SUPERVISORY CONTROL OF HEAP MODELS USING SYNCHRONOUS COMPOSITION

Jan Komenda
Mathematical Institute, Czech Academy of Sciences, Brno Branch Zizkova 22, 616 62 Brno, Czech Republic

Jean-Louis Boimond and Sébasten Lahaye
LISA Angers, 62, Avenue Notre Dame du Lac, 49000 Angers, France

Keywords: Heap models, synchronous product, supervisory control, dioid algebra.

Abstract: Heaps models are powerful models for concurrent timed discrete event systems. They admit linear description using dioid algebras. Inspired by supervisory control of logical discrete event systems we introduce parallel composition of heap models, called synchronous product, to formally describe the action of supervisor (represented by another heap model) on the system. This additional explicit concurrency for naturally concurrent heap models is useful for studying supervisory control in the algebraic framework of dioid algebras. Timing aspects of supervisory control, *i.e.* optimal timing of the controller, is studied based on residuation theory.

1 INTRODUCTION

There are two major streams in control theory of Discrete Event (dynamical) Systems (DES). The first stream, known as supervisory control theory, has been introduced by Wonham and Ramadge for logical automata (e.g. (Ramadge and Wonham, 1989)). The second stream more particularly deals with the class of timed Petri nets, called timed event graphs, based on linear representation in the (max,+) algebra. Being inspired by papers on (max,+) automata (e.g. (Gaubert and 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. However 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 and 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 (*e.g.* 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

Komenda J., Boimond J. and Lahaye S. (2007).
SUPERVISORY CONTROL OF HEAP MODELS USING SYNCHRONOUS COMPOSITION.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 467-475
DOI: 10.5220/0001636104670475
Copyright © SciTePress

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.

The paper is organized as follows. Algebraic preliminaries needed in the paper are recalled in the next section. 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. Residuation theory can be used in supervisory control of heap models.

2 DIOID ALGEBRAS

Basic algebraic structures and their properties that will be used in the paper are briefly presented in this section.

An idempotent semiring is a set M endowed with two inner operations. A commutative and associative addition denoted $\oplus$ that has a unit element ε and satisfies the idempotency condition ($\forall a \in M:\ a \oplus a = a$). A second operation, called multiplication, and denoted $\otimes$ is associative and has a unit element e, distributes over $\oplus$ both on the left and on the right, and $\forall a \in M:\ a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$. An idempotent semiring is said to be commutative if the multiplication $\otimes$ is commutative.

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 semiring M is called to be complete if any nonempty subset A of M admitss a least upper bound denoted by $\bigoplus_{x \in A} x$ and the distributivity axiom extends to infinite sums. Idempotent semirings are usually called dioids.

Let $\mathbb{N}$ denote the set of natural numbers with zero. In complete dioids the star operation can be introduced by the formula

$$a^* = \bigoplus_{n \in \mathbb{N}} a^n$$

with $a^0 = e$. Matrix dioids are introduced in the same manner as in the conventional linear algebra.

The simplest examples of commutative dioids are number dioids such as $\mathbb{R}_{max} = (\mathbb{R} \cup \{-\infty\}, max, +)$ with maximum playing the role of idempotent addition, denoted by $\oplus$: $a \oplus b = max(a,b)$, and conventional addition playing the role of multiplication, denoted by $a \otimes b$ or ab when no mistake is possible. Examples of non commutative dioids are matrix number dioids, formal languages and formal power series.

Let us recall from (Baccelli et al., 1992) and (Gaubert, 1992) the following results.

Theorem 2.1 *Let D be a complete dioid, x, a, b in D and*

$$x = x \otimes a \oplus b. \tag{1}$$

The least solution to equation (1) exists and is given by $b \otimes a^*$.

Lemma 2.2 *Let D be a complete dioid, a, b in D. Then $(a \oplus b)^* = (a^* b)^* a^* = a^*(ba^*)^* = b^*(ab^*)^* = (b^* a)^* b^*$.*

In the sequel we will work with the dioid of formal power series in the noncommutative variables from A and coefficients from $\mathbb{R}_{max}$ (corresponding to time). 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 $\mathbb{R}_{max}(A)$, where addition and (Cauchy or convolution) multiplication are defined as follows. For two formal power series in $\mathbb{R}_{max}(A)$:
$s = \oplus_{w \in A^*} s(w)w$ and $s' = \oplus_{w \in A^*} s'(w)w$, we have

$$s \oplus s' = \oplus_{w \in A^*} (s(w) \oplus s'(w))w \text{ and}$$

$$s \otimes s' = \oplus_{w \in A^*} (\oplus_{uv=w} s(u)s'(v))w.$$

This dioid is isomorphic to the dioid of generalized dater functions from A^* to $\mathbb{R}_{max}$ *via* a natural isomorphism similarly as the dioid $\mathbb{Z}_{max}(\gamma)$ of formal power series, used to study Timed Event Graphs (TEG), is isomorphic to the dioid of daters from $\mathbb{Z}$ to $\mathbb{Z}_{max}$. This isomorphism associates to any $y:\ A^* \to \mathbb{R}_{max}$ the formal power series $\oplus_{w \in A^*} y(w)w$ in $\mathbb{R}_{max}(A)$. The zero and identity series are denoted by ε and e, respectively, because it will be clear from the context whether a number dioid or dioid of formal power series is meant. Let us recall that $\forall w \in A^*:\ \varepsilon(w) = -\infty$ and

$$e(w) = \begin{cases} 0 & \text{if } w = 1 \\ -\infty & \text{if } w \neq 1. \end{cases}$$

We consider in the sequel the complete version of $\mathbb{R}_{max}(A)$ with coefficients in $R_{max} \cup \infty$. The notion of projection will be needed. Similarly as for formal languages, (natural) projection will be introduced such that it will be morphism with respect to both $\oplus$ and $\otimes$.

Finally, residuation of matrix multiplication will be needed in Section 5. 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 $f(x) \preceq b$ or the least solution to the inequality $b \preceq f(x)$ are considered. In the case these exist for all $b \in D$, the mapping f is called residuated, and dually residuated, respectively. The corresponding mappings that to any $b \in D$ associate the greatest, resp. least, solutions of the corresponding inequalities are called residuated, resp. dually residuated mappings. 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 $A \otimes X \preceq B$, is denoted $A \backslash B$. Similarly, the residuated mapping of the right matrix multiplication, *i.e.* the greatest solution to the inequality $X \otimes A \preceq B$, is denoted B / A. Recall from (Gaubert, 1992) that for matrices $A \in D^{m\times n}, B \in D^{m\times p}, C \in D^{n\times p}$ over a complete dioid D

$$A \backslash B \in D^{n\times p} : \ (A \backslash B)_{ij} = \bigwedge_{l=1}^{m} A_{li} \backslash B_{lj}$$

$$B / C \in D^{m\times n} : \ (B / C)_{ij} = \bigwedge_{k=1}^{p} B_{ik} / C_{jk}.$$

We recall the notation $\bigwedge_{s\in S} s$ for the infimal element of a set $S \subseteq D$ (recall that $\bigoplus_{s\in S} s$ is the corresponding supremal element of S when lattice structure of a complete dioid is considered).

3 SYNCHRONOUS COMPOSITION OF HEAP MODELS

Let us first recall the definition of time extension of heap models (Gaubert and Mairesse, 1999), also called task-resource systems, which model an important class of TDES. Formally, a heap model is the structure $\mathcal{H} = (A, R, r, l, u)$, where

- A is a finite set of pieces (also called tasks).
- R is a finite set of slots (also called resources).
- $r : A \to Pwr(R)$ gives the subset of slots required by a piece. It is assumed that $\forall a \in A : \ R(a) \neq \emptyset$.
- $l : A \times R \to \mathbb{R} \cup \{-\infty\}$ is function such that $l(a,r)$ gives the height of the lower contour of piece a at the slot r.
- $u : A \times R \to \mathbb{R} \cup \{-\infty\}$ is function such that $u(a,r)$ gives the height of the upper contour of piece a at the slot r. By convention, $u \geq l$, $l(a,r) = u(a,r) = -\infty$ if $r \notin R(a)$, and $\min_{r\in R(a)} l(a,r) = 0$.

The dynamics of heap models is described by row vectors $x(w)_r$, $w \in A^*$, $r \in R$ corresponding to the height of the heap $w = a_1 \dots a_n$ on slot $r \in R$. It has been shown in (Gaubert and 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 dynamic equations:

$$x(1) = (0 \dots 0) \quad (2)$$

$$x(wa) = x(w)\mu(a) \quad (3)$$

$$y(w) = x(w)(0 \dots 0)^T, \quad (4)$$

where

$$\mu(a)_{sr} = \begin{cases} 0, & \text{if } s = r \text{ and } s \notin R(a) \\ u(a,r) - l(a,s), & \text{if } r \in R(a) \text{ and } s \in R(a) \\ -\infty, & \text{otherwise} \end{cases} \quad (5)$$

is the morphism matrix associated to the heap model $\mathcal{H}$. It is shown in (Gaubert and Mairesse, 1999) that heap models are special (max,+)-automata with input and output functions as row, resp. column vectors of identity elements, and the morphism matrix above. The (max,+)-automaton given by the triple input function, output function, and the morphism matrix described above is then called heap automaton. Therefore we can associate heap models with special (max,+)-automata called heap automata.

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 and 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 3.1 *(Synchronous product) Let $\mathcal{H}_i = (A_i, R_i, r_i, l_i, u_i)$, $i = 1,2$ be two heap models with $R_1 \cap R_2 = \emptyset$. Their* synchronous product *is the heap model*

$$\mathcal{H}_1 \| \mathcal{H}_2 = (A_1 \cup A_2, R_1 \cup R_2, r, u, l) \quad (6)$$

$$(7)$$

with

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

$$u(a,r) = \begin{cases} u_1(a,r), & \mathit{if}\, r \in R_1 \\ u_2(a,r), & \mathit{if}\, r \in R_2 \end{cases},$$

and

$$l(a,r) = \begin{cases} l_1(a,r), & \mathit{if}\, r \in R_1 \\ l_2(a,r), & \mathit{if}\, r \in R_2 \end{cases}.$$

Since the slots (resources) of component heaps are disjoint, l and u are well defined: even though $A_1 \cap A_2 \neq \emptyset$, for any $r \in R$ there is only one $i \in \{1,2\}$, namely such that $r \in R_i$, with $l_i(a,r)$ 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 and Heymann, 2000)).

Note that if the above definition is used for control purposes, it is symmetric with respect to both the plant (say $\mathcal{H}_1$) and the controller (say $\mathcal{H}_2$). We then implicitly assume in the above definition that the supervisor is complete, *i.e.* that it never attemps to disable an uncontrollable task. This is always true if all tasks are controllable.

Now (max,+)-linear representation (2), (3), (4) of heap models will be used in 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 in Figure 1.

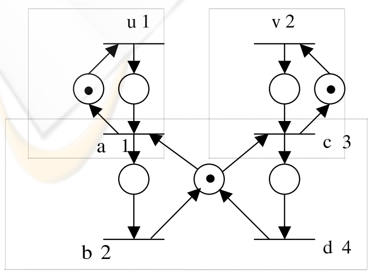

Figure 1: The Petri net model of a flexible manufacturing system.

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.

An order relation on $\mathbb{R}_{max}(A)$ 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 $\mathbb{R}_{max}(A)$. For $s, s' \in \mathbb{R}_{max}(A)$ we put $s \preceq s'$ iff $\forall w \in A^* : s(w) \leq s(w')$, where in the latter inequality usual order in R_{max} that coincides with the natural order is used.

Similarly as a TEG admits a linear representation in the dioid of formal power series $\mathbb{Z}_{max}(\gamma)$ (Baccelli et al., 1992), a heap model admits linear representation in the dioid of formal power series with noncommutative variables from A: $\mathbb{R}_{max}(A)$. As example let us consider the following heap automaton $\mathcal{H}$.

The set of tasks is $A = \{a,b,c,d\}$. The set of resources is $R = \{r_1, r_2, r_3\}$. Let $r(a) = r(b) = \{r_1, r_2\}$ and $r(c) = r(d) = \{r_1, r_3\}$. The lower and upper contours are given by

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

It has been shown in (Gaubert and Mairesse, 1999) that any heap model is a special (max,+)-automaton with the morphism matrix defined in equation (5). 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 $k \neq -\infty$, in case $k = -\infty$ there is no transition from i to j. Note that the morphism matrix μ of a heap model can be also considered as element of $R_{max}(A)^{|R|\times|R|}$, i.e. $\mu = \oplus_{w \in A^*} \mu(w) w$ by extending the definition of μ from $a \in A$ to $w \in A^*$ using the morphism property

$$\mu(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n).$$

However μ has an important property of being finitely generated, because it is completely determined by its values on A. For this reason we have in fact $\mu^* = (\oplus_{a \in A} \mu(a) a)^*$. Since we are interested in behaviors of heap models that are given in terms of μ^* we abuse the notation and write simply $\mu = \oplus_{a \in A} \mu(a) a$.

The corresponding heap automaton is in Figure 2 below. The state vector is associated to resources of $\mathcal{H}$ variables (formal power series) $x_2, x_1, x_3 \in \mathbb{R}_{max}(A)$ from left to right. We obtain the following

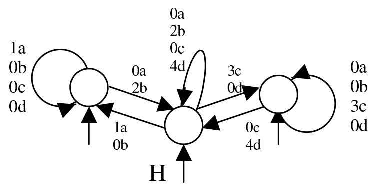

Figure 2: Example of a heap automaton.

equations in dioid $\mathbb{R}_{max}(A)$ endowed with pointwise addition and convolution multiplication:

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

The corresponding matrix form is

$$\begin{aligned} x &= x\mu \oplus \alpha \\ y &= x\beta, \end{aligned}$$

with $x = (x_1\ x_2\ x_3)$, $\alpha = (e\ e\ e)$, $\beta = (e\ e\ e)^T$, $\mu =$

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

In general we have the following linear description of (max,+) automata in the dioid $\mathbb{R}_{max}(A)$ of formal power series:

$$x = x\mu \oplus \alpha \qquad (8)$$

$$y = x\beta, \qquad (9)$$

where $\mu = \bigoplus_{a \in A} \mu(a)a \in \mathbb{R}_{max}(A)$ is also called the morphism matrix.

Recall that the least solution to this equation is $y = \alpha\mu^*\beta$.

4 MORPHISM MATRIX OF SYNCHRONOUS PRODUCT OF HEAP MODELS

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 Definition 3.1, 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 $l(a,r) = 0$ whenever $r \in R(a)$, which 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 $l(a,r) = -\infty$ for $r \notin R(a)$.

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two heap models with morphism matrices denoted by μ_1 and μ_2, respectively. Let $\varepsilon_{ij},\ i,j = 1,2$ be the rectangular matrices of zeros ($-\infty$) of dimensions $m_i \times m_j$ and $E_i,\ i = 1,2$ the square (max,+)-identity matrices of dimensions $m_i \times m_i$. We have the following block form of $\mu_{\mathcal{H}}(a),\ a \in A$.

Theorem 4.1 *The morphism matrix of $\mathcal{H} = \mathcal{H}_1 \| \mathcal{H}_2$ admits the following decomposition:*

- *If $a \in A_1 \cap A_2$ then:*

$$\mu_{\mathcal{H}}(a) = \begin{bmatrix} \mu_1(a) & \overline{\mu_2}(a) \\ \overline{\mu_1}(a) & \mu_2(a) \end{bmatrix}, \text{ where}$$

$$\overline{\mu_1}(a)_{ij} = \begin{cases} \mu_1(a)_{i'j} : \ i' \in r_1(a) \text{ arb.}, & \text{if } i \in r_2(a) \\ -\infty, & \text{if } i \notin r_2(a) \end{cases}$$

and

$$\overline{\mu_2}(a)_{ij} = \begin{cases} \mu_2(a)_{i'j} : \ i' \in r_2(a) \text{ arb.}, & \text{if } i \in r_1(a) \\ -\infty, & \text{if } i \notin r_1(a) \end{cases}$$

- *If $a \in A_1 \setminus A_2$ then :*

$$\mu_{\mathcal{H}}(a) = \begin{bmatrix} \mu_1(a) & \varepsilon_{12} \\ \varepsilon_{21} & E_2 \end{bmatrix}$$

- *If $a \in A_2 \setminus A_1$ then:*

$$\mu_{\mathcal{H}}(a) = \begin{bmatrix} E_1 & \varepsilon_{12} \\ \varepsilon_{21} & \mu_2(a) \end{bmatrix}$$

Proof For $a \in A = A_1 \cup A_2$ three cases must be distinguished. Firstly, $a \in A_1 \cap A_2$ is a shared task. Then for resources r, s from $R_1 \cup R_2$ there are four possibilities depending on whether these are in R_1 or R_2, whence the block form of $\mu_{\mathcal{H}}(a)$. We recall here that $R_1 \cap R_2 = \emptyset$. It is easy to see that in the diagonal blocks the individual morphism matrices appear. Also,

$$\overline{\mu_1}(a)_{ij} = \begin{cases} u(a,j) - l(a,i) = u_1(a,j) \\ \quad \text{if } i \in R_2(a) \text{ and } j \in R_1(a) \\ -\infty \\ \quad \text{if } i \notin R_2(a) \text{ or } j \notin R_1(a) \end{cases}$$

and similarly for $\overline{\mu_2}(a)$. 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 $-\infty$ otherwise, we can see that the rows corresponding to $i \in r_2(a)$ of $\overline{\mu_1}(a)$, i.e. $\overline{\mu_1}(a)(i,.)$, are the same as the rows $\mu_1(a)(i',.)$ for $i' \in r_1(a)$, which are all the same, *i.e.* an arbitrary one can be taken. Thus, the corresponding entry does not depend on j anymore. Hence, the morphism matrix of the composed heap $\mu_{\mathcal{H}}(a)$ has the claimed form. In the much easier situation when $a \in A_1 \setminus A_2$ it is sufficient to notice that no resource from R_2 is used by a. It is easily seen that $\mu_{\mathcal{H}}(a)$ has again the claimed form. Finally, the case $a \in A_2 \setminus A_1$ is symmetric.

Theorem 4.1 can be generalized to $n \in \mathbb{N}$, 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.

We recall at this point that

$$\mu_H = \bigoplus_{a\in A} \mu_H(a) \otimes a \in \mathbb{R}_{max}(A).$$

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

5 SUPERVISORY CONTROL OF HEAP MODELS

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. Since heap models are special (max,+)-automata there are two aspects of supervision: disabling and delaying of tasks (events). Here we are only interested in delaying the different tasks which 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.

Now we show how Theorem 4.1 can be used for control of heap models. The synchronous product of heap models $G = (A_g, R_g, r_g, l_g, u_g)$ and $C = (A_c, R_c, r_c, l_c, u_c)$ of dimensions m and n corresponds to the controlled (closed-loop) system. The event alphabet of the controlled system is denoted by A. According to Definition 3.1, we 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 $\mathbb{R}_{max}(A)$. The vector of formal power series from $\mathbb{R}_{max}(A)$ associated to generalized dater functions $x_{\mathcal{G}\|\mathcal{C}} : A^* \to \mathbb{R}_{max}^{m\times n}$ satisfies the following equations:

$$x_{\mathcal{G}\|\mathcal{C}} = x_{\mathcal{G}\|\mathcal{C}}\mu_{\mathcal{G}\|\mathcal{C}} \oplus \alpha, \quad (10)$$
$$y_{\mathcal{G}\|\mathcal{C}} = x_{\mathcal{G}\|\mathcal{C}}\beta, \quad (11)$$

where $\mu_{\mathcal{G}\|\mathcal{C}}$ is the morphism matrix of $\mathcal{G}\|\mathcal{C}$, α and β are row, resp. column, vectors of 0's of dimension $m+n$. The structure of the morphism matrix described in Theorem 4.1 is now used for control purposes. According to Theorem 2.1 the greatest solutions to equations (10) and (11) are

$$x_{\mathcal{G}\|\mathcal{C}} = \alpha\mu^*_{\mathcal{G}\|\mathcal{C}}, \quad (12)$$
$$y_{\mathcal{G}\|\mathcal{C}} = \alpha\mu^*_{\mathcal{G}\|\mathcal{C}}\beta, \quad (13)$$

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

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 task executions of the plant. This is done by the choice of upper contour functions (*i.e.* duration of controller's tasks) from $\mu_c(u)$. 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

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

In order to simplify the approach our attention is from now on limited to the case $A_c = A_g$, which is a standard assumption in the supervisory control with complete observations. Since state vector in equation (8) is associated to resources, it can be written as $x_{\mathcal{C}\|\mathcal{G}} = (x \ u)$, where the first component corresponds to the (uncontrolled) plant heap and the second to the controller heap. Owing to Theorem 4.1 we have:

$$(x \ u) = (x \ u) \bigoplus_{a\in A} \begin{bmatrix} \underline{H}(a)a & \overline{F}(a)a \\ \overline{H}(a)a & F(a)a \end{bmatrix} \oplus (\alpha_1, \alpha_2),$$

where α_1 and α_2 are vectors of zeros of corresponding dimensions, $\mu_g(a)$ is for convenience denoted by

$H(a)$, $\overline{\mu_g}(a)$ by $\overline{H}(a)$, $\mu_c(a)$ by $F(a)$, and $\overline{\mu_c}(a)$ by $\overline{F}(a)$. This can be written as

$$(x\ \ u) = (x\ \ u)\begin{bmatrix} A & \overline{F} \\ \overline{A} & F \end{bmatrix} \oplus \alpha,$$

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

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

Theorem 2.1 yields $u = (x\overline{F} \oplus \alpha_2)F^*$ as the least solution of the second equation, which substituted into the first equation leads to

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

The least solution is given by

$$x = (\alpha_2 F^*\overline{H} \oplus \alpha_1)(H \oplus \overline{F}F^*\overline{H})^*. \qquad (14)$$

The last equation can be viewed as the expression of the closed-loop system using the feedback given by F. Notice that unlike classical control theory the control variables have their inner dynamics, which is caused by adopting a supervisory control approach, where a controller is itself a dynamical system of the same kind as the uncontrolled system. 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.

There is a strong analogy with the feedback approach for control of TEG, see (Cottenceau et al., 2001), 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 F such that

$$y \le y_{ref},$$

thus,

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

We obtain from Lemma 2.2 that

$$(H \oplus \overline{F}F^*\overline{H})^* = H^*(\overline{F}F^*\overline{H}H^*)^*, \text{i.e.}$$

inequality (15) becomes

$$(\alpha_2 F^*\overline{H} \oplus \alpha_1)H^*(\overline{F}F^*\overline{H}H^*)^*\beta \le y_{ref} \qquad (16)$$

Note that while $\otimes$ and $\oplus$ are lower semicontinuous and residuated, the Kleene star is not in general, but only when the image is suitably constrained (in which case it is trivially residuated with identity as the corresponding residuated mapping). Moreover, as follows from Theorem 4.1, $\overline{F}$ can be simply 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 $\mu_{\mathcal{G}\|C}$ such that $y_{\mathcal{G}\|C}$ satisfies a given specification (e.g. is less than or equal to a given reference output series y_{ref}). In fact we obtain from the above inequality the greatest $\overline{F}F^*$ such that inequality (16) is satisfied. Thus it is not 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 $r_c : A \to Pwr(R_c)$) mimicks the structure of the plant, formally there exists an isomorphism between R_c and R_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), i.e. in the closed-loop system there are always parallel places of the controller corresponding to places of the uncontrolled 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 4.1 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 n. Hence, inequality (15) becomes $\alpha(F^*H \oplus E)(H \oplus \overline{F}F^*H)^*\beta \le y_{ref}$, where E is the identity matrix. An easy calculation yields $(H \oplus \overline{F}F^*H) = (E \oplus F^+)H = F^*H$, hence

$$(F^*H \oplus E)(F^*H)^* = (F^*H)^+ \oplus (F^*H)^* = (F^*H)^*.$$

Thus, $y = \alpha(F^*H)^*\beta \le y_{ref}$, *i.e.* the problem is to find the greatest F such that

$$(F^*H)^* \le \alpha \backslash\!\!\circ\, y_{ref} /\!\!\circ\, \beta.$$

Since the Kleene star is not a residuated mapping in general, such a problem has only a solution if $\alpha \backslash\!\!\circ\, y_{ref} /\!\!\circ\, \beta$, playing the role of reference model G_{ref} from (Cottenceau et al., 2001) is of a special form to be studied. Let us notice that $H \ge E$, which follows from the form of morphism matrix and the usual assumption that any resource of the system is used by at least one task: $\forall r \in R_g\ \exists a \in A$ such that $r \in r_g(A)$. The following Lemma is useful.

Lemma 5.1 *If $H \ge E$ then for any $B \in R_{max}(A)^{n\times n}$ every solution of $(X^*H)^* \le B$ is a solution of $H(X^*H)^* \le B$ and vice versa.*

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

$$H(X^*H)^* \leq X^*H(X^*H)^* = (X^*H)^+ \leq B,$$

where the first inequality follows from isotony of multiplication and $E \leq X^*$. Conversely, if X is a solution of $H(X^*H)^* \leq B$, then $(X^*H)^* = E \otimes (X^*H)^* \leq H(X^*H)^* \leq B$ as follows from isotony of multiplication and the assumption that $E \leq H$.

Using Lemma 5.1 our problem is to find the greatest solution in F of

$$H(F^*H)^* \leq \alpha \backslash y_{ref} / \beta.$$

It follows from Lemma 2.2 that

$$H(F^*H)^* = (H \oplus F)^* = H^*(FH^*)^*,$$

thus we get formally the same problem as the one solved in (Cottenceau et al., 2001) with H^* playing the role of transfer function H in the TEG setting. The following result adapted from (Cottenceau et al., 2001), *Proposition 3*, is useful: If there exists $D \in R_{max}(A)$ such that $\alpha \backslash y_{ref} / \beta = H^*D^*$ or there exists $D' \in R_{max}(A)$ such that $\alpha \backslash y_{ref} / \beta = D'^*H^*$ then there exists the greatest F such that $H^*(FH^*)^* \leq \alpha \backslash y_{ref} / \beta$, namely

$$F^{opt} = H^* \backslash [\alpha \backslash y_{ref} / \beta] / H^* = \alpha H^* \backslash y_{ref} / H^* \beta.$$

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 special forms. In the special case of a controller with fixed logical structure only timed behavior is under control.

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

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

where $char(L) = \bigoplus_{w \in L} e.w$ is the series with Boolean coefficients, *i.e.* the formal series of language L. Let us recall (Gaubert and Mairesse, 1999) that such a restricton 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 (Jackson, 1955).

6 CONCLUSION

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 very first step in control of heap automata. Sharing of resources is only allowed inside component heap models. Of potential interest is supervisory control with partial controllability, partial observations, and decentralized control of heap automata.

ACKNOWLEDGEMENTS

Partial financial support of Université d'Angers, of the Grant Agency GA AV No. KJB100190609, and of the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503 is gratefully acknowledged.

REFERENCES

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

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

B. Cottenceau, L. Hardouin, J.L. Boimond, and J.L. Ferrier. *Model Reference Control for Timed Event Graphs in Dioid,* Automatica, vol. 37, pp. 1451-1458, 2001.

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

S. Gaubert. *Performance evaluation of (max,+) automata,* IEEE Trans. on Automatic Control, 40(12), pp. 2014-2025, 1995.

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

S. Gaubert and J. Mairesse. *Modeling and analysis of timed Petri nets using heaps of pieces.* IEEE Trans. on Automatic Control, 44(4): 683-698, 1999.

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

J.Komenda, M.Al Saba, and J.L. Boimond. *Supervisory Control of Maxplus Automata: Quantitative Aspects.* In Proceedings ECC 2007, Kos (Greece).

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

F. Lin and W.M. Wonham, On Observability of Discrete-Event Systems, *Information Sciences*, 44: 173-198, 1

E. Menguy. *Contribution à la commande des systèmes linaires dans les diodes*. Thèse de doctorat, Université d'Angers, 1997.

P.J. Ramadge and W.M. Wonham. The Control of Discrete-Event Systems. *Proc. IEEE*, 77:81-98, 1989.

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