It is now easy to see that in general we always have the following linear description of (max,+) automata in the dioid $R_{max}(A)$ of formal power series:

$$\begin{array}{l} x = x\mu \oplus \alpha \\ \mathrm{y} = x\beta \end{array} \quad \text{with } \mu = \underset{a \in A}{\oplus} \mu(a)a \text{ the so called morphism matrix.}$$

Let us recall from Theorem 1 that the least solution to this equation is $\mathrm{y} = \alpha\mu^* \beta$.

4. Linear representation of synchronous product

Since we introduce supervisory control of heap models using the synchronous product of the plant heap model with the controller heap model, it is important to study properties of the synchronous product. In this section the behavior of a synchronous product of two heap models will be represented in terms of morphism matrix of the synchronous product. According to the definition of synchronous product, the dimension (here number of resources) of the synchronous product of subsystems is the sum of dimensions of each subsystem. It should be intuitively clear that morphism matrices of synchronous products are block matrices, where the blocks are formed according to dimensions (number of resources) of the component heap models.

In order to simplify the approach we require that $\mathrm{l}(\mathrm{a},\mathrm{p}) = 0$ whenever $p \in r(a)$. This is a reasonable assumption that simply means that pieces are on the ground in all slots. Thus, the upper contour gives information about the duration of tasks for different resources used. We recall at this point that $\mathrm{l}(\mathrm{a},\mathrm{p}) = -\infty$ whenever $p \notin r(a)$.

Let $\Re_1$ and $\Re_2$ be two heap models with morphism matrices denoted by μ_1 and μ_2, respectively. Their dimensions, i.e. the number of resources of $\Re_1$ and $\Re_2$ are m_1 and m_2 , respectively. The Booleans morphism matrices of underlying Boolean automata are denoted by B_1 and B_2 , respectively. Let us recall from (Komenda et al, 2007) that

$$[B_1(a)]_{ij} = \begin{cases} e & \text{if } \mu_1(a)_{ij} \neq \varepsilon \\ \varepsilon & \text{if } \mu_1(a)_{ij} = \varepsilon \end{cases}$$

Let $\varepsilon_{ij}, i, j = 1{,}2$ be the rectangular matrices of (max,+) zeros of dimensions $m_i \times m_j$ and $E_i, \mathrm{i} = 1{,}2$ the square (max,+)-identity matrices of dimensions $m_i \times m_i$.

It will be shown that the morphism matrix of the parallel composition admits an interesting decomposition into blocks.
Indeed, the following block form of $\mu_{\mathfrak{R}}(a)$ depending on $a \in A = A_1 \cup A_2$ is now claimed.
Theorem 2. The morphism matrix of $\mathfrak{R} = \mathfrak{R}_1 \,||\, \mathfrak{R}_2$ admits the following decomposition:

- if $a \in A_1 \cap A_2$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} \mu_1(a) & \overline{\mu}_2(a) \\ \overline{\mu}_1(a) & \mu_2(a) \end{pmatrix}$, where

$$\overline{\mu}_1(a)_{ij} = \begin{cases} \mu_1(a)_{kj},\ k \in r_1(a) \text{arbitrary} & \text{if } \mathrm{i} \in \mathrm{r}_2(a) \\ -\infty & \text{if } \mathrm{i} \notin \mathrm{r}_2(a) \end{cases} \tag{6}$$

and similarly, $\overline{\mu}_2(a)_{ij} = \begin{cases} \mu_2(a)_{lj},\ l \in r_2(a) \text{arbitrary} & \text{if } \mathrm{i} \in \mathrm{r}_1(a) \\ -\infty & \text{if } \mathrm{i} \notin \mathrm{r}_1(a) \end{cases}$

- if $a \in A_1 \setminus A_2$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} \mu_1(a) & \varepsilon_{12} \\ \varepsilon_{21} & E_2 \end{pmatrix}$ (7)

- if $a \in A_2 \setminus A_1$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} E_1 & \varepsilon_{12}(a) \\ \varepsilon_{21}(a) & \mu_2(a) \end{pmatrix}$ (8)

Proof. This result can be obtained from the form of the morphism matrix in equation (3). In accordance with the definition of morphism matrix of heap models for $a \in A = A_1 \cup A_2$ three cases are to be distinguished. Let us start with the most complicated case, where $a \in A_1 \cap A_2$ is a shared task. It should be clear that for resources p, q from $P = P_1 \cup P_2$ there are four possibilities depending on whether p and q belong to P_1 or P_2, whence the block form of $\mu_{\mathfrak{R}}(a)$. At this point we recall our assumption that $P_1 \cap P_2 = \varnothing$. It is then straightforward to see that in the diagonal blocks the individual morphism matrices appear. Also, the remaining non diagonal terms according to the definition of the morphism matrix of heap automata are equal to:

$$\overline{\mu}_1(a)_{ij} = \begin{cases} u(a,j) - l(a,i) = u_1(a,j) - l_2(a,i) = u_1(a,j) & \text{if } i \in r_2(a) \text{and} j \in r_1(a) \\ -\infty & \text{if } i \notin r_2(a) \text{or} j \notin r_1(a) \end{cases} \tag{9}$$

Note that any such element above in the lower left block must satisfy $i \in P_2$ and $j \in P_1$, hence $u(a,j) = u_1(a,j)$, $l(a,i) = l_2(a,i)$, and $l_2(a,i) = 0$ for $i \in r_2(a)$ according to our assumption and similarly for elements in the right upper block of $\mu_{\mathfrak{R}}(a)$ we get:

$$\overline{\mu}_2(a)_{ij} = \begin{cases} u(a,j) - l(a,i) = u_2(a,j) - l_1(a,i) = u_2(a,j) & \text{if } i \in r_1(a) \text{ and } j \in r_2(a) \\ -\infty & \text{if } i \notin r_1(a) \text{ or } j \notin r_2(a) \end{cases} \quad (10)$$

Since $\overline{\mu}_1(a)_{ij}$ equals either $u_1(a,j)$ (for $i \in r_2(a)$ and $j \in r_1(a)$) or to (max,+) zero otherwise, we can see that the rows corresponding $i \in r_2(a)$ of $\overline{\mu}_1(a)$, i.e. $\overline{\mu}_1(a)_{(i,.)}$ are the same as the rows $\overline{\mu}_1(a)_{(k,.)}$ for $k \in r_2(a)$, which are all the same.

Thus, the corresponding entry does not depend on *j* anymore. Similar arguments can show that $\overline{\mu}_2(a)$ is of the form above. The morphism matrix $\mu_{\Re}(a)$ of the composed heap has then the claimed form for $a \in A_1 \cap A_2$. In much easier situation when $a \in A_1 \setminus A_2$ it is sufficient to notice that no resource from P_2 is used by a. It is easily seen that $\mu_{\Re}(a)$ has again the claimed form. Finally, the case $a \in A_2 \setminus A_1$ is symmetric to the previous one.

Theorem on decomposition can be generalized to the case of n component heaps, where morphism matrix of synchronous product are matrices with $n \times n$ blocks. This is useful for decentralized control, but in this paper we only need synchronous product of the system with its controller, i.e. the case $n = 2$.

The algebraization of synchronous product presented in this section will be useful for control purposes in the next section.

5. Application to supervisory control

In this section supervisory control of heap models is studied. The aim is to satisfy a behavioral specification given by a formal power series. The closed-loop system is represented by parallel composition (synchronous product) of the plant with a supervisor to be found, which is itself represented by a heap model.

In general a supervisor acts on both timing and logical properties of the plant's behavior under supervision. In this section we focus mainly on timing aspects, when the supervisor does not disable, but only delay the execution of tasks corresponding to transitions in underlying timed Petri nets. This is similar to control of TEG in the maxplus algebra, where input transitions are added in order to delay the timed behavior of a TEG (Cottenceau et al, 2000).

Note that adding input transitions to a TEG corresponds to synchronization of the TEG with a TEG of the controller that has predefined structure given by the graph topology of input transitions. This way control of TEG can be seen in this sense as a special case of supervisory control, where the supervisor has a predefined fixed structure (logical behavior) and only output values (timing) of transitions are to be determined such that a specified behavior is met. Note however that such a methodology enables only to consider periodic inputs without a transient regime, which would require to either consider a controller, where weights are time depending or equivalently to use a unfolding (max,+) automaton representation of the controller, which would have much more states.

Now we apply Theorem 2 to the control of heap models. The synchronous product of heap models $G=(A_g,P_g,r_g,l_g,u_g)$ and $C=(A_c,P_c,r_c,l_c,u_c)$ of dimensions m and n (respectively) corresponds to the controlled (closed-loop) system. The event alphabets of G and C are denoted by A_g and A_c, respectively. The event alphabet of the controlled system is denoted by A. According to definition of synchronous product, we then have $A=A_g\cup A_c$. Let us denote the morphism matrices of G and C by μ_g and μ_c, respectively. Now let us return to the description of behaviors of heap models in the dioid of formal power series $R_{max}(A)$. The vector of formal power series from $R_{max}(A)$ associated to generalized dater functions $x_{G||C}:\mathrm{A}^*\to R_{max}^{m+n}$ satisfies the following equations:

$$\begin{aligned} x_{G||C} &= x_{G||C}\otimes\mu_{G||C}\oplus\alpha \\ y_{G||C} &= x_{G||C}\otimes\beta \end{aligned} \tag{11}$$

where $\mu_{G||C}$ the morphism matrix of $G||C$, α, and β are row, resp. column, vectors of zeros of dimension $m+n$, *i.e.* $\alpha=(e...e)$, and $\beta=(e...e)^T$. First of all, according to Theorem 1 the greatest solutions to equations above are

$$\begin{aligned} x_{G||C} &= \alpha\mu_{G||C}^{*} \\ y_{G||C} &= \alpha\mu_{G||C}^{*}\beta \end{aligned} \tag{12}$$

whence an interest in studying properties of $\mu_{G||C}^{*}$.

Given a specification behavior (e.g. language or formal power series), the goal in supervisory control of DES is to find a supervisor that achieves this specification as the behavior of the controlled system. In a first approach we assume, similarly as in control of TEG, that the structure of the controller is given, which means here that the controller heap model only delays executions of different tasks in the plant. This is done by the choice of upper contour functions (*i.e.* duration of controller's tasks) from $\mu_c(u), u\in A_c$. The delaying effect of the controller is naturally realized via its tasks (transitions of the corresponding heap automaton) shared with the plant heap.

The morphism matrix of the composed system is given by

$$\mu_{G||C}=\bigoplus_{a\in A}\mu_{G||C}(a)a=\bigoplus_{u\in A_c\setminus A_g}\mu_{G||C}(u)u\oplus\bigoplus_{a\in A_c\cap A_g}\mu_{G||C}(a)a\oplus\bigoplus_{u\in A_g\setminus A_c}\mu_{G||C}(a)a\,. \tag{13}$$

In order to simplify the approach our attention is from now on limited to the case $\mathrm{A_c} = A_g$. This is a standard assumption in the supervisory control with complete observations. Since state vector in equation (12) is associated to resources, it can be written as $x_{G||C} = (x \quad u)$, where the first component corresponds to the (uncontrolled) plant (heap G) and the second to the controller heap C.
Owing to Theorem 2 we have:

$$(x \quad u) = (x \quad u)\left[\bigoplus_{a\in A}\begin{pmatrix} H(a)a & \overline{F}(a)a \\ \overline{H}(a)a & F(a)a \end{pmatrix}\right] \oplus (\alpha_1 \quad \alpha_2)\ , \tag{14}$$

where α_1 and α_2 are vectors of zeros of corresponding dimensions, for any $a \in A$ $\mu_g(a)$ is for convenience denoted by $H(a)$, i.e. $\overline{H}(a)$ denotes $\overline{\mu}_g(a)$. Similarly, $F(a)$ is the notation for $\mu_c(a)$, i.e. $\overline{F}(a)$ denotes $\overline{\mu}_c(a)$
This way we obtain

$$(x \quad u) = (x \quad u)\begin{pmatrix} H & \overline{F} \\ \overline{H} & F \end{pmatrix} \oplus (\alpha_1 \quad \alpha_2),$$

where $H = \bigoplus_{a\in A} H(a)a$ and similarly $F = \bigoplus_{a\in A} F(a)a$, $\overline{H} = \bigoplus_{a\in A} \overline{H}(a)a$, and $\overline{F} = \bigoplus_{a\in A} \overline{F}(a)a$.
Hence,

$$x = xH \oplus u\overline{H} \oplus \alpha_1$$
$$u = x\overline{F} \oplus uF \oplus \alpha_2$$

According to Theorem 1 the least solution of the second equation is $u = (x\overline{F} \oplus \alpha_2)F^*$. The above equation gives us the expression of the closed-loop system transfer, where the matrix F plays the role of a feedback. It is worth to notice that there is a significant difference from standard feedback control: unlike classical control theory the control variables have their inner dynamics. This is caused by adopting a supervisory control approach, where a controller is itself a dynamical system of the same kind as the uncontrolled system: a heap automaton. Therefore, our $\overline{F}$, which plays the role of feedback mapping, is determined by inner dynamics of the controller given by its morphism matrix F. The substitution into the first equation then yields

$$x = xH \oplus \left[(x\overline{F} \oplus \alpha_2)F^*\right]\overline{H} \oplus \alpha_1$$

Another application of Theorem 1 leads to the least solution given by

$$x = (\alpha_2 F^* \overline{H} \oplus \alpha_1)(H \oplus \overline{F}F^*\overline{H})^*.$$

From a different viewpoint, there is a strong analogy with the feedback approach for control of TEG, see (Cottenceau et al, 2000), which should not be surprising, because supervisory control (realized here by synchronous product) is based on a feedback control architecture. In supervisory control the control specification (as counterpart of reference output from control of TEG using dioid algebras) are given in terms of behaviors of (max,+) automata (*i.e.* formal power series). In fact, for a reference output y_{ref} we are interested in the greatest feedback F such that the output of the closed-loop system y is less or equal to the specification y_{ref} : $y \le y_{ref}$ .We obtain in our situation:

$$\left(\alpha_2 F^* \overline{H} \oplus \alpha_1\right)\left(H \oplus \overline{F} F^* \overline{H}\right)^* \beta \le y_{ref} . \tag{15}$$

And $\left(H \oplus \overline{F} F^* \overline{H}\right)^* \le \left(\alpha_2 F^* \overline{H} \oplus \alpha_1\right) \backslash y_{ref} / \beta$, where $\left(\alpha_2 F^* \overline{H} \oplus \alpha_1\right) \backslash y_{ref} / \beta$ is a matrix playing the role of reference model in (Cottenceau et al, 2000). An application of Lemma 1 leads to:

$$\left(H \oplus \overline{F} F^* \overline{H}\right)^* = H^* \left(\overline{F} F^* \overline{H} H^*\right)^* \le \left(\alpha_2 F^* \overline{H} \oplus \alpha_1\right) \backslash y_{ref} / \beta . \tag{16}$$

Note that all operations involved in the description above, i.e. $\oplus, \otimes$, and the Kleene star, are lower semicontinuous and residuated when the image is suitably constrained. Hence, using residuation theory (see Section 2) it should be possible to obtain the greatest series F corresponding to the "controller part" of the morphism matrix.

Moreover, as follows from Theorem 2 that $\overline{F}$ can simply be expressed using F. Hence, there is a hope that at least in some special cases residuation theory (see Section 2) can be applied to obtain the greatest series F corresponding to the "controller part" of the morphism matrix $\mu_{G||C}$ such $y_{G||C}$ satisfies a given specification (e.g. it is less than or equal to a given reference output series y_{ref}).

We aim at obtaining the greatest $\overline{F} F^*$ such that inequality (16) is satisfied. This is far from being an easy problem. The situation is much simpler in case $\overline{F} = F$ and $\overline{H} = H$. This is satisfied if we assume that the controller heap model has the same number of resources as the uncontrolled heap model and the logical structure of the controller (given by $r_c : A \to Pwr(P_c)$ mimics the logical structure of the plant (given by $r_g : A \to Pwr(P_g)$). Formally stated it is required that there exists an isomorphism between P_c and P_g such that r_c and r_g are equal up to this isomorphism.

In terms of Petri nets this can be interpreted as having a controller net with the same net topology (*i.e.* logical structure as the uncontrolled net). This means that in the closed-loop system there are always parallel places of the controller corresponding to places of the uncontrolled net.

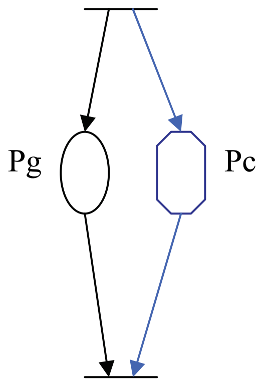

Fig. 3. The interpretation of controller as a Timed Petri Net.

The role of the controller is only to act on the system through holding times of the controller's places that correct the holding times of the places in the original net. Because of the fixed parallel structure of the controller it is clear that the controller can in this case only delay the firing of the transitions, which are all shared by the system and the controller.

It is easy to check that Theorem 2 in such a case gives $\overline{F} = F$ as well as $\overline{H} = H$ and $\alpha_1 = \alpha_2 = \alpha$, row vector of zeros of dimension *m*=*n.*
Hence, inequality (15) leads to
$\alpha\left(F^* H \oplus E\right)\left(H \oplus FF^* H\right)^* \beta \le y_{ref}$, where E is the identity matrix.
An easy calculation yields $\left(H \oplus FF^* H\right)^* = \left(\left(E \oplus F^+\right)H\right)^* = \left(F^* H\right)^*$ and then $\left(F^* H \oplus E\right)\left(F^* H\right)^* = \left(F^* H\right)^+ \oplus \left(F^* H\right)^* = \left(F^* H\right)^*$. Thus, the control problem becomes :

$$y = \alpha\left(F^* H\right)^* \beta \le y_{ref} . \tag{17}$$

i.e. the problem is to find the greatest F such that $\left(F^* H\right)^* \le \alpha \backslash y_{ref} / \beta$.
Since the Kleene star is not a residuated mapping in general, such a problem has only a solution if $\alpha \backslash y_{ref} / \beta$, playing the role of reference model G_{ref} from see (Cottenceau et al, 2000), is of a special form to be found.
We notice that in our particular situation $E \le H$, which follows from the form of morphism matrix of heap models (see Section 3).
The following Lemma is now useful.

Lemma 2. If $E \le H$ then for any $B \in R_{max}(A)^{n \times n}$ any solution of $\left(X^* H\right)^* \le B$ is a solution of $\left(X^* H\right)^* X^* \le B$ and vice versa.

Proof. If X a solution of $\left(X^*H\right)^* \le B$, then $\left(X^*H\right)^* = E \oplus \left(X^*H\right)^+ \le B$, hence also $\left(X^*H\right)^+ \le B$. Therefore,

$$\left(X^*H\right)^* X^* = \left(X^*H\right)^* X^*E \le \left(X^*H\right)^* X^*H = \left(X^*H\right)^+ \le B,$$

where the first inequality follows from isotony of multiplication and the assumption $E \le H$. Conversely, if X is a solution of $\left(X^*H\right)^* X^* \le B$, then

$$\left(X^*H\right)^* = \left(X^*H\right)^* E \le \left(X^*H\right)^* X^* \le B$$

as follows from isotony of multiplication and the assumption that $E \le X^*$, a general and very well known property of the Kleene star.

Using Lemma 2 (with F playing the role of X) our problem is equivalently formulated as follows : find the greatest solution in F of $\left(F^*H\right)^* F^* \le \alpha \setminus y_{ref} / \beta$. It follows from Lemma 1 that $\left(F^*H\right)^* F^* = (H \oplus F)^* = H^*\left(FH^*\right)^*$, thus we get formally the same problem as the one solved in (Cottenceau et al, 2000) with H^* playing the role of transfer function H in the TEG setting. The following result adapted from Proposition 3 in (Cottenceau et al, 2000) is useful: If there exists a matrix $D \in R_{max}(A)^{n \times n}$ such that $\alpha \setminus y_{ref} / \beta = H^* D^*$ or there exists $\widetilde{D} \in R_{max}(A)^{n \times n}$ such that $\alpha \setminus y_{ref} / \beta = \widetilde{D}^* H^*$ then there exists the greatest F such that $H^*\left(FH^*\right)^* \le \alpha \setminus y_{ref} / \beta$, namely

$$F^{opt} = \left(\alpha H^*\right) \setminus y_{ref} / \left(H^* \beta\right). \tag{18}$$

Let us note that it is very difficult to compute the optimal feedback according to formula (18), although the formula is similar to the corresponding one used in feedback control of timed event graphs. In fact, dealing with formal power series in several non commutative variables from A is much more difficult than handling formal power series from $Z_{max}(\gamma)$. Similar simplificaton rules, the one proposed in (Benveniste et al, 1998a) and (Benveniste et al, 1998b), should be used in the computation according to formula (18). These simplificaton rules correspond to the fact that nondecreasing series are useful for practical computation.
In the special case we have restricted attention to, our methods yields the gretest feedback such that timing specification given by y_{ref} is satisfied, provided y_{ref} is of one of the above special forms. In our case of a controller with fixed logical structure only timed behavior is under control. At this point it is not clear yet how to leave the restriction on the form of y_{ref}. In timed event graphs this has been done by using the concept of compensator borrowed (extended) from the classical control theory. However there is no similar structure

for heap models, because there is no input function and we use another heap model as a controller.

If we are interested in manufacturing systems, where specificatons are given in terms of Petri nets, the reference output is not typically required to be met for all sequences of tasks, but only those having a real interpretation. These are given by the correponding (logical) Petri net language, say L. Thus, the problem is to find the greatest F, such that

$$\alpha H^*\left(FH^*\right)^* \beta \, char(L) \le y_{ref} char(L),$$

where $char(L) = \underset{w \in L}{\oplus} e.w$ is the series with Boolean coefficients, *i.e.* the formal series of language L. Let us recall (Gaubert & Mairesse, 1997) that such a restriction is formally realized by the tensor product (residuable operation) of the heap automaton with the logical (marking) automaton recognizing the Petri net language L, which is compatible with Theorem 4.1 of (Komenda et al, 2007).

Note that specifications based on (multivariable) formal power series are not easy to obtain in many practical problems, in particular those coming from production systems, often represented by Petri nets. In fact, given a reference output series amounts to solve a scheduling problem. A formal power series specification is not given, but it is to be found: e.g. using Jackson rule (Jacson, 1955).

6. Example

The following simple example is given in order to illustrate our approach. We consider the following simple timed Petri nets with their underlying heap models described below.

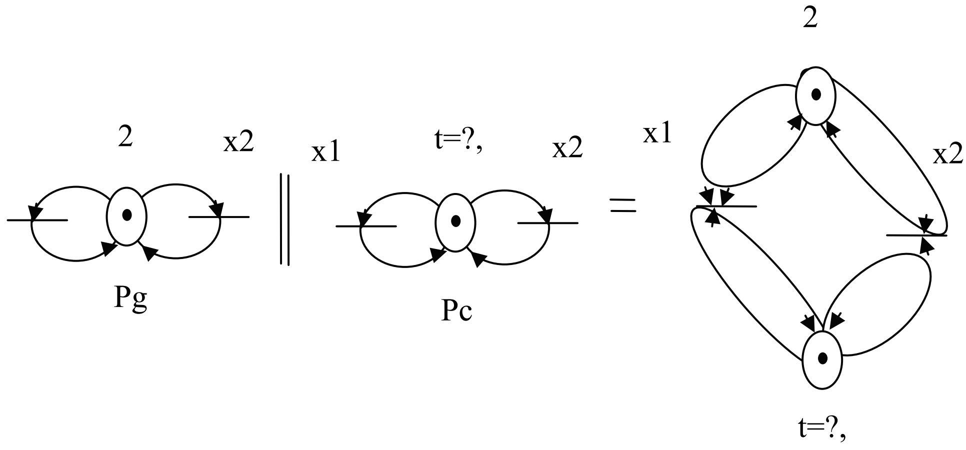

Fig. 4. Control of a simple heap model corresponding to TPNs above.

In the timed Petri nets above the timing (holdig time) of places *Pg* and *Pc* are *2* and *t*, respectively. In the controller net the value *t* is the control parameter.
The heap model *G* corresponding to the above simplest possible timed Petri nets with resource sharing is given below togetherwith its controller *C*.

$G=(A_g,P_g,r_g,l_g,u_g)$ and $C=(A_c,P_c,r_c,l_c,u_c)$ with $A=A_c=A_g=\{x_1,x_2\}$, $P_g=\{Pg\}$, $P_c=\{Pc\}$, $r_g(x_1)=r_g(x_2)=\{Pg\}$, $r_c(x_1)=r_c(x_2)=\{Pc\}$, $l_g(x_i,Pg)=\varepsilon$ for $i\in\{1,2\}$, $l_c(x_i,Pc)=\varepsilon$ for $i\in\{1,2\}$, $u_g(x_1,Pg)=2=u_g(x_2,Pg)$, and finally $u_c(x_1,Pc)=t=u_c(x_2,Pc)$.

$$\begin{aligned} x_G &= x_G \otimes \mu_G \oplus \alpha \\ x_C &= x_C \otimes \mu_C \oplus \alpha \end{aligned},$$

where

$\mu_G = 2x_1 \oplus 2x_2$ and $\mu_C = tx_1 \oplus tx_2$ are morphism matrices (here scalars of dimension 1).
In accordance with Theorem 2 we obtain

$$\begin{aligned} x_{G||C} &= x_{G||C} \otimes \begin{bmatrix} \mu_G & \overline{\mu}_C = \mu_C \\ \overline{\mu}_G = \mu_G & \mu_C \end{bmatrix} \oplus \alpha \\ y_{G||C} &= x_{G||C} \otimes \beta \end{aligned}$$

with

$$\mu_{G||C} = \begin{bmatrix} 2(x_1 \oplus x_2) & t(x_1 \oplus x_2) \\ 2(x_1 \oplus x_2) & t(x_1 \oplus x_2) \end{bmatrix}.$$

Let us choose the control reference (specification) as $y_{\text{ref}} = (4x_1 \oplus 4x_2)^*$. This specification is of the form

$$\alpha \backslash y_{ref} / \beta = y_{ref} = (2x_1 \oplus 2x_2)^* (4x_1 \oplus 4x_2)^* = H^* D^*.$$

We compute

$$F^{opt} = (\alpha H^*) \backslash y_{ref} / (H^* \beta) = H^* \backslash y_{ref} / H^* = (4x_1 \oplus 4x_2)^*.$$

Indeed, by definition

$$H^* \backslash y_{ref} = \max_x \{ H^* x \le y_{ref} \} = \max_x \{ (2x_1 \oplus 2x_2)^* x \le (4x_1 \oplus 4x_2)^* \} = (4x_1 \oplus 4x_2)^*,$$

and similarly $(4x_1 \oplus 4x_2)^* / H^* = (4x_1 \oplus 4x_2)^*$, whence the expected result. In the above computation we use the simple fact that for two series (here polynomial) s and t with $s \le t$, $s \ge e$, and $t \ge e$ we have $s^* t^* = t^*$.

Let us remark that if one would choose some different specification, e.g. $y_{\text{ref}} = (3x_1 \oplus 4x_2)^*$, then this specification is not compatible with the system, because there is only one place, where the timing is a control parameter. In order to achieve such a specification one would need a heap model corresponding to the following net structure.

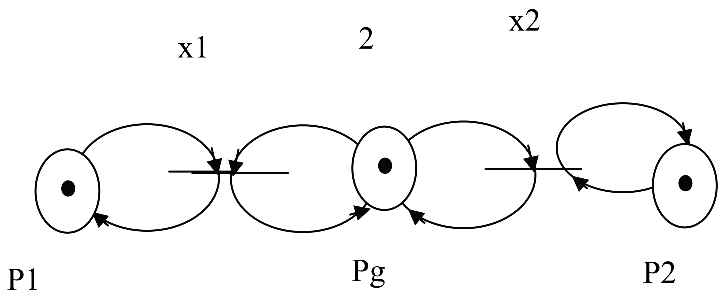

Fig. 5. Another TPN corresponding to a heap model allowing for more general specifications.

7. Concluding remarks

It has been shown how methods of dioid algebras can be used in supervisory control of heap models. We have proposed a synchronous product of heap models. The structure of the morphism matrix of synchronous product of two heap models is derived and applied to control of heap models.

The present reseach is a preliminary step in control of heap automata. We have limited our attention to a particular type of synchronous composition. In this work we have assumed that all events were controllable, i.e. any event may be delayed and even disabled (prevented from happening) by a suitable controller heap automaton. This assumption is however often unrealistic in practice: for instance one can hardly imagine that different kinds of system failures can always be avoided. Another restriction is the one we have imposed on the form of the reference input y_{ref} (in supervisory control also called control specification) . One possible way of leaving this restriction is to formulate heap automata in terms of input output automata and use the concept of precompensator from the classical control theory. Let us recall that recently different types of automata (e.g. classical automata, timed automata, stochastic automata) have been formulated in an equivalent way using explicit input and output functions as input output automata (e.g. IO timed automata and IO stochastic automata). It seems therefore interesting to work with IO (max,+) automata in order to extend the techniques based on precompensator from timed event graphs to our setting of heap automata.

Of potential interest is also supervisory control with partial controllability and partial observations or decentralized control of heap automata. Modular control of (explicitly) concurrent heap automata that are formed as synchronous compositions of heap models is particularly worthy to investigate.

8. Acknowledgement

KJB100190609, of the French-Czech bilateral project Barrande N. 14235XG and of the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503 are gratefully acknowledged.

9. References

Al Saba, M.; Boimond J.L &. Lahaye, S (2006). On just in time control of flexible manufacturing systems via dioid algebra. *Proceedings of INCOM'06*, Vol.2, pp. 137-142, Saint-Etienne, France.

Baccelli, F.; Cohen, G.; Olsder, G.J. & Quadrat, J.P. (1992). *Synchronization and linearity. An algebra for discrete event systems, John* Wiley & Son, New York.

Benveniste A. ; Jard C &. Gaubert S (1998a). Algebraic techniques for timed systems. *Proceedings of CONCUR'98, International Conference on Concurrency Theory,* 1998.

Benveniste A. ; Jard C &. Gaubert S (1998b). Monotone rational series and max-plus algebraic models of real-time systems. *Proceedings of the 4th Workshop on Discrete Event Systems, WODES'98*, Cagliari, Italy, august 1998.

Cottenceau, B.; Hardouin, L.; Boimond J.L &. Ferrier, J.L. (2001). Model Reference Control for Timed Event Graphs in Dioids. *Automatica,* Vol. 37, pp. 1451-1458.

Gaubert, S. (1992). *Theorie des systèmes linéaires dans les dioïdes.* Thèse de doctorat, Ecole des Mines de Paris, 1992.

Gaubert, S. (1995). Performance evaluation of (max,+) automata. *IEEE Transactions on Automatic Control,* Vol. 40, N12, pp. 2014-2025.

Gaubert, S. & Mairesse, J. (1997). Task resource models and (max,+) automata, In J. Gunawardena, Editor: *Idempotency.* Cambridge University Press, 1997.

Gaubert, S. & Mairesse, J. (1999). Modeling and analysis of timed Petri nets using heaps of pieces. *IEEE Transactions on Automatic Control,* Vol. 44, N4, pp. 683-698.

Jackson, J.R. (1955). Scheduling a Production Line to Minimize Maximum Tardiness. *Research report 43. University of California Los Angeles*. Management Science Research Project.

Komenda, J.; Al Saba, M. & Boimond, J.L. (2007). Supervisory Control of Maxplus Automata: Timing Aspects. *In Proceedings of the European Control Conference (ECC) 2007*, Kos (Greece).

Kumar, R. & Heymann, M. (2000). Masked prioritized synchronization for interaction and control of discrete-event systems. *IEEE Transaction Automatic Control 45,* 1970-1982, 2000.

Lin, F. & Wonham, W.M. (1998). On Observability of Discrete-Event Systems. *Information Sciences*, Vol. 44, pp. 173-198.

Menguy, E. (1997). Contribution à la commande des systèmes linéaires dans les dioïdes. *Thèse de doctorat,* Université d'Angers.

Ramadge, P.J. & Wonham, W.M. (1989). The Control of Discrete-Event Systems. *Proceedings of IEEE*, Vol. 77, pp. 81-98, 1989.

Sifakis, J. & Yovine, S. (1996). Compositional specification of timed systems. *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science, STACS'96*, pp. 347-359, . LNCS 1046.