# SYNTHESIS OF SUPERVISORS FOR TIME-VARYING DISCRETE EVENT SYSTEMS

**Eduard Montgomery Meira Costa***
edmonty@ig.com.br

**Antonio Marcus Nogueira Lima**†
amnlima@dee.ufcg.edu.br

*Departamento de Engenharia Elétrica, Universidade Federal da Bahia
40210-630 Salvador, BA, Brasil

†Departamento de Engenharia Elétrica, Universidade Federal de Campina Grande
58109-970 Campina Grande, PB, Brasil

## ABSTRACT

We introduce a time-varying automaton to model discrete event systems. The structure of this time-varying automaton is very similar structure to (max,+) automaton, but allowing variable event lifetimes. Based on this time-varying automaton the design of timed supervisors is obtained by using the dioid algebra, where the languages used to describe the discrete event system as well the desired specification are replaced by matrices defined in such algebra and the supervisor synthesis is achieved through simple matrix operations. The proposed synthesis algorithm allows one to synthesize supervisors for un-timed DES, timed DES with constant event lifetime and timed DES with variable event lifetime. All these cases are treated with the same basic algorithm, the differences rely only on the definition of the event lifetime functions. The proposed algorithm presents a complexity order equal to the supervisor synthesis algorithm of un-timed discrete event systems. The proposed approach can be considered as an alternative procedure, based on a non-traditional algebraic structure, to achieve the supervisor synthesis for discrete event systems.

**KEYWORDS**: Time-Varying Automata, discrete event systems, supervisory control.

## RESUMO

O autômato com temporização variável é introduzido nesse artigo para modelar sistemas a eventos discretos. A estrutura desse autômato é bastante similar à estrutura do autômato (max,+), mas apresentando tempos de vida variáveis. Baseado nesse autômato o projeto de supervisores temporizados é obtido por meio da álgebra de dióides, onde as linguagens utilizadas para descrever o sistema a eventos discretos, bem como a especificação de comportamento desejada são definidas por matrizes descritas nesta álgebra e a síntese do supervisor é formalizada através de simples operações matriciais. O algoritmo de síntese proposto permite sintetizar supervisores para sistemas a eventos discretos não temporizados, sistemas a eventos discretos temporizados com tempos de vida constantes e e variáveis. Todos esses casos são tratados com o mesmo algoritmo básico, em que a diferença existe apenas na definição das funções de tempos de vida dos eventos. O algoritmo proposto apresenta uma ordem de complexidade igual ao algoritmo de síntese do supervisor para sistemas a eventos discretos não temporizados. A formulação proposta pode ser considerada como um procedimento alternativo baseado numa estrutura algébrica não tradicional, para construir supervisores para sistemas a eventos discretos.

**PALAVRAS-CHAVE**: Autômatos com Temporização Variável, Sistemas a Eventos Discretos, Controle Supervisório.

Artigo submetido em 24/04/2002
1a. Revisão em 31/07/2002; 2a. Revisão em 20/11/2003
3a. Revisão em 6/04/2004
Aceito sob recomendação do Ed. Assoc. Prof. Takashi Yoneyama

## 1 INTRODUCTION

Manufacturing systems, traffic systems, computer networks, communication protocols, process control plants are examples of discrete event systems (DES). A DES is a dynamic system in which state changes occur in response to the occurrence of events. The behavior of a DES can be expressed in terms of a language $L$ and the desired behavior is specified by another language $E$. In general, $L \subseteq E$ meaning that the DES may generate illegal behaviors. A supervisor is designed to eliminate the illegal behaviors. The set of events ($\Sigma$) is partitioned into controllable ($\Sigma_c$) and uncontrollable ($\Sigma_{uc}$) events thus creating the possibility for controlling the DES. Controllable events can be enabled or disabled by a supervisor which actively monitors the DES and can intervene at any moment to prevent an illegal behavior. The supervisor control problem can be elegantly solved by using the Supervisory Control Theory (SCT) (Ramadge e Wonham, 1987b; Ramadge e Wonham, 1987a; Ramadge e Wonham, 1982). Automata theory and formal languages theory form the basis of this theory. The supervisor as synthesized by the SCT deals only with the un-timed or logic behavior where there is no timing information explicitly associated to the events of the DES.

The timed behavior of DES is of great applied interest but also of significant complexity. Timed automata are one of the most studied models for representing real-time systems (Saksena e Selic, 1999). In some cases the timing information must be explicitly introduced to guarantee that the control action be feasible within the designated time bounds. Furthermore, these time bounds may be uncertain or time-varying and thus an adaptive supervisor must synthesized.

One approach to control a timed DES is to adopt a global clock and to introduce a clock-tick event (Brandin e Wonham, 1994). The use of the '*tick*' event, introduces new classes of events (prospective, remote, forcible, prohibitible and eligible) and increases the complexity of the problem since the state space is augmented in order to represent the timing information related to each activity in the system. In this case there are two representations for the DES: i) for displaying the activity transition graph (ATG) (Ostroff e Wonham, 1990) is employed and ii) the timed transition graph (TTG) (Lawford, 1997) where the timing information is explicitly represented by using '*tick*' transitions. The desired behavior is given by a timed-language and the classical supervisor synthesis provided by the SCT is applied to the TTG.

The use of a max-algebra provides another alternative to deal with some control problems of timed DES (Cofer e Garg, 1996). In this case the DES is represented by a timed-event graph and the dynamics is modelled by a system of linear equations written in a nontraditional algebraic structure known as a dioid. The desired behavior is given by a range of acceptable execution times for the events and a supervisor synthesis algorithm was developed since the max-algebra model prevents the use of the classical algorithm provided by the SCT. It is important to remark that the synthesis of timed and un-timed supervisors is usually done by employing different models for the DES.

This article proposes an unifying framework based on dioid algebra for synthesizing both timed and un-timed supervisors. It exploits the advantage of the dioid algebra for compactly representing the timing information and on the other hand allowing to express the DES model as well as the desired behavior in terms of languages. The proposed approach employs a time-varying automaton (TVA) to represent the DES by using its incidence matrix. The synthesis algorithm is based on the classical algorithm provided by the SCT and it is implemented in the matrix form.

This paper is organized as follows: in Section 2 some basic definitions concerning dioid algebra, (max,+) automaton, formal series and languages are presented; in Section 3 the time-varying automaton is introduced; in Section 4 the supervisor synthesis is formulated; in Section 5 the supervisor synthesis algorithm is presented; in Section 6 several illustrating examples are studied and in Section 7 the conclusions of this work are presented.

## 2 PRELIMINARIES

**Definition 1** A dioid is a set $D$ endowed with two operations: $\oplus$ (addition) and $\otimes$ (multiplication), that satisfy the following axioms:

**Axiom 1** Commutativity of $\oplus$:

$$a \oplus b = b \oplus a, \forall a, b \in D \tag{1}$$

**Axiom 2** Associativity of $\oplus$:

$$(a \oplus b) \oplus c = a \oplus (b \oplus c), \forall a, b, c \in D \tag{2}$$

**Axiom 3** Associativity of $\otimes$:

$$(a \otimes b) \otimes c = a \otimes (b \otimes c), \forall a, b, c \in D \tag{3}$$

**Axiom 4** Distributivity of $\otimes$ over $\oplus$:

$$(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c), \forall a, b, c \in D \tag{4}$$

$$c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b), \forall a, b, c \in D \tag{5}$$

**Axiom 5** Null element in $\oplus$:

$$\exists \epsilon \in D : \forall a \in D,\ a \oplus \epsilon = a \tag{6}$$

**Axiom 6** Absorbing element in $\otimes$:

$$a \otimes \epsilon = \epsilon \otimes a = \epsilon, \forall a \in D \tag{7}$$

**Axiom 7** Identity element in $\otimes$:

$$\exists e \in D : \forall a \in D,\ a \otimes e = e \otimes a = e \tag{8}$$

**Axiom 8** Idempotency in $\oplus$:

$$a \oplus a = a, \forall a \in D \tag{9}$$

A dioid is said commutative if $\otimes$ is commutative.

Considering that $D = \mathbb{R}_{\max} := \mathbb{R} \cup -\infty$, the null element is $\epsilon := -\infty$ and the identity element is $e := 0$. The $\oplus$ operator is the usual $\max$ and the $\otimes$ is the usual sum. In this case $D$ is commutative and the related algebra is known as the (max,+) algebra.

## 2.1 (max,+) automaton

The (max,+) automaton can be used to model the timed behavior of DES. In this case the system dynamics can be described through formal series as explained in the following and it is also possible to consider the partition of the set of events ($\Sigma$) into controllable ($\Sigma_c$) and uncontrollable ($\Sigma_{uc}$) events, i.e., $\Sigma = \Sigma_c \cup \Sigma_{uc}$, as usual in the SCT.

**Definition 2** A finite (max,+) automaton over an alphabet $\Sigma$ is a quadruple $A_{(\max,+)} = (Q, \theta, T, \phi)$ where $Q$ is a finite set of states, $\theta$, $\phi$, $T$ are maps $\theta : Q \to \mathbb{R}_{\max}$, $\phi : Q \to \mathbb{R}_{\max}$ $T : Q \times \Sigma \times Q \to \mathbb{R}_{\max}$ (named initial delays, final delays, and transition times, respectively).

For displaying purposes a (max,+) automaton is represented by valued multigraph where the set of vertices is given by $Q$ and there are three types of arcs:

i) The internal arcs $i \xrightarrow{\sigma} j, \forall i, j \in Q, \sigma \in \Sigma$. If $T_{i,\sigma,j} \neq \epsilon$ the arc $i \to j$ is valued by the scalar $T_{i,\sigma,j}$ like $i \xrightarrow{T_{i,\sigma,j}\,\sigma} j$;

ii) The input arcs $\to i, \forall i \in Q$. If $\theta_i \neq \epsilon$ the input arc is valued by the scalar $\theta_i$ like $\xrightarrow{\theta_i} i$;

iii) The output arcs $i \to, \forall i \in Q$. If $\phi_i \neq \epsilon$ the output arc is valued by the scalar $\phi_i$ like $i \xrightarrow{\phi_i}$.

**Example 1** Let $\Sigma = \{\alpha, \beta\}$. The automaton with set the of states $Q = \{0, 1, 2\}$, transition times $T_{0,\alpha,1} = 1$, $T_{0,\alpha,2} = 3$, $T_{1,\alpha,2} = 4$ , $T_{2,\beta,2} = 1$, $T_{2,\beta,1} = 5$, $T_{2,\beta,0} = 7$, $T_{1,\beta,1} = 1$, $T_{1,\beta,0} = 2$, final and initial delays $\phi_0 = 2$ and $\theta_0 = 0$, respectively (all the other values of $\phi$, $\theta$ and $T$ are equal to $\epsilon$) is shown in Figure 1. The values equal to 0 are omitted from the diagram, i.e., the not valued arc $\to 0$ stands for $\theta_0 = 0$.

Comparing a (max,+) automaton with an un-timed automaton having the same structure we can note that the transition function is embedded in the $T$ map. The initial state is defined at the vertex $q_0 \in Q$ in which $\theta_{q_0} \neq \epsilon$ and the marked states are those vertices $q_m \in Q$ where $\phi_{q_m} \neq \epsilon$. The dynamics of a (max,+) automaton can be explained as follows:

1. There is a global clock that is continuously being incremented;
2. The arcs are valued by $T_{i,\sigma,j} \in T$, $i, j \in Q$. This means that the transition from $i$ to $j$ takes at least $T_{i,\sigma,j}$ units of time, or, in other words, given that the automaton reached state $i$, then it will jump to state $j$ after the occurrence of $\sigma$, however the event $\sigma$ will be enabled to occur only after $T_{i,\sigma,j}$ units of time. The term $T_{i,\sigma,j}$ is denoted the lifetime of the event $\sigma$.
3. The initial state $q_0 \in Q$ will be reached only after $\theta_{q_0}$ units of time with respect to the global clock time origin;
4. When the automaton reaches a given state $i$ the counters associated to events such that $T_{i,\sigma,j} \neq \epsilon$ will be initialized with $T_{i,\sigma,j}$ and all start decrementing simultaneously. When a given counter reaches zero it stops decrementing and the respective event becomes enabled and may occur from now on;
5. When a given event occurs all the running counters are stopped and the state of the automaton changes;
6. When a marked state $q_m$ is reached it takes the final delay for the automaton to recognize the event sequence started at $q_0$;
7. If at a given state $i$ there are two arcs leaving that state and the labels are equal but valued with different lifetimes then the automaton is said non-deterministic.

**Example 2** The dynamics of the (max,+) automaton shown in Figure 2 can be described in the following way: i) After started the global clock at $t = 0$ it takes 2 units of time to reach the initial state 1. ii) At $t = 3$ the automaton may recognize the string $3\varepsilon$, where $\varepsilon$ is the empty string. iii) At

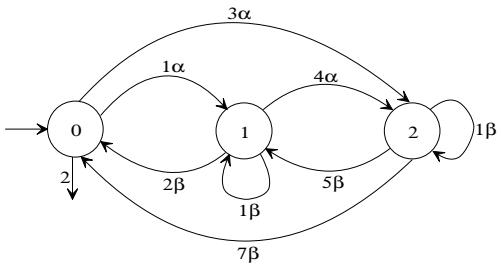


Figure 1: Example of a (max,+) automaton.

$t = 4$ the counter of event $\alpha$ reaches zero and it becomes enabled until its occurrence. iv) If $\alpha$ occurs at state 1 the automaton will jump to state 2. v) Once at state 2 it takes 3 units of time for event $\beta$ to become enabled. vi) If $\beta$ occurs at state 2 the automaton will jump to state 1. vii) Once again in state 1 it takes one more unit of time for the automaton to recognize the string $8\alpha\beta$. viii) If $\beta$ did not occurred at state 2 then after one more unit of time $\kappa$ becomes enabled. Then, after $t = 8$ the events $\beta$ and $\kappa$ are both enabled until one of them occurs. ix) If $\kappa$ occurs at state 2, the state 3 is reached. x) At state 3 it takes 2 units of time for $\beta$ to become enabled. xi) When $\beta$ occurs the automaton returns to state 1 and then, after one unit of time it recognizes the string $11\alpha\kappa\beta$.

In a (max,+) automaton a sequence of states is defined as a path. A path of length $n$ is given by

$$p = (q_0, ..., q_n) \in Q^n \tag{10}$$

where

$$Q^n := \{p | p = (q_0, \cdots, q_n) \wedge q_0, ..., q_n \in Q\}. \tag{11}$$

A string $s = \sigma_1 ... \sigma_n$ is recognized in a path if

$$W(p,s) := \theta(q_0) \otimes T_{q_0,\sigma_1,q_1} \otimes \cdots \otimes T_{q_{n-1},\sigma_n,q_n} \otimes \phi(q_n) \neq \epsilon \tag{12}$$

where $W(p, s)$ is the path weight function. In other words, a string is recognized if it takes a finite time for its completion. The multiplicity of a string is the maximum of the weights for all the paths where that string is recognized, namely

$$\left(A_{(\max,+)} | s\right) := \max_{p \in Q^n} W(p, s) \tag{13}$$

A dater is a map $y : \Sigma^* \to \mathbb{R}_{\max}$ and $(y|s)$ is interpreted as the time for completing the sequence of events that composes $s$. A (max,+) automaton recognizes a dater if

$$(y|s) = \left(A_{(\max,+)} | s\right) \tag{14}$$

**Example 3** The string $s = \alpha\beta$ is recognized by the automaton shown in Figure 2 since

$$P(p,s) = \theta_1 + T_{1,\alpha,2} + T_{2,\beta,1} + \phi_0 = 5 \neq \epsilon \tag{15}$$

with $p = (1, 2, 1)$.

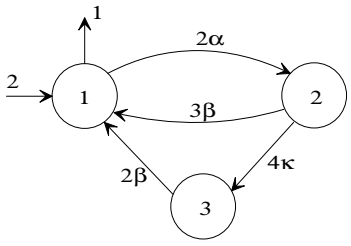


Figure 2: Explaining the dynamics of a (max,+) automaton.

**Example 4** The two automata shown in Figure 3(a) and (c) recognize the same timed-languages. For the sake of comparison, the automaton given in Figure 3(a) is an ATG (Brandin e Wonham, 1994) where the events are defined by $(\sigma_{q,q'}, t_\sigma, \infty) \in \Sigma$. Then, in terms of Brandin and Wonham notation we have that the events $(\alpha_{1,2}, 2, \infty)$, $(\beta_{2,3}, 2, \infty)$, $(\beta_{3,4}, 2, \infty)$, $(\alpha_{4,1}, 3, \infty)$, $(\kappa_{4,5}, 1, \infty)$, $(\lambda_{5,1}, 2, \infty)$ are all of remote type. The automaton shown in Figure 3(b) is a TTG representing the dynamics of the ATG. Figure 3(c) shows a (max,+) automaton that exhibits the same dynamics. In the TTG, each arc labelled with a $t$ ('*tick*') indicates one unit of time. It is worth noting that the number of states of the (max,+) automaton is largely inferior to the number of states of the TTG. Besides, the timing information is more compact and directly represented with the (max,+) automaton.

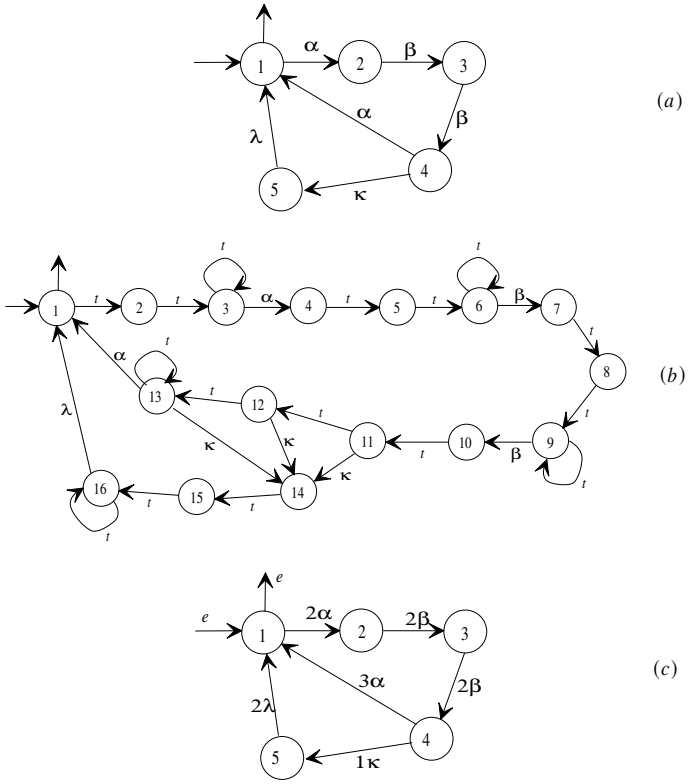


Figure 3: Automata comparison: (a) ATG, (b) TTG and (c) (max,+) Automaton.

## 2.2 (max,+) automaton and formal series

Formal series can be used to describe timed and un-timed languages (Berstel e Reutenauer, 1988; Klimann, 1999).

**Definition 3** A formal series $f$ over an alphabet $\Sigma$ with coefficients in $D$ is a map

$$f : \Sigma^* \to D \tag{16}$$

where each string $s \in \Sigma^*$ has its image $f(s) \in D$ that is denoted $(f|s)$ and represents the coefficient of $s$ in $f$.

The set of all the formal series over $f$ with coefficients in $D$ is denoted $D\langle\langle\Sigma\rangle\rangle$. Given $f_1 : \Sigma^* \to D$, $f_2 : \Sigma^* \to D$ and $\forall s \in \Sigma^*$ the set $D\langle\langle\Sigma\rangle\rangle$ is endowed with the following operations

$$(f_1 \oplus f_2|s) = (f_1|s) \oplus (f_2|s) \tag{17}$$

$$(f_1 \otimes f_2|s) = \bigoplus_{uv=s} (f_1|u) \otimes (f_2|v)\,. \tag{18}$$

These operations are known as the Cauchy sum and product, respectively. The Cauchy sum represents the union of languages and the Cauchy product represents the concatenation. The convention $(f|s) = \epsilon$ indicates that $s$ never occurs.

The star operation can also be defined for the formal series.

**Definition 4** The star operation for a formal series $f \in D\langle\langle\Sigma\rangle\rangle$ is defined by

$$f^* = e \oplus f \oplus f^2 \oplus f^3 \oplus \cdots \tag{19}$$

where '$e$' is the identity element and

$$f^n = \overbrace{f \otimes \cdots \otimes f}^{n \text{ times}}\,. \tag{20}$$

The formal series allows to describe languages by using equation (18). An un-timed language can also be described as formal series if $D = \mathbb{B} = \{\epsilon, e\}$ that is a binary semi-ring.

**Definition 5** A regular language $L = \{s_0, s_1, ...\} \subseteq \Sigma^*$ can be described by

$$Y_{\mathbb{B}\langle\langle\Sigma\rangle\rangle} = \bigoplus_{s\in\Sigma^*} (y|s)\, s \tag{21}$$

where $\mathbb{B}\langle\langle\Sigma\rangle\rangle$ is the semi-ring of formal series with $(y|s) \in \mathbb{B}$ and $s \in \Sigma^*$.

**Example 5** The language $L = \{\varepsilon, \alpha, \alpha\beta, \beta\alpha, \alpha\alpha, \beta\beta, \beta\alpha\beta\}$ defined over $\Sigma = \{\alpha, \beta\}$ can be represented by the following formal series

$$Y_{\mathbb{B}\langle\langle\Sigma\rangle\rangle} = e\varepsilon \oplus e\alpha \oplus e\alpha\beta \oplus e\beta\alpha \oplus e\alpha\alpha \oplus e\beta\beta \oplus e\beta\alpha\beta \underbrace{\oplus \epsilon\alpha\alpha\alpha \oplus \cdots}_{s\notin\Sigma^*-L} \tag{22}$$

where $(y|s) = e, \forall s \in L$, and $(y|s) = \epsilon, \forall s \in \Sigma^* - L$. The above formal series can also be written as

$$Y_{\mathbb{B}\langle\langle\Sigma\rangle\rangle} = e\varepsilon \oplus e\alpha \oplus e\alpha\beta \oplus e\beta\alpha \oplus e\alpha\alpha \oplus e\beta\beta \oplus e\beta\alpha\beta \tag{23}$$

or

$$Y_{\mathbb{B}\langle\langle\Sigma\rangle\rangle} = \varepsilon \oplus \alpha \oplus \alpha\beta \oplus \beta\alpha \oplus \alpha\alpha \oplus \beta\beta \oplus \beta\alpha\beta \tag{24}$$

since $\epsilon \otimes L = \epsilon$, $e \otimes L = L, \forall L \subseteq \Sigma^*$.

Similarly, a timed language (Alur e Henzinger, 1992; Tripakis, 1998; Fribourg, 1998; Asarin, 1998) can be described by a formal series defining that $D = \mathbb{R}_{\max}$. A timed language $L$ is a language where each $s \in L$ has a scalar value $t_s$ associated to it. This scalar is a time stamp that indicates how much units of time it takes for a timed automaton to recognize that string (Alur e Dill, 1990; Alur e Dill, 1994; Alur e Dill, 1995; Alur, 1997). This concept can be formalized by the following definition:

**Definition 6** A timed language $L = \{t_{s_0}s_0, t_{s_1}s_1, ...\}$ with $\{s_0, s_1, ...\} \in \Sigma^*$ and $t_{s_0}, t_{s_1}, ... \in \mathbb{R}_{\max}$, can be represented by

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = \bigoplus_{s\in\Sigma^*} (y|s)\, s \tag{25}$$

where $\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle$ is a semi-ring of formal series with coefficients in $\mathbb{R}_{\max}$ and noncommutative variables in $\Sigma$.

**Example 6** The language $L = \{3\varepsilon, 4\alpha, 2\alpha\beta, 3\beta\alpha, 5\alpha\alpha, 2\beta\beta, \beta\alpha\beta\}$ defined over $\Sigma = \{\alpha, \beta\}$ can be represented by

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = 3\varepsilon \oplus 4\alpha \oplus 2\alpha\beta \oplus 3\beta\alpha \oplus 5\alpha\alpha \oplus 2\beta\beta \oplus e\beta\alpha\beta \underbrace{\oplus \cdots}_{s\notin L} \tag{26}$$

where $(y|\varepsilon) = 3$, $(y|\alpha) = 4$, $\cdots$, $(y|\beta\alpha\beta) = e$. The above expression can be simplified to

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = 3\varepsilon \oplus 4\alpha \oplus 2\alpha\beta \oplus 3\beta\alpha \oplus 5\alpha\alpha \oplus 2\beta\beta \oplus e\beta\alpha\beta \tag{27}$$

or

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = 3 \oplus 4\alpha \oplus 2\alpha\beta \oplus 3\beta\alpha \oplus 5\alpha\alpha \oplus 2\beta\beta \oplus \beta\alpha\beta \tag{28}$$

since $\epsilon \otimes L = \epsilon$, $\forall L \subseteq \Sigma^*$.This series represents a language recognized by a (max,+) automaton. The meaning of $(y|s) = \epsilon$ is that the automaton does not recognizes the string $s$.

The above definitions allows to state that formal series can be employed to describe the automaton dynamics and to determine its marked language. Then, the language recognized by a (max,+) automaton can be described by its daters as

$$L_m = Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = \bigoplus_{s\in\Sigma^*} (y|s) \otimes s \tag{29}$$

with $s \in \mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle$.

In general, when there is no ambiguity, the $\otimes$ is omitted from the expressions. Thus, from now on the notation $ab$ meaning $a \otimes b$ will be used to represent this operation between any elements in the dioid algebra.

**Definition 7** The map $\mu : \Sigma \to \mathbb{R}_{\max}^{|Q|\times|Q|}$, is a map from $T_{i,\sigma,j}$, $\forall i, j \in Q$, $\sigma \in \Sigma$ over $\mathbb{R}_{\max}^{|Q|\times|Q|}$, where $\mathbb{R}_{\max}^{|Q|\times|Q|}$ is a dioid $(\mathbb{R}_{\max}, \max, +)$ defined for square matrices having dimension equal to $|Q| \times |Q|$.

By using this map we can construct a matrix $\mu(\sigma)_{ij} := T_{i,\sigma,j}$. From this matrix and by identifying $\Theta$ as a row vector with all the input arcs and $\Phi$ as column vector with all the output arcs we can write that:

$$(y|s) = \big(A_{(\max,+)}|s\big) = \Theta\mu(\sigma_1)\cdots\mu(\sigma_n)\,\Phi = \Theta\mu(s)\,\Phi \tag{30}$$

where $s = \sigma_1 \cdots \sigma_n$. Then, we say that the series $Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}$ is recognizable if exists a (max,+) automaton, represented by the triple $(\Theta, \mu, \Phi)$, $\Theta \in \mathbb{R}_{\max}^{1\times Q}$, $\Phi \in \mathbb{R}_{\max}^{Q\times 1}$, $\mu{:}\Sigma^* \to \mathbb{R}_{\max}^{Q\times Q}$ for finite $Q$ such that

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = \bigoplus_{s\in\Sigma^*} \Theta\mu(s)\,\Phi s \tag{31}$$

**Example 7** For the automaton shown in Figure 4 we have that

$$\Theta = \begin{bmatrix} e & \epsilon & \epsilon \end{bmatrix} \qquad \Phi = \begin{bmatrix} 2 \\ \epsilon \\ \epsilon \end{bmatrix}$$
$$\mu(\alpha) = \begin{bmatrix} \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & 4 \\ \epsilon & \epsilon & \epsilon \end{bmatrix} \quad \mu(\beta) = \begin{bmatrix} \epsilon & \epsilon & \epsilon \\ 2 & \epsilon & \epsilon \\ \epsilon & 5 & \epsilon \end{bmatrix}. \tag{32}$$

Computing $(y|s)$ when $s = \alpha\beta$ we obtain $(y|\alpha\beta) = \Theta\mu(\alpha\beta)\Phi = \Theta\mu(\alpha)\mu(\beta)\Phi = 5$. On the other hand, $Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}$ for this automaton is given by

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = 2(3\alpha(9\alpha\beta)^*\beta)^* = 2 \oplus 5\alpha\beta \oplus 8\alpha\beta\alpha\beta \oplus 14\alpha\alpha\beta\beta \oplus \ldots \tag{33}$$

Where $Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}$ represents all the strings formed by $\alpha$ and $\beta$, always starting with $\alpha$ and recognized by the automaton.

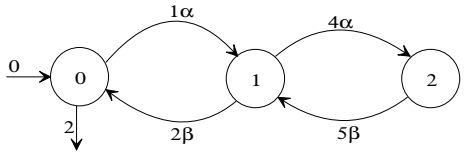


Figure 4: Deterministic (max,+) automaton.

In the present paper the basic tool for representing discrete event systems is the incidence matrix as defined in the following.

## 2.3 (max,+) automata and incidence matrices

**Definition 8** Let $A_{(\max,+)}$ a (max,+) automaton. Its incidence matrix represented by $\mathbf{A}$, is defined by

$$\mathbf{A} = [a_{i,j}]; a_{i,j} = \begin{cases} t_\sigma \sigma & \text{if } \exists\sigma \text{ from vertex } i \text{ to vertex } j; \\ \epsilon & \text{otherwise} \end{cases} \tag{34}$$

where $t_\sigma$ is the lifetime of $\sigma$. If there is more one event linking state $i$ to state $j$ then $a_{i,j} = \bigoplus_k t_{\sigma_k}\sigma_k$ is a regular expression. The initial state is always the state 1 and is represented by the row vector $\Theta(\mathbf{A}) = \begin{bmatrix} t_{in} & \epsilon & \cdots & \epsilon \end{bmatrix}$, where $t_{in}$ is the initial delay. The marked states are defined by the column vector $\Phi(\mathbf{A}) = \begin{bmatrix} t_{m_1} & t_{m_2} & \cdots & t_{m_n} \end{bmatrix}^T$, where $t_{m_i}$ represent the final delays.

**Example 8** The (max,+) automaton shown in Figure 4, has the following matrix representation

$$\mathbf{A} = \begin{bmatrix} \epsilon & 1\alpha & \epsilon \\ 2\beta & \epsilon & 4\alpha \\ 5\beta & \epsilon & \epsilon \end{bmatrix}$$
$$\Theta(\mathbf{A}) = \begin{bmatrix} e & \epsilon & \epsilon \end{bmatrix} \tag{35}$$
$$\Phi(\mathbf{A}) = \begin{bmatrix} 2 & \epsilon & \epsilon \end{bmatrix}^T$$

The incidence matrix can also be constructed by

$$\mathbf{A} = \bigoplus_{i=1}^{n} \mu(\sigma_i) \otimes \sigma_i \tag{36}$$

where $\mu(\sigma_i)$ is as defined by Definition 7. Then, the incidence matrix satisfies all the properties described in the previous section.

Formal languages can be used to construct languages from the timed incidence matrices. The following definition will help the construction of languages.

**Definition 9** Let $\mathbf{A}$ be an incidence matrix. An element $a_{i,j} = \sigma \in (\Sigma_c \cup \Sigma_{uc})$ can be interpreted as a path of length 1 through which the automaton jumps from state $i$ to state $j$ with a lifetime equal to $t_\sigma$. Then, the matrix

$$\mathbf{A}^n = \overbrace{\mathbf{A} \otimes \mathbf{A} \otimes \ldots \otimes \mathbf{A}}^{n \text{ times}} \tag{37}$$

is a path matrix where each element $a^n_{i,j}$ represents one or strings $s = \sigma_n \cdots \sigma_n$ of length $n$ from state $i$ to state $j$ and the total lifetime is $t_s = t_{\sigma_1} + \cdots + t_{\sigma_n}$. The initial and final vectors of $\mathbf{A}^n$ are the same of $\mathbf{A}$.

If there is no path from state $i$ to state $j$ then $a^n_{i,j} = \epsilon$.

**Example 9** The (max,+) automaton shown in Figure 5 can be represented by

$$\mathbf{A} = \begin{bmatrix} \epsilon & 4\alpha & \epsilon \\ 5\beta & \epsilon & 2\mu \\ \epsilon & 2\beta & 2\alpha \end{bmatrix},$$
$$\Theta(\mathbf{A}) = \begin{bmatrix} 1 & \epsilon & \epsilon \end{bmatrix}, \tag{38}$$
$$\Phi(\mathbf{A}) = \begin{bmatrix} 3 & \epsilon & 2 \end{bmatrix}^T$$

The path matrix $A^2$ for this automaton is

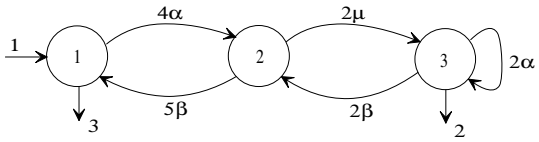


Figure 5: Strings in a timed incidence matrix: the path matrix of the automaton.

$$\mathbf{A}^2 = \mathbf{A} \otimes \mathbf{A} = \begin{bmatrix} 9\alpha\beta & \epsilon & 6\alpha\mu \\ \epsilon & 4\mu\beta + 9\beta\alpha & 4\mu\alpha \\ 7\beta\beta & 4\alpha\beta & 4\alpha\alpha + 4\beta\mu \end{bmatrix} \tag{39}$$

with $\Theta(\mathbf{A}^2) = \Theta(\mathbf{A})$ and $\Phi(\mathbf{A}^2) = \Phi(\mathbf{A})$. The path matrix $\mathbf{A}^3$ is

$$\mathbf{A}^3 = \begin{bmatrix} \epsilon & 13\alpha\beta\alpha + 8\alpha\mu\beta & 8\alpha\mu\alpha \\ 9\mu\beta\beta + 14\beta\alpha\beta & 6\alpha\mu\beta & 6\mu\alpha\alpha \\ 9\alpha\beta\beta & 11\beta\beta\alpha + 6\alpha\alpha\beta & 6\alpha\beta\mu + 6\alpha\alpha\alpha \\ & +6\beta\mu\beta & +6\beta\mu\alpha \end{bmatrix} \tag{40}$$

also with $\Theta(\mathbf{A}^3) = \Theta(\mathbf{A})$ and $\Phi(\mathbf{A}^3) = \Phi(\mathbf{A})$. In these matrices, each element represents a string of length 2 and 3, respectively. The string $\alpha\beta$ is recognized since

$$(y|\alpha\beta) = \Theta(\mathbf{A}) \otimes \mu(\alpha) \otimes \mu(\beta) \otimes \Phi(\mathbf{A}) = 13 \neq \epsilon, \tag{41}$$

according to Eq. (30).

Based on these operators it is possible to construct the timed language $L(\mathbf{A})$ as specified in the following definition.

**Definition 10** Given an incidence matrix $\mathbf{A}$, the language $L(\mathbf{A})$ is defined by

$$L(\mathbf{A}) = \bigoplus_i \left(\Theta(\mathbf{A}) \otimes \mathbf{A}^i\right) = \bigoplus_i \bigoplus_{j=1}^{n} \left(\theta_1(\mathbf{A}) \otimes a_{1,j}^i\right), \quad (42)$$

where $\theta_1(\mathbf{A})$ denotes the first element of the initial state row vector $\Theta(\mathbf{A})$ and $a_{1,j}^i \in \mathbf{A}^i$.

The following definition determines how to construct the marked language $L_m(\mathbf{A})$.

**Definition 11** Given an incidence matrix $\mathbf{A}$, the language $L_m(\mathbf{A})$ is defined by

$$L_m(\mathbf{A}) = \bigoplus_i \left(\Theta(\mathbf{A}) \otimes \mathbf{A}^i \otimes \Phi(\mathbf{A})\right) \quad (43)$$

$$L_m(\mathbf{A}) = \bigoplus_i \bigoplus_{j|\phi_j(\mathbf{A})\neq\epsilon}^{n} \left(\theta_1(\mathbf{A}) \otimes a_{1,j}^i \otimes \phi_j(\mathbf{A})\right) \quad (44)$$

where $\phi_j(\mathbf{A})$ is $j$-th element of the column vector $\Phi(\mathbf{A})$ and $a_{1,j}^i \in \mathbf{A}^i$.

**Example 10** The matrix representation for the automaton shown in Figure 6 is

$$\mathbf{A} = \begin{bmatrix} \epsilon & 5\alpha \\ 3\beta & \kappa \end{bmatrix}, \Theta(\mathbf{A}) = \begin{bmatrix} 1 & \epsilon \end{bmatrix}, \Phi(\mathbf{A}) = \begin{bmatrix} 3 \\ e \end{bmatrix}. \quad (45)$$

To determine its language we compute

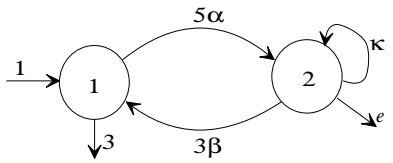


Figure 6: Timed language of a (max,+) automaton.

$$\mathbf{A}^2 = \begin{bmatrix} 8\alpha\beta & 5\alpha\kappa \\ 3\kappa\beta & 8\beta\alpha + \kappa\kappa \end{bmatrix}$$
$$\mathbf{A}^3 = \begin{bmatrix} 8\alpha\kappa\beta & 13\alpha\beta\alpha + 5\alpha\kappa\kappa \\ 11\beta\alpha\beta + 3\kappa\kappa\beta & 8\kappa\beta\alpha + 8\beta\alpha\kappa + \kappa\kappa\kappa \end{bmatrix} \quad (46)$$
$$\vdots$$

The elements of these matrices are strings of length 2, 3, and so on. Multiplying these matrices by $\theta_1(\mathbf{A})$ we found that

$$L(\mathbf{A}) = \{1\varepsilon, 6\alpha, 9\alpha\beta, 6\alpha\kappa, 9\alpha\kappa\beta, 14\alpha\beta\alpha, 6\alpha\kappa\kappa, ...\}. \quad (47)$$

Note that $\varepsilon \in L(\mathbf{A})$ and $t_\varepsilon = 1$ that is the initial delay of the automaton. On the other hand, all the strings of length $i$ are found at the first line of the $\mathbf{A}^i$ matrices. By multiplying the first line $\mathbf{A}^i$ by $\theta_1(\mathbf{A})$ and $\phi_k(\mathbf{A})$ we found the marked language of the automaton

$$L_m(\mathbf{A}) = \{4\varepsilon, 6\alpha, 12\alpha\beta, 6\alpha\kappa, 12\alpha\kappa\beta, 14\alpha\beta\alpha, 6\alpha\kappa\kappa, ...\}. \quad (48)$$

That is the same marked language we obtain by using formal series as given by

$$L_m(\mathbf{A}) = 4\varepsilon \oplus 6\alpha \oplus 12\alpha\beta \oplus 6\alpha\kappa \oplus 12\alpha\kappa\beta \oplus 14\alpha\beta\alpha \oplus 6\alpha\kappa\kappa ... \quad (49)$$

with $L_m(\mathbf{A}) = Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}$.

With these definitions, the languages related to the $A_{(\max,+)}$ automaton will be referred as $L(\mathbf{A})$ or $L_m(\mathbf{A})$ to make an explicit reference to its incidence matrix.

The reachability and co-reachability of a (max,+) automaton can be determined from its incidence matrix as defined below.

**Definition 12** The $j$-th row of an incidence matrix $\mathbf{A}$ is said reachable if $\exists i$, $i \in N^*$, such that

$$\Theta(\mathbf{A}) \otimes \mathbf{A}^i \otimes \Pi \neq \epsilon, \quad (50)$$

where $\Pi$ is a column vector, its $j$-th element is $\pi_j = e$ and all the other elements are $\pi_k = \epsilon$, $k \neq j$, $k = 1, \cdots, \dim(\mathbf{A})$.

The row $j$ is said reachable if, starting from the first row, there is at least one string $s \neq \epsilon$ that leads to row $j$.

**Definition 13** The row $i$ of an incidence matrix $\mathbf{A}$ is said co-reachable if $\exists k$, $k \in \mathbb{N}^*$, such that

$$\Upsilon \otimes \mathbf{A}^k \otimes \Phi(\mathbf{A}) \neq \epsilon, \quad (51)$$

where $\Upsilon$ is a row vector, its $i$-th element is $\upsilon_i = e$, and all the remaining elements are $\upsilon_k = \epsilon, k \neq i, k = 1, \cdots, \dim(\mathbf{A})$.

The row $i$ is said co-reachable if, starting from that row, there is, at least, one string $s \neq \epsilon$ that leads to a marked row $j$.

An incidence matrix is said reachable if all its rows are reachable. An incidence matrix is said co-reachable if all its rows are co-reachable. An incidence matrix is said trim if its both reachable and co-reachable at the same time.

**Example 11** The automaton shown in Figure 6 is reachable since

$$\Theta(\mathbf{A}) \otimes \mathbf{A}^1 \otimes \Phi = 6\alpha \neq \epsilon, \ \Phi = \begin{bmatrix} \epsilon & e \end{bmatrix}^T, \quad (52)$$

and

$$\Theta(\mathbf{A}) \otimes \mathbf{A}^2 \otimes \Phi = 9\alpha\beta \neq \epsilon, \Phi = \begin{bmatrix} e & \epsilon \end{bmatrix}^T. \quad (53)$$

This automaton is also co-reachable since

$$\Phi(\mathbf{A}) = \begin{bmatrix} 3 & e \end{bmatrix}^T, \quad (54)$$

$$\Upsilon \otimes \mathbf{A}^1 \otimes \Phi(\mathbf{A}) = 5\alpha \neq \epsilon, \Upsilon = \begin{bmatrix} e & \epsilon \end{bmatrix} \quad (55)$$

$$\Upsilon \otimes \mathbf{A}^1 \otimes \Phi(\mathbf{A}) = 6\beta \neq \epsilon, \Upsilon = \begin{bmatrix} \epsilon & e \end{bmatrix} \quad (56)$$

Then the automaton is trim.

The equivalence between timed-automata can be defined in terms of its respective incidence matrices as shown below.

**Definition 14** Let $\mathbf{A}$ and $\mathbf{B}$ denote two timed matrices, such that $\forall a_{i,j}, a_{i,j} \in \Sigma^*, \forall b_{i,j}, b_{i,j} \in \Sigma^*$. These matrices are said equivalent, denoted by $\mathbf{A} \equiv \mathbf{B}$, if for any two timed strings $s_{\mathbf{A}}$ $(\mathbf{A})$ and $s_{\mathbf{B}}$ $(\mathbf{B})$, such that $(y_{\mathbf{A}}|s_{\mathbf{A}}) = (y_{\mathbf{B}}|s_{\mathbf{B}})$, there exists a $\sigma \in \Sigma$ that is feasible both in $\mathbf{A}$ and $\mathbf{B}$, such that $(y_{\mathbf{A}}|s_{\mathbf{A}}\sigma) = (y_{\mathbf{B}}|s_{\mathbf{B}}\sigma)$.

**Example 12** The timed matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are equivalent

$$\mathbf{A}_1 = \begin{bmatrix} \epsilon & 2\alpha & \epsilon \\ 2\lambda & \epsilon & 3\beta \\ \epsilon & 3\beta & \epsilon \end{bmatrix}, \Theta(\mathbf{A}_1) = \begin{bmatrix} 2 & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{A}_1) = \begin{bmatrix} \epsilon \\ 3 \\ 3 \end{bmatrix}, \mathbf{A}_2 = \begin{bmatrix} \epsilon & 2\alpha \\ 2\lambda & 3\beta \end{bmatrix}$$
$$\Theta(\mathbf{A}_2) = \begin{bmatrix} 2 & \epsilon \end{bmatrix}, \Phi(\mathbf{A}_2) = \begin{bmatrix} \epsilon \\ 3 \end{bmatrix} \quad (57)$$

and thus exhibit the same timed language

$$L(\mathbf{A}_1) = L(\mathbf{A}_2) = \{2\varepsilon, 4\alpha, 6\alpha\lambda, 8\alpha\lambda\alpha, 7\alpha\beta, 10\alpha\beta\beta, \cdots\} \quad (58)$$

and its respective automata recognize the same marked language

$$L_m(\mathbf{A}_1) = L_m(\mathbf{A}_2) = \{7\alpha, 11\alpha\lambda\alpha, 7\alpha\beta, 10\alpha\beta\beta, \cdots\} \quad (59)$$

The automata represented by these two matrices are shown in Figure 7.

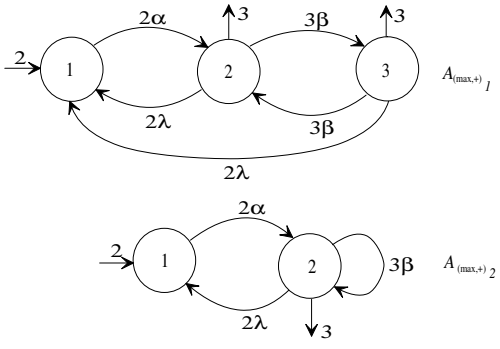


Figure 7: Two equivalent (max,+) automata.

## 2.4 Synchronous composition

To define the synchronous composition in terms of the timed incidence matrices it is necessary to introduce the following operator.

**Definition 15** Given a dioid $D = \mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle$, the operator $\circledast$ defines the intersection between elements as

$$a \circledast b = \bigoplus_{i=1}^{k} (t_{\sigma_i} \oplus t'_{\sigma_i})\sigma_i, \text{ if } t_{\sigma_i}\sigma_i \subset a \wedge t'_{\sigma_i}\sigma_i \subset b. \quad (60)$$

$\forall a, b \in D$, where

$$\begin{aligned} t_\sigma\sigma \circledast \epsilon &= \epsilon \\ t_{\sigma_1}\sigma_1 \circledast t_{\sigma_2}\sigma_2 &= \epsilon \\ t_\sigma\sigma \circledast t'_\sigma\sigma &= (t_\sigma\sigma \oplus t'_\sigma\sigma)\sigma, \end{aligned} \quad (61)$$

$\forall \sigma, \sigma_1, \sigma_2 \in \Sigma$ and $\forall t_\sigma, t_{\sigma_1}, t_{\sigma_2}, t'_\sigma \in \mathbb{R}_{\max}$. For timed incidence matrices the intersection operation is defined by

$$\mathbf{C} = \mathbf{A} \circledast \mathbf{B}, c_{i,j} = a_{i,j} \circledast b_{i,j}, \quad (62)$$

where $a_{i,j} \circledast b_{i,j}$ is as defined above with

$$\Theta(\mathbf{C}) = \Theta(\mathbf{A}) \circledast \Theta(\mathbf{B}) \text{ and } \Phi(\mathbf{C}) = \Phi(\mathbf{A}) \circledast \Phi(\mathbf{B}). \quad (63)$$

From this definition and considering that the elements of $\Theta(\cdot)$ and $\Phi(\cdot)$ belong to $\mathbb{R}_{\max}$, the $\circledast$ operation is equivalent to the $\oplus$ operation. However, when dealing with the control of a DES this definition must be restricted since the alphabet of events is partitioned as $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and only the controllable events $\Sigma_{uc}$ can be delayed by the supervisor.

**Definition 16** Consider that the incidence matrix $\mathbf{A}$ denotes the desired timed behavior for a given timed DES represented by an incidence matrix $\mathbf{B}$. The elements of the matrices $\mathbf{A}$ and $\mathbf{B}$ ($a_{i,j}$ and $b_{i,j}$) can be regular expressions like $t_{\sigma_1}\sigma_1 + ... + t_{\sigma_k}\sigma_k$ and $t'_{\sigma_1}\sigma_1 + ... + t'_{\sigma_k}\sigma_k$, respectively. In this case the $a_{i,j} \circledast b_{i,j}$ is defined by

$$a_{i,j} \circledast b_{i,j} = \begin{cases} \bigoplus_{i=1}^{k} (t_{\sigma_i} \oplus t'_{\sigma_i})\sigma_i & \text{if } (t_{\sigma_i}\sigma_i \subset a_{i,j}) \wedge (t'_{\sigma_i}\sigma_i \subset b_{i,j}) \\ & \qquad \wedge (\sigma_i \in \Sigma_c) \\ \bigoplus_{i=1}^{k} t'_{\sigma_i}\sigma_i & \text{if } (t_{\sigma_i}\sigma_i \subset a_{i,j}) \wedge (t'_{\sigma_i}\sigma_i \subset b_{i,j}) \\ & \qquad \wedge (\sigma_i \in \Sigma_{uc}). \end{cases} \quad (64)$$

The $\circledast$ operator as defined previously can be employed in the supervisor synthesis algorithm as it will be shown in the following.

To deal with the possibility of having uncontrollable events with variable lifetimes the synchronous product of timed incidence matrices must be defined as follows:

**Definition 17** Given two timed incidence matrices $\mathbf{A}$, $\dim(A) = m$, and $\mathbf{B}_{n\times n}$, $\dim(B) = n$, corresponding, respectively, to the automaton $A_{(\max,+)}$ with an alphabet of events $\Sigma_A$ and to the automaton $B_{(\max,+)}$ with an alphabet of events $\Sigma_B$. The timed incidence matrix $\mathbf{P}$ , $\dim(P) = q = mn$ and an alphabet of events given by $\Sigma = \Sigma_A \cup \Sigma_B$ corresponding to the automaton resulting from the synchronous composition of these two automata is defined as

$$\mathbf{P} = \mathbf{A}||\mathbf{B}, p_{k,l} = a_{i_{\mathbf{A}},j_{\mathbf{A}}} \circledast b_{i_{\mathbf{B}},j_{\mathbf{B}}} \quad (65)$$

where $k = i_{\mathbf{A}} + m\,(i_{\mathbf{B}} - 1)$ and $l = j_{\mathbf{A}} + m\,(j_{\mathbf{B}} - 1)$ such that

$$\mathbf{P}=\left[\begin{array}{c|c|c}
b_{1,1}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] & ... & b_{1,m}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] \\ \hline
\vdots & \vdots & \vdots \\ \hline
b_{1,1}\circledast[a_{n,1}\ a_{n,2}...a_{n,n}] & ... & b_{1,m}\circledast[a_{n,1}\ a_{n,2}...a_{n,n}] \\ \hline
b_{2,1}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] & ... & b_{2,m}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] \\ \hline
\vdots & \vdots & \vdots \\ \hline
b_{2,1}\circledast[a_{n,1}\ a_{n,2}...a_{n,n}] & ... & b_{2,m}\circledast[a_{n,1}\ a_{n,2}...a_{n,n}] \\ \hline
\vdots & \vdots & \vdots \\ \hline
b_{m,1}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] & ... & b_{m,m}\circledast[a_{1,1}\ a_{1,2}...a_{1,n}] \\ \hline
\vdots & \vdots & \vdots \\ \hline
b_{m,1}\circledast[a_{n,1}\ a_{n,2}...a_{n,n}] & ... & b_{m,m}\circledast[a_{n,1}\ a_{n,2}\ ...\ a_{n,n}]
\end{array}\right] \tag{66}$$

if $\Sigma_A = \Sigma_B$. The marked states of the composite automaton are defined by a column vector $\Phi_{q\times 1}\,(\mathbf{P})$ and its $k$-th element is given by

$$\phi_k\,(\mathbf{P}) = \phi_{i_{\mathbf{A}}}(\mathbf{A}) \circledast \phi_{i_{\mathbf{B}}}(\mathbf{B}), \tag{67}$$

with $k = i_{\mathbf{A}} + m\,(i_{\mathbf{B}} - 1)$. The initial state vector of the composite automaton a row vector defined as

$$\Theta_{1\times q}\,(\mathbf{P}) = \left[\ \theta_1(\mathbf{A}) \oplus \theta_1(\mathbf{B}) \quad \epsilon \quad \cdots \quad \epsilon\ \right]. \tag{68}$$

If $\exists\,\sigma_{\mathbf{A}} \notin \Sigma_A$, or $\exists\,\sigma_{\mathbf{B}} \notin \Sigma_B$, such that $\Sigma_A \cap \Sigma_B \neq \varnothing$, then

$$\mathbf{P}_{q\times q} = (\mathbf{A}||\mathbf{B}) \oplus \mathbf{C}^{\neg}, \tag{69}$$

where

$$\mathbf{C}^{\neg}=\left[\begin{array}{c|c|c|c}
\mathbf{A}^{\neg\mathbf{B}}\oplus\mathbf{D}^{\neg\mathbf{A}}_{(b_{1,1})} & \mathbf{D}^{\neg\mathbf{A}}_{(b_{1,2})} & ... & \mathbf{D}^{\neg\mathbf{A}}_{(b_{1,n})} \\ \hline
\mathbf{D}^{\neg\mathbf{A}}_{(b_{2,1})} & \mathbf{A}^{\neg\mathbf{B}}\oplus\mathbf{D}^{\neg\mathbf{A}}_{(b_{2,2})} & ... & \mathbf{D}^{\neg\mathbf{A}}_{(b_{2,n})} \\ \hline
\mathbf{D}^{\neg\mathbf{A}}_{(b_{3,1})} & \mathbf{D}^{\neg\mathbf{A}}_{(b_{3,2})} & ... & \vdots \\ \hline
\vdots & \vdots & ... & \mathbf{D}^{\neg\mathbf{A}}_{(b_{n-1,n})} \\ \hline
\mathbf{D}^{\neg\mathbf{A}}_{(b_{n,1})} & \mathbf{D}^{\neg\mathbf{A}}_{(b_{n,2})} & ... & \mathbf{A}^{\neg\mathbf{B}}\oplus\mathbf{D}^{\neg\mathbf{A}}_{(b_{n,n})}
\end{array}\right] \tag{70}$$

$\dim\,(C^{\neg}) = m \times n$, $A^{\neg\mathbf{B}}$ is the matrix with elements of $\Sigma_A$ that does not belong to $\Sigma_A$ and $\mathbf{D}^{\neg\mathbf{A}}_{(b_{i,j})}$ is a matrix having dimension $m \times m$ such that its diagonal contains the elements $b_{i,j}$ not defined in $\mathbf{A}$ while $\epsilon$ is put in all the other elements.

The algorithm for constructing the synchronous product $\mathbf{P} = \mathbf{A}||\mathbf{B}$ of two timed incidence matrices is given below:

**Algorithm 1** Algorithm for constructing the synchronous product $\mathbf{P} = \mathbf{A}||\mathbf{B}$

1. $j_{\mathbf{A}} \leftarrow 1$;
2. $j_{\mathbf{B}} \leftarrow 1$;
3. $i_{\mathbf{A}} \leftarrow 1$;
4. $i_{\mathbf{B}} \leftarrow 1$;
5. $\theta_1\,(\mathbf{P}) \leftarrow \theta_1\,(\mathbf{A}) \oplus \theta_1\,(\mathbf{B})$;
6. while $i_{\mathbf{B}} \leq n$, do:
   - **a)** if $b_{i_{\mathbf{B}},j_{\mathbf{B}}} = \epsilon$, do $j_{\mathbf{B}} \leftarrow j_{\mathbf{B}} + 1$;
   - **b)** if $j_{\mathbf{B}} > n$, do $i_{\mathbf{B}} \leftarrow i_{\mathbf{B}} + 1$ and $j_{\mathbf{B}} \leftarrow 0$;
   - **c)** while $i_{\mathbf{A}} \leq m$, do:
     - **i.** if $a_{i_{\mathbf{A}},j_{\mathbf{A}}} = \epsilon$, do $j_{\mathbf{A}} \leftarrow j_{\mathbf{A}} + 1$;
     - **ii.** if $j_{\mathbf{A}} > m$, do $i_{\mathbf{A}} \leftarrow i_{\mathbf{A}} + 1$ and $j_{\mathbf{A}} \leftarrow 0$;
     - **v.** $k \leftarrow i_{\mathbf{A}} + m\,(i_{\mathbf{B}} - 1)$;
     - **vi.** $l \leftarrow j_{\mathbf{A}} + m\,(j_{\mathbf{B}} - 1)$;
     - **vii.** $p_{k,l} \leftarrow a_{i_{\mathbf{A}},j_{\mathbf{A}}} \circledast b_{i_{\mathbf{B}},j_{\mathbf{B}}}$;
     - **viii.** $\phi_k\,(\mathbf{P}) \leftarrow \phi_{i_{\mathbf{A}}}\,(\mathbf{A}) \circledast \phi_{i_{\mathbf{B}}}\,(\mathbf{B})$;
     - **ix.** $j_{\mathbf{A}} \leftarrow j_{\mathbf{A}} + 1$;
     - **x.** if $j_{\mathbf{A}} > m$, do $i_{\mathbf{A}} \leftarrow i_{\mathbf{A}} + 1$ and $j_{\mathbf{A}} \leftarrow 0$;
   - **d)** $j_{\mathbf{B}} \leftarrow j_{\mathbf{B}} + 1$;
   - **e)** if $j_{\mathbf{B}} > n$, do $i_{\mathbf{B}} \leftarrow i_{\mathbf{B}} + 1$ and $j_{\mathbf{B}} \leftarrow 0$;
7. if $\exists\sigma_{\mathbf{A}} \notin \Sigma_2$ or $\exists\sigma_{\mathbf{B}} \notin \Sigma_1$, make $\mathbf{C}^{\neg}$, and do $p_{k,l} \leftarrow p_{k,l} \oplus c^{\neg}_{k,l}$.

**Remark 1** This algorithm is very similar to the algorithm employed for the synchronous composition of un-timed automata. The computational complexity of this algorithm has the order of $O\,(nm)$.

**Example 13** Given

$$\mathbf{A}=\left[\begin{array}{cc} \epsilon & 2\alpha \\ 3\beta & \epsilon \end{array}\right],\ \Theta\,(\mathbf{A}) = \left[\begin{array}{cc} e & \epsilon \end{array}\right],\ \Phi(\mathbf{A})=\left[\begin{array}{c} 3 \\ \epsilon \end{array}\right], \tag{71}$$

$$\mathbf{B}=\left[\begin{array}{ccc} \epsilon & 2\alpha & \epsilon \\ 2\beta & \epsilon & 3\eta \\ 2\lambda & 4\beta & \epsilon \end{array}\right],\Theta\,(\mathbf{B}) = \left[\begin{array}{ccc} 2 & \epsilon & \epsilon \end{array}\right],\Phi(\mathbf{B})=\left[\begin{array}{c} 2 \\ 1 \\ \epsilon \end{array}\right] \tag{72}$$

representing the automata shown in Figures 8(a) and 8(b), respectively, the synchronous product matrix $\mathbf{P}$ can be determined. The elements of such matrix defined by $p_{k,l} = p_{(i_{\mathbf{A}},i_{\mathbf{B}}),(j_{\mathbf{A}},j_{\mathbf{B}})}$, $k = i_{\mathbf{A}} + m\,(i_{\mathbf{B}} - 1)$ and $l = j_{\mathbf{A}} + m\,(j_{\mathbf{B}} - 1)$ when $a_{i_{\mathbf{A}},j_{\mathbf{A}}} = b_{i_{\mathbf{B}},j_{\mathbf{B}}} \neq \epsilon$, are given by

- $k = 1 + 2 \times 0 = 1, l = 2 + 2 \times 1 = 4, p_{1,4} = p_{(1,1),(2,2)} = a_{1,2} \circledast b_{1,2} = 2\alpha$
- $k = 2 + 2 \times 1 = 4, l = 1 + 2 \times 0 = 1, p_{4,1} = p_{(2,2),(1,1)} = a_{2,1} \circledast b_{2,1} = 3\beta$
- $k = 2 + 2 \times 2 = 6, l = 1 + 2 \times 1 = 3, p_{6,3} = p_{(2,3),(1,2)} = a_{2,1} \circledast b_{3,2} = 4\beta$

$$\mathbf{P} = \left[\begin{array}{cccccc}
\epsilon & \epsilon & \epsilon & 2\alpha & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
3\beta & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & 4\beta & \epsilon & \epsilon & \epsilon
\end{array}\right] \tag{73}$$

$$\Theta\,(\mathbf{P}) = \left[\begin{array}{cccccc} 2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{array}\right],$$
$$\Phi(\mathbf{P}) = \left[\begin{array}{cccccc} 3 & \epsilon & 3 & \epsilon & \epsilon & \epsilon \end{array}\right]^T$$

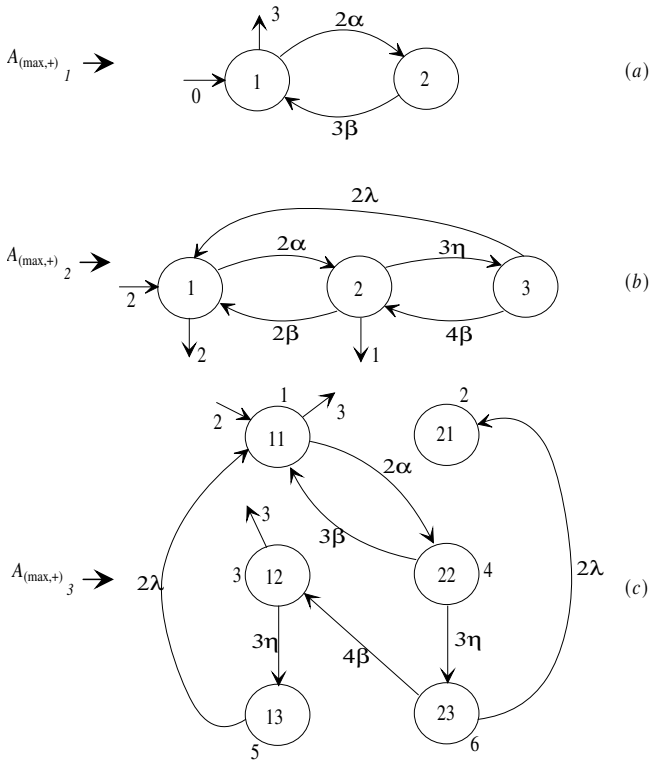


Figure 8: Illustrating the use of the synchronous composition: (a) $A_{(\max,+)_1}$, (b) $A_{(\max,+)_2}$ and (c) $A_{(\max,+)_3}$.

Considering that $\lambda, \eta \in \mathbf{B}$, but $\lambda, \eta \notin \mathbf{A}$, it is required to build the matrix $\mathbf{C}^{\neg}$ given by

$$\mathbf{C}^{\neg} = \begin{bmatrix} \mathbf{A}^{\neg\mathbf{B}} \oplus \mathbf{D}^{\neg\mathbf{A}}_{b_{1,1}} & \mathbf{D}^{\neg\mathbf{A}}_{b_{1,2}} & \mathbf{D}^{\neg\mathbf{A}}_{b_{1,3}} \\ \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} & \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} & \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} \\ \mathbf{D}^{\neg\mathbf{A}}_{b_{2,1}} & \mathbf{A}^{\neg\mathbf{B}} \oplus \mathbf{D}^{\neg\mathbf{A}}_{b_{2,2}} & \mathbf{D}^{\neg\mathbf{A}}_{b_{2,2}} \\ \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} & \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} & \begin{matrix} 3\eta & \epsilon \\ \epsilon & 3\eta \end{matrix} \\ \mathbf{D}^{\neg\mathbf{A}}_{b_{3,1}} & \mathbf{D}^{\neg\mathbf{A}}_{b_{3,2}} & \mathbf{A}^{\neg\mathbf{B}} \oplus \mathbf{D}^{\neg\mathbf{A}}_{b_{1,1}} \\ \begin{matrix} 2\lambda & \epsilon \\ \epsilon & 2\lambda \end{matrix} & \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} & \begin{matrix} \epsilon & \epsilon \\ \epsilon & \epsilon \end{matrix} \end{bmatrix} \quad (74)$$

and thus

$$\mathbf{P} = \mathbf{P} \oplus \mathbf{C}^{\neg} = \begin{bmatrix} \epsilon & \epsilon & \epsilon & 2\alpha & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 3\eta & \epsilon \\ 3\beta & \epsilon & \epsilon & \epsilon & \epsilon & 3\eta \\ 2\lambda & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 2\lambda & 4\beta & \epsilon & \epsilon & \epsilon \end{bmatrix} \quad (75)$$

where $\Theta(\mathbf{P})$ and $\Phi(\mathbf{P})$ are given by (73). Finally, matrix $\mathbf{P}$ together with $\Theta(\mathbf{P})$ and $\Phi(\mathbf{P})$ represent the automaton $A_{(\max,+)_3}$ shown in Figure 8(c).

It must be noted that all the elements of row 2 are such that $p_{2,j} = \epsilon$ and it can be reached after the string $s = \Theta(\mathbf{P}) \otimes \mathbf{P}^3 \otimes \pi = p_{1,4}p_{4,6}p_{6,2} = 9\alpha\eta\lambda$, for

$$\pi = \begin{bmatrix} \epsilon & e & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}^T. \quad (76)$$

This shows that from the composition of $A_{(\max,+)_1}$ with $A_{(\max,+)_2}$ may result an automaton with a blocking state.

**Remark 2** The definition 17 establishes that the composition of two automaton when $\Sigma_{\mathbf{A}} \cap \Sigma_{\mathbf{B}} = \varnothing$ requires the determination of the matrix $\mathbf{C}^{\neg}$. It is worth noting that such composition is based on the operator $\circledast$ (see Definition 15). However, when the alphabet of events $\Sigma$ is partitioned into controllable $\Sigma_c$ and uncontrollable events $\Sigma_{uc}$, the intersection operator $\circledast$ must be redefined according to Definition 16.

# 3 TIME-VARYING AUTOMATON

Based on the formalism outlined the previous sections we introduce here an extended version of the (max,+) automaton named time-varying automaton (TVA).

**Definition 18** A time-varying automaton TVA over an alphabet $\Sigma$ is a quintuple given by

$$TVA = (Q, \Sigma, q_0, t_0, t_i, t_f), \quad (77)$$

where:

- $Q = \{q_0, ..., q_n\}$ is a finite set of states;
- $\Sigma = \{\sigma_1, ..., \sigma_m\}$ is an alphabet of events;
- $q_0$ is the initial state;
- $t_0 : q_0 \rightarrow R_{\max}$ is the initial delay;
- $t_i : t_{i-1} \times Q \times \Sigma \times Q \rightarrow R_{\max}$ is the transition function and
- $t_f : t_{i-1} \times Q \rightarrow R_{\max}$ is the final delay function.

The graphical representation of an TVA is a graph constituted by vertices representing $Q$ and three types of arcs:

1. Internal arcs $q_j \stackrel{\sigma}{\rightarrow} q_{j+1}, \forall q_j, q_{j+1} \in Q, \sigma \in \Sigma$ such that $t_i \neq \epsilon$. The $q_j \stackrel{\sigma}{\rightarrow} q_{j+1}$ is valued by $t_i = f(t_{i-1})$;
2. Input arc $\rightarrow q_0$ valued by $t_0 \neq \epsilon$;
3. Output arcs $q_j \rightarrow$ valued by $t_f$, $\forall q_f \in Q$ such that $t_f = f(t_{j-1}) \neq \epsilon$.

Note that the (max,+) automaton is simply a special case of the TVA automaton when all the transition functions are considered to be constants. Then, consequently, the dynamics of the TVA is quite similar to the dynamics of the (max,+) automaton.

1. There is a global clock that is continuously being incremented;

2. Given that the automaton reached state $i-1$, then it will jump to state $i$ after the occurrence of $\sigma$, however the event $\sigma$ will be enabled to occur only after $t_i$ units of time. In a TVA $t_i$ is not constant as in a (max,+) automaton but depends on $t_{i-1}$, i.e., $t_i = f(t_{i-1})$.
3. The initial state $q_0 \in Q$ will be reached only after $t_0$ units of time with respect to the global clock time origin;
4. When the automaton reaches a given state $i$ the counters associated to events $t_i\,(\sigma)$ all start decrementing simultaneously. When a given counter reaches zero it stops decrementing and the respective event becomes enabled and may occur from now on;
5. When a given event occurs all the running counter are stopped and the state of the automaton changes;
6. If at a given state $i$ there are two arcs leaving that state and the labels are equal but valued with different transition times then the TVA is said non-deterministic.

**Example 14** Let $\Sigma = \{\alpha, \beta\}$, $Q = \{1, 2, 3\}$, $t_0 = 2$, the following transition functions

$$\begin{array}{l} f_1(t) = t^2 - t, \\ f_2(t) = 3t, \\ f_3(t) = 4, \\ f_4(t) = 1 + 2t \end{array} \tag{78}$$

and the final delay function

$$f_5(t) = 5 - 2/t. \tag{79}$$

The other transition functions are equal to $\epsilon$. The corresponding automaton is shown in the Figure 9. The $t$ in the above expressions is measured in units of time (of a given time base) must be interpreted as the last lifetime, i.e., $t_{i-1}$ as explained previously. The value of the lifetime for $\alpha$ related with the arc $q_1 \overset{\alpha}{\rightarrow} q_2$ is

$$t_1 = f_1\,(2) = 2, \tag{80}$$

since $t_0 = 2$. The final delay for recognizing $\varepsilon$ (if $\alpha$ does not occur) is

$$t_f = f_5(2) = 4, \tag{81}$$

and consequently the total time it takes for recognizing $\varepsilon$ is 6 units of time. However, if $\alpha$ occurs the automaton jump to state 2 where the lifetime for $\alpha$ is

$$t_2 = f_2(2) = 6 \tag{82}$$

and the lifetime for $\beta$ is

$$t_2 = f_4(2) = 5. \tag{83}$$

If at state 2, $\alpha$ occurs the automaton jumps to state 3 where the lifetime for $\beta$ is

$$t_3 = f_3(5) = 4. \tag{84}$$

If at state 2, $\beta$ occurs the automaton jumps to state 1 where the lifetime for $\alpha$ is

$$t_3 = f_1(6) = 30, \tag{85}$$

the final delay for recognizing $\alpha\beta$ is

$$t_f = f_5\,(5) = 4.6, \tag{86}$$

and consequently the total time it takes for recognizing $\alpha\beta$ is 13.6 units of time.

It is worth noting that the recognition of a given string occurs when a marked state is reached and the final delay has passed. This string recognition process can also be described by the daters of the strings as presented in the following:

**Definition 19** A dater is a map given by

$$Y : \Sigma^* \rightarrow \mathbb{R}_{\max}, \tag{87}$$

where $Y$ is the time it takes for the string

$$s = \sigma_1\sigma_2...\sigma_n \in \Sigma^*. \tag{88}$$

to be processed within a TVA.

The dater defines the lifetimes related to the strings of $\Sigma^*$. Consequently, a TVA takes $Y$ units of time to change its state from $q$ to $q'$ by processing $s$, where $(y|s)$ denotes the value of $Y$ for the string $s$.

**Definition 20** A dater is said recognized if there exists a TVA such that

$$(y|s) \neq \epsilon. \tag{89}$$

**Example 15** For the automaton of the Example 14, we can see that $s = \alpha\beta$ is a recognized string since

$$(y|\alpha\beta) = (t_0 + f_1(t_0) + f_4(t_1) + t_f) = 13,6 \neq \epsilon. \tag{90}$$

## 3.1 TVA and formal series

The use of formal series to define the dynamics of a TVA is quite similar what has been done for the (max,+) automaton. Thus, it follows that:

**Definition 21** A timed language

$$L = \{t_s s, t_{s'} s', ...\}, \tag{91}$$

with $s, s', ... \in \Sigma^*$ and $t_s, t_{s'}, ... \in \mathbb{R}_{\max}$, can be represented by a formal series

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = \bigoplus_{s\in\Sigma^*} (y|s)\, s, \tag{92}$$

where

$$\mathbb{R}_{\max}\,\langle\langle\Sigma\rangle\rangle \tag{93}$$

denotes the semi-ring of the formal series having coefficients in $\mathbb{R}_{\max}$ and non commutative variables in $\Sigma$.

**Example 16** The language $L = \{3\varepsilon, 4\alpha, 2\alpha\beta, 3\beta\alpha\}$ over $\Sigma = \{\alpha, \beta\}$ can also be represented by

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle} = 3\varepsilon \oplus 4\alpha \oplus 2\alpha\beta \oplus 3\beta\alpha = 3 \oplus 4\alpha \oplus 2\alpha\beta \oplus 3\beta\alpha \tag{94}$$

since $\epsilon \otimes L = \epsilon$, $\forall L \subseteq \Sigma^*$. This series represents the language recognized by a TVA. The term $(y|s)$ denotes the coefficient of the string $s$ that is equal to '$\epsilon$' if $s$ is not recognized by the TVA.

According to this formalism, a dater function $Y$ can be described by using a formal series defined over $\Sigma$ with coefficients defined in $\mathbb{R}_{\max}$, as presented in the following:

**Definition 22** A timed language of a TVA is defined by a formal series

$$L\left(TVA\right)=\bigoplus_{s\in\Sigma^{*}}\left(y|s\right)s, \tag{95}$$

where $(y|s)\in\mathbb{R}_{\max}$ denotes the dater of the string $s\in\Sigma^{*}$.

**Example 17** For the automaton of the Example 14, the formal series $Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}$ related to the recognized timed language is given by

$$Y_{\mathbb{R}_{\max}\langle\langle\Sigma\rangle\rangle}=6\oplus13,6\alpha\beta\oplus74,95\alpha\beta\alpha\beta\oplus27,78\alpha\alpha\beta\beta\oplus... \tag{96}$$

## 3.2 TVA and incidence matrices

The definition of an incidence matrix for a TVA follows similar rules as it was done for the (max,+) automaton. However, instead of fixed lifetimes, the elements of the incidence matrix will be related with the transition functions as shown in the following:

**Definition 23** Given a TVA, its incidence matrix, denoted by $A$, is defined as

$$\mathbf{A}=[a_{i,j}];\ a_{i,j}=\begin{cases}f\left(t_{\sigma}\right)\sigma & \text{if } \exists\sigma \text{ from vertex } i \text{ to vertex } j;\\ \epsilon & \text{otherwise,}\end{cases} \tag{97}$$

where $f\left(t_{\sigma}\right)$ represents the lifetime of event $\sigma$. The automaton jumps from state $i$ to state $j$ upon the occurrence of $\sigma$. If more than one string may provoke the jump from state $i$ to state $j$ then the respective element of the matrix must be written as a timed regular expression like $a_{i,j}=\bigoplus_{k}f\left(t_{\sigma^{k}}\right)\sigma^{k}$. For mathematical convenience, the state 1 is always considered as the initial state and can be represented by the row vector $\Theta(\mathbf{A})=[\ t_0\quad\epsilon\quad...\quad\epsilon\ ]$, the marked states are also represented by the column vector $\Phi(\mathbf{A})=[\ f(t_{f_1})\quad f(t_{f_2})\quad...\quad f(t_{f_n})\ ]^{T}$.

**Example 18** The TVA shown in Figure 9 has the following matrix representation

$$\mathbf{A}=\begin{bmatrix}\epsilon & (t^{2}\text{-}t)\alpha & \epsilon\\ (1+2t)\beta & \epsilon & (3t)\alpha\\ \epsilon & 4\beta & \epsilon\end{bmatrix},\Theta(\mathbf{A})=[2\ \epsilon\ \epsilon\ ],\Phi(\mathbf{A})=\begin{bmatrix}5\text{-}2t\\ \epsilon\\ \epsilon\end{bmatrix} \tag{98}$$

The $t$ in the above matrices and vectors is measured in units of time (of a given time base) and must be referred to the last transition time.

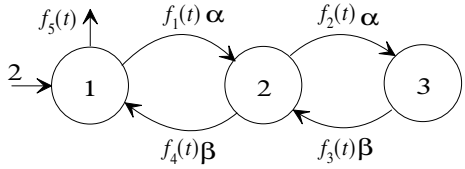


Figure 9: Example of a time-varying automaton.

All the matrix formalisms already presented for the (max,+) are also valid for the time-varying automaton. However, when defining languages by using path matrices we must consider that the lifetimes change at every automaton execution as illustrated by the following example:

**Example 19** The matrix representation for the TVA shown in Figure 10 is given by

$$\mathbf{A}=\begin{bmatrix}\epsilon & 2t\alpha\\ (2\text{-}t)\beta & \epsilon\end{bmatrix},\ \Theta\left(\mathbf{A}\right)=\begin{bmatrix}2 & \epsilon\end{bmatrix},\ \Phi\left(\mathbf{A}\right)=\begin{bmatrix}3t\\ \epsilon\end{bmatrix} \tag{99}$$

The path matrix $\mathbf{A}^{2}$ for this automaton is given by

$$\mathbf{A}^{2}=\mathbf{A}\otimes\mathbf{A}=\begin{bmatrix}((2t_1)+(t_2\text{-}2))\,\alpha\beta & \epsilon\\ \epsilon & ((t_2\text{-}2)+(2t_3))\,\beta\alpha\end{bmatrix} \tag{100}$$

where $\Theta(A^{2})=\Theta\left(\mathbf{A}\right)$ and $\Phi(A^{2})=\Phi\left(\mathbf{A}\right)$. Note that some elements of $\mathbf{A}^{2}$ contain different lifetimes since every $t_i$ is calculated separately.

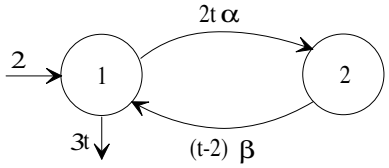


Figure 10: Time-varying automaton for illustrating how to construct the path matrix.

For the proposed TVA formulation the language definitions given in 10 and 11, the definitions for reachability, co-reachability as well as the automata composition operator defined previously for the (max,+) automaton are also valid.

# 4 SUPERVISOR SYNTHESIS

The so called *SupC*(*L*) algorithm (Ramadge e Wonham, 1987a) provides a procedure for synthesizing a supervisor for a DES. The synthesis of a timed (fixed lifetime) supervisor based on this algorithm has already been proposed (Brandin e Wonham, 1994). The supervisor synthesis as proposed in the present article is also based on the *SupC*(*L*) algorithm but allows one to consider variable event lifetime. The following definitions are required in order to state the proposed synthesis procedure:

**Definition 24** Given an alphabet of events $\Sigma$, partitioned into $\Sigma=\Sigma_c\cup\Sigma_{uc}$, and an timed incidence matrix $\mathbf{A}$, we can define its uncontrollable timed incidence matrix $A_{uc}$ by

$$\mathbf{A}_{uc}=\left[\left(a_{uc}\right)_{i,j}\right]; \tag{101}$$

$$\left(a_{uc}\right)_{i,j}=\begin{cases}f(t_{\sigma_{uc}})\sigma_{uc} & \text{if } \exists\sigma_{uc} \text{ from vertex } i \text{ to vertex } j;\\ \epsilon & \text{otherwise,}\end{cases} \tag{102}$$

where $f\left(t_{\sigma_{uc}}\right)$ represents the lifetime of $\sigma_{uc}$. The automaton jumps from state $i$ to state $j$ upon the occurrence of $\sigma_{uc}$. If more than one string may provoke the change from state $i$ to state $j$ then the respective element of the matrix must be written as a time regular expression like $\left(a_{uc}\right)_{i,j}=\bigoplus_{l}f(t_{\sigma^{l}})\sigma^{l}$. The initial state and the marked state vectors are given by $\Theta(\mathbf{A}_{uc})=\Theta(\mathbf{A})$ and $\Phi(\mathbf{A}_{uc})=\Phi(\mathbf{A})$.

The desired behavior for the closed-loop DES, usually denoted by specification, is an essential input information for any supervisor synthesis procedure.

**Definition 25** The specification for a TVA is denoted by a matrix $\mathbf{E}$ defined by

$$\mathbf{E}=[e_{i,j}];\ e_{i,j}=\begin{cases} f(t_\sigma)\sigma & \text{if } \exists\sigma \text{ from vertex } i \text{ to vertex } j; \\ \epsilon & \text{otherwise,} \end{cases} \tag{103}$$

where $f(t_\sigma)$ represents the lifetime of $\sigma$. The automaton jumps from state $i$ to state $j$ upon the occurrence of $\sigma$. If more than one string may provoke the change from state $i$ to state $j$ then the respective element of the matrix must be written as a time regular expression like $e_{i,j} = \bigoplus_l f(t_{\sigma^l})\sigma^l$. For mathematical convenience, the state 1 is always considered as the initial state and can be represented by the row vector $\Theta(\mathbf{E}) = [\ t_0 \quad \epsilon \quad ... \quad \epsilon\ ]$, the marked states are also represented by the following column vector $\Phi(\mathbf{E}) = [\ f(t_{f_1}) \quad f(t_{f_2}) \quad ... \quad f(t_{f_n})\ ]^T$. If the state $k$ is not a marked state, then $f(t_{f_k}) = \epsilon$.

**Definition 26** A supervisor synthesized for a given specification $\mathbf{E}$ is also defined as a matrix

$$\mathbf{S}=[st_{i,j}];\ st_{i,j}=\begin{cases} f(t_\sigma)\sigma\subset e_{i,j} & \text{if } f(t_\sigma)\sigma\subset e_{i,j} \text{ is allowed in } \mathbf{A}; \\ \epsilon & \text{otherwise} \end{cases} \tag{104}$$

where $st_{i,j} = \epsilon$ for $a_{i,j} \neq \epsilon$, indicates that either it is possible to prevent the occurrence the event enabled at $a_{i,j}$, $a_{i,j} \in \Sigma_c$ or that the state $j$ (row j) its not reachable. If more than one string is allowed at $a_{i,j}$ then $st_{i,j} = \bigoplus_l f(t_{\sigma^l})\sigma^l$. For mathematical convenience, the state 1 is always considered as the initial state and can be represented by the row vector $\Theta(\mathbf{S}) = [\ t_0 \quad \epsilon \quad ... \quad \epsilon\ ]$ the marked states are also represented by the column vector $\Phi(\mathbf{S}) = [\ f(t_{f_1}) \quad f(t_{f_2}) \quad ... \quad f(t_{f_n})\ ]^T$. If the state $k$ is not a marked state, then $f(t_{f_k}) = \epsilon$.

The control action of the timed supervisor is executed as follows: the supervisor observes the events and its respective lifetimes and determines the control input for the DES by delaying the occurrence of the controllable events as illustrated in Figure 11.

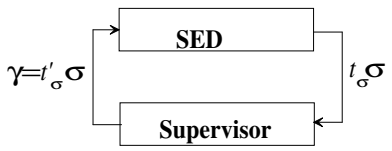


Figure 11: Supervision of a timed DES.

The set of control inputs for the timed supervisor is defined by:

**Definition 27** The set of control inputs for a timed supervisor is defined by

$$\Gamma = \{\gamma_1, \gamma_2, ...\}, \tag{105}$$

where

$$\gamma_i = f(t'_{\sigma_k})\sigma_k, \forall\sigma_k \in \Sigma(i), k = 1, 2, ..., \tag{106}$$

with

$$\begin{array}{l} f(t'_{\sigma_k}) = f(t_{\sigma_k}), \forall\sigma_k \in \Sigma_{uc} \\ (f(t'_{\sigma_k}) \geq f(t_{\sigma_k})), \forall\sigma_k \in \Sigma_c. \end{array} \tag{107}$$

The meaning of $f(t'_{\sigma_k}) \geq f(t_{\sigma_k})$ is that $\sigma_k$ will inhibited by $t = f(t'_{\sigma_k}) - f(t_{\sigma_k})$ units of time, i.e., the lifetime is increased.

According to the definition 27, the control input provided by the timed supervisor is $\gamma = f(t'_\sigma)\sigma$ that defines what are the enabled events in a given state as well as its respective lifetimes ($f(t'_\sigma) \geq f(t_\sigma)$). In other words, the supervisor may inhibit a given event at a given state by $f(t'_\sigma) - f(t_\sigma)$ units of time or block completely its occurrence if $f(t'_\sigma) = \epsilon$. The dynamics of the supervised time system can be studied either by constructing the recognized language through formal series or by using the synchronous composition of $\mathbf{S}||\mathbf{A}$ as obtained according to Definition 11.

In order to make a formal binding of the proposed approach with the standard supervisory control theory it is important to consider the following remarks:

1. A sub-automaton has a similar structure but contains a sub-set of the states or a sub-set of the arcs of a given automaton. The lifetimes associated to the arcs of the sub-automaton must be greater than or equal to the respective lifetimes of a given automaton;
2. A timed sub-language presents a sub-set of the strings of a given timed language. The lifetimes associated to the strings of the sub-language must be greater than or equal to the respective lifetimes of the strings of the language. Given that $L' \subset L$ this does not necessarily imply that the automaton built for recognizing $L'$ is a sub-automaton of the automaton built for recognizing $L$;
3. A sub-matrix is formed by the first $m$ rows and first $m$ columns of a given incidence matrix. The sub-matrix has an elementwise relationship with the matrix but considering that the lifetimes of its elements are greater or equal to the respective lifetimes of the elements of the matrix.

In order to synthesize the timed supervisor as proposed in the present paper it is required that the incidence matrix representing the specification be a sub-matrix of the incidence matrix representing the DES; i.e., $\mathbf{E}$ must be a sub-matrix of $\mathbf{A}$. If this is not the case we must transform $\mathbf{A}$ into $\mathbf{A}^\#$ and build $\mathbf{E}^\#$ to be a sub-matrix of $\mathbf{A}^\#$ such that

$$\begin{array}{rcl} L(\mathbf{A}^\#) & = & L(\mathbf{A}) \\ L(\mathbf{E}^\#) & = & L(\mathbf{E}). \end{array}$$

By doing this, whenever $\mathbf{E}$ is a sub-matrix of $\mathbf{A}$, or due to the suggested transformation $\mathbf{E}^\#$ is a sub-matrix of $\mathbf{A}^\#$, the following order relationship

$$L(\mathbf{E}) \subseteq L(\mathbf{A}) \tag{108}$$

is always satisfied.

The transformation to be applied when $\mathbf{E}$ is not a sub-matrix of $\mathbf{A}$, is described in the following procedure:

1. Build a synchronous composition of $\mathbf{A}$ with a matrix $\mathbf{E}^*$ that generates $\Sigma^*$ to create $\mathbf{A}^\#$ such that

$$L(\mathbf{A}^\#) = L(\mathbf{A}). \tag{109}$$

The matrix $\mathbf{E}^*$ is derived from $\mathbf{E}$ by adding to it a forbidden state (row $i_e$/column $j_e$), named error state. Then the matrix $\mathbf{A}^\#$ is obtained by

$$\mathbf{A}^\# = \mathbf{A}||\mathbf{E}^*; \tag{110}$$

2. The matrix $\mathbf{E}^\#$ is obtained from $\mathbf{A}^\#$ by making all the elements of the form $(i_\mathbf{A}, i_e) = \epsilon$ and $(j_\mathbf{A}, j_e) = \epsilon$. By doing this, the matrix $\mathbf{E}^\#$ is such that

$$L(\mathbf{E}^\#) \subset L(\mathbf{A}^\#) = L(\mathbf{A}), \tag{111}$$

with

$$L(\mathbf{E}) = L(\mathbf{E}^\#) \tag{112}$$

and such that $\mathbf{E}^\#$ is a sub-matrix of $\mathbf{A}^\#$;

3. The lifetimes for $\mathbf{E}$ are used to determine the lifetimes for $\mathbf{E}^\#$ (that are equal to the lifetimes of $\mathbf{A}$) by computing $t_\sigma \oplus t_\sigma^{'}$, where $t_\sigma$ is the lifetime of $\sigma \subset e_{i,j}$ and $t_\sigma^{'}$ is the lifetime of $e^\#_{k,l}$, for $k$ representing the pair $(i_\mathbf{A}, i_\mathbf{E})$ for $i_\mathbf{E} = i$ and $l$ representing the pair $(j_\mathbf{A}, j_\mathbf{E})$ for $j_\mathbf{E} = j$;
4. The vector $\Theta(\mathbf{E}^\#)$ is obtained from $\Theta(\mathbf{E})$ by

$$\theta_1(\mathbf{E}^\#) = \theta_1(\mathbf{E}^\#) \circledast \theta_1(\mathbf{E}); \tag{113}$$

5. The vector $\Phi(\mathbf{E}^\#)$ is also determined from $\Phi(\mathbf{E})$ by

$$\phi_k(\mathbf{E}^\#) = \phi_k(\mathbf{E}^\#) \circledast \phi_i(\mathbf{E}) \tag{114}$$

where $k$ represents the pair $(i_\mathbf{A}, i_\mathbf{E})$ with $i = i_\mathbf{E}$ and $\phi_i(\mathbf{E}) \neq \epsilon$.

Following these steps, $\mathbf{E}^\#$ will be a sub-matrix of $\mathbf{A}^\#$ and consequently

$$\begin{aligned} L(\mathbf{E}^\#) &\subset L(\mathbf{A}^\#) = L(\mathbf{A}) \\ L(\mathbf{E}^\#) &= L(\mathbf{E}). \end{aligned}$$

**Remark 3** The lifetimes for $\mathbf{E}^*$ must be $t_\sigma = e$ such that $L(\mathbf{A}) = L(\mathbf{A}||\mathbf{E}^*)$.

The transformation of $\mathbf{E}$ into $\mathbf{E}^*$ and the transformation of $\mathbf{E}$ into $\mathbf{E}^\#$, respectively can be achieved with following algorithms:

**Algorithm 2** Algorithm for transforming $\mathbf{E}$ into $\mathbf{E}^*$

1. $\forall i$, do $\phi_i(\mathbf{E}) \leftarrow e$, and

$$\forall e_{i,j} = f(t_{\sigma^1})\sigma^1 + ... + f(t_{\sigma^n})\sigma^n, \tag{115}$$

do

$$f(t_{\sigma^k}) \leftarrow e,\ k = 1, ..., \dim(\mathbf{E}); \tag{116}$$

2. for each row $i$ of $\mathbf{E}$, include the self-loops $e_{i,i} = \Sigma - \{\sigma^1, \sigma^2, ..., \sigma^n\}$, such that $\sigma^k \not\subset e_{i,j}, k = 1, 2, ..., \dim(\mathbf{E})$;
3. make $\dim(\mathbf{E}) \leftarrow \dim(\mathbf{E}) + 1$, where $(i, \dim(\mathbf{E}))$ and $(\dim(\mathbf{E}), j)$ are the elements of the error row/column $(i_e/j_e)$;
4. $\theta_{\dim(\mathbf{E})}(\mathbf{E}) \leftarrow \epsilon$;
5. $\phi_{\dim(\mathbf{E})}(\mathbf{E}) \leftarrow e$;
6. for $i = 0$ until $i < i_e$, do:

**a)** for $j = 0$ until $j < j_e$, do:

**i.** for $k = 0$ until $k < j_e$, do:

**I)** if $\sigma \subset e_{j,i} \vee \sigma \subset e_{j,j}$ and $e_{i,k} \neq \sigma$ do $e_{i,j_e} \leftarrow \sigma$;

**ii.** if $e_{i,j} \neq \sigma^{'}$ do $e_{i,j_e} \leftarrow \sigma^{'}$;

7. $e_{i_e,j_e} \leftarrow \Sigma$.

**Algorithm 3** Algorithm for transforming $\mathbf{E}$ into $\mathbf{E}^\#$

1. Transform $\mathbf{E}$ in $\mathbf{E}^*$;
2. Construct $\mathbf{A}^\# = \mathbf{A}||\mathbf{E}^*$;
3. for $i = 1$ until $n \times m$ do:

**i.** for $j = 1$ until $n \times m$, do:

**a)** if $a^\#_{i,j} \neq \epsilon$, do $e^\#_{i,j} \leftarrow a^\#_{i,j}$.

4. for $k = 1$ until $n \times m$:

**a)** if $\phi_k\left(\mathbf{A}^\#\right) \neq \epsilon$, do

$$i_\mathbf{E} \leftarrow ((k - (k \bmod m))/m) + 1, \tag{117}$$

and

$$\phi_k\left(\mathbf{E}^\#\right) \leftarrow \phi_k\left(\mathbf{E}^\#\right) \circledast \phi_{i_\mathbf{E}}(\mathbf{E}); \tag{118}$$

5. do $\theta_1\left(\mathbf{E}^\#\right) \leftarrow \theta_1\left(\mathbf{E}^\#\right) \circledast \theta_1(\mathbf{E})$.

In this algorithm mod is the operator that determines the remainder of the division $k/m$. The step 4 of this algorithm define the final delays and step 5 defines the initial delay in order to make sure that $L\left(\mathbf{E}^\#\right) = L(\mathbf{E})$ and $L_m\left(\mathbf{E}^\#\right) = L_m(\mathbf{E})$.

The use of these algorithms will be illustrated by an example where the transition functions of the TVA are constants, like in a (max,+) automaton.

**Example 20** Consider the matrix

$$\mathbf{A} = \begin{bmatrix} \epsilon & 2\alpha_1 & 3\alpha_2 & \epsilon \\ 2\beta_1 & \epsilon & \epsilon & 3\alpha_2 \\ 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 \\ \epsilon & 4\beta_2 & 2\beta_1 & \epsilon \end{bmatrix}, \ \Theta(\mathbf{A}) = \begin{bmatrix} 3 & \epsilon & \epsilon & \epsilon \end{bmatrix}, \ \Phi(\mathbf{A}) = \begin{bmatrix} e & \epsilon & \epsilon & 3 \end{bmatrix}^T$$

and the desired specification

$$\mathbf{E} = \begin{bmatrix} \alpha_1 + \beta_2 & 5\beta_1 \\ 4\alpha_2 & \alpha_1 + \beta_2 \end{bmatrix}, \ \Theta(\mathbf{E}) = \begin{bmatrix} 4 & \epsilon \end{bmatrix}, \ \Phi(\mathbf{E}) = \begin{bmatrix} 2 & \epsilon \end{bmatrix}^T$$

where $\Sigma_{uc} = \{\beta_1, \beta_2\}$. To build $E^{\#}$ such that $L\left(\mathbf{E}^{\#}\right) \subset L\left(\mathbf{A}^{\#}\right) = L\left(\mathbf{A}\right)$ we first create $E^{*}$ that generates $\Sigma^{*}$. In this case

$$\mathbf{E}^{*} = \begin{bmatrix} \alpha_1+\beta_2 & \beta_1 & \alpha_2 \\ \alpha_2 & \alpha_1+\beta_2 & \beta_1 \\ \epsilon & \epsilon & \alpha_1+\beta_1+\alpha_2+\beta_2 \end{bmatrix},$$
$$\Theta\left(\mathbf{E}\right) = \begin{bmatrix} e & \epsilon & \epsilon \end{bmatrix}, \Phi\left(\mathbf{E}\right) = \begin{bmatrix} e & e & e \end{bmatrix}^{T}$$

where the error row/column is the third one. Now computing $\mathbf{A}||\mathbf{E}^{*}$ we obtain

$$\mathbf{A}^{\#}=\begin{bmatrix} \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 3\alpha_2 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 3\alpha_2 \\ 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & 3\alpha_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 3\alpha_2 & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 2\alpha_1 & 3\alpha_2 & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & 3\alpha_2 \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & 2\alpha_1 \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & 2\beta_1 & \epsilon \end{bmatrix}$$
$$\Theta(\mathbf{A}^{\#}) = \begin{bmatrix} 3 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{A}^{\#}) = \begin{bmatrix} e & \epsilon & \epsilon & 3 & e & \epsilon & \epsilon & 3 & e & \epsilon & \epsilon & 3 \end{bmatrix}^{T}$$

and then we get $\mathbf{E}^{\#}$ as given by

$$\mathbf{E}^{\#}=\begin{bmatrix} \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}$$
$$\Theta(\mathbf{E}^{\#}) = \begin{bmatrix} 4 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{E}^{\#}) = \begin{bmatrix} 2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}^{T}$$

Note that only $\phi_1(\mathbf{E}^{\#}) \neq \epsilon$ since $\phi_1(\mathbf{E}^{\#}) = \phi_1\left(\mathbf{A}\right) \circledast \phi_1\left(\mathbf{E}\right)$.

In order to synthesize the timed supervisor $\mathbf{S}$ it is required to introduce the following operators:

**Definition 28** The $ACES$ operator is defined as

$$ACES(\mathbf{A}) = \mathbf{B}, b_{i,j} = \begin{cases} a_{i,j} & \text{if } i \text{ is reachable} \\ \epsilon & \text{otherwise.} \end{cases} \quad (119)$$

The operation $ACES(\mathbf{A})$ eliminates all the elements $a_{i,j}$ of a non-reachable $i$ row. For a given matrix $\mathbf{F}$, the $ACES$ operator can be implemented by the following algorithm:

**Algorithm 4** $ACES$ Operator

1. Create a vector $vac_{1\times N}$ of reachable states.
2. for $i = 1$ until $N$, do:
   **a)** for $j = 1$ until $N$, do:
      **i.** if $\mathbf{F}(1,j) \neq \epsilon$, do $vac(j) \leftarrow 1$.
      **ii.** if $i > 1$ and $\mathbf{F}(i,j) \neq \epsilon$ and $vac(i) = 1$, then do $vac(j) \leftarrow 1$.
3. for $i = N-1$ until 1, do:
   **b)** for $j = 1$ until $N$, do:
      **i.** if $\mathbf{F}(i,j) \neq \epsilon$ and $vac(i) = 1$, then do $vac(j) \leftarrow 1$.
4. for $i = 1$ until $N$, do:
   **a)** for $j = 1$ until $N$, do:
      **i.** if $vac(i) = 0$, do $\mathbf{F}\left(i,j\right) \leftarrow \epsilon$.

**Definition 29** The $COACES$ operator is defined as

$$COACES(\mathbf{A}) = \mathbf{B},$$
$$b_{i,j}=\begin{cases} a_{i,j} & \text{if } \exists s | s = a_{i,j_1}a_{j_1,j_2}...a_{j_{n-1},j_n}, \\ & a_{i,j_1}, a_{j_1,j_2}, .., a_{j_{n-1},j_n} \neq \epsilon \text{ and } \phi_{j_n}(\mathbf{A}) = e; \\ \epsilon & \text{otherwise} \end{cases} \quad (120)$$

The $COACES\left(\mathbf{A}\right)$ operation eliminates the elements that lead to non-reachable rows. For a given matrix $\mathbf{F}$, the $COACES$ operator can be implemented by the following algorithm:

**Algorithm 5** $COACES$ Operator

1. Create a vector $vco_{N\times 1}$ of coreachable states.
2. for $i = 1$ until $N$, do:
   **a)** for $j = 1$ until $N$, do:
      **i.** if $\mathbf{F}(1,j) \neq \epsilon$ and $\phi_j\left(\mathbf{F}\right) = e$, do $vco(i) \leftarrow 1$.
      **ii.** if $i > 1$ and $\mathbf{F}(i,j) \neq \epsilon$ and $(\phi_j\left(\mathbf{F}\right) = e$ or $vco(j) = 1)$, then do $vco(i) \leftarrow 1$.
3. for $j = 1$ until $N$, do:
   **b)** for $i = N-1$ until 1, do:
      **i.** if $\mathbf{F}(i,j) \neq \epsilon$ and $(\phi_j\left(\mathbf{F}\right) = e$ or $vco(j) = 1)$, then do $vco(i) \leftarrow 1$.
4. For $j = 1$ until $N$, do:
   **a)** for $i = 1$ until $N$, do:
      **i.** if $vco(j) = 0$, do $\mathbf{F}\left(i,j\right) \leftarrow \epsilon$.

**Definition 30** The operator $TRIM$ is defined by

$$TRIM\left(\mathbf{A}\right) = ACES\left(COACES(\mathbf{A})\right) = B. \quad (121)$$

The use of the $TRIM$ operator for a given incidence $\mathbf{A}$ yields an incidence matrix $\mathbf{B}$ that is both reachable and co-reachable.

The last but not least operator to be introduced allows to compare two matrices to determine if the language generated by one is contained in the language generated by the other.

**Definition 31** Given two timed incidence matrices $\mathbf{A} = [a_{i,j}]$ and $\mathbf{B} = [b_{i,j}]$, the operator $\trianglelefteq$ is defined by

$$\mathbf{A} \trianglelefteq \mathbf{B} \Rightarrow L(\mathbf{A}) \subseteq L(\mathbf{B}). \tag{122}$$

Similarly, the operators $\trianglerighteq$, $\triangleleft$ and $\triangleright$ can be defined by

$$\begin{array}{l} \mathbf{A} \trianglerighteq \mathbf{B} \Rightarrow L(\mathbf{A}) \supseteq L(\mathbf{B}), \\ \mathbf{A} \triangleleft \mathbf{B} \Rightarrow L(\mathbf{A}) \subset L(\mathbf{B}), \\ \mathbf{A} \triangleright \mathbf{B} \Rightarrow L(\mathbf{A}) \supset L(\mathbf{B}). \end{array} \tag{123}$$

Based on definition 31, and considering a generic element $\sigma = f(t_{\sigma^1})\sigma^1 + ... + f(t_{\sigma^n})\sigma^n$, the determination of $\mathbf{A} \trianglelefteq \mathbf{B}$ can be achieved through the following algorithm:

**Algorithm 6** Algorithm for determining $\mathbf{A} \trianglelefteq \mathbf{B}$

1. for $i = 1$ until $N$, do:elemento $a_{i,j}$ faça:
   - **i.** for $j = 1$ until $N$, do:
     - **a)** if $\sigma \notin \mathbf{B}$ and $a_{i,j} = \sigma$, then $\mathbf{A} \not\trianglelefteq \mathbf{B}$;
     - **b)** if $\left(a_{i,j} = t'_{\sigma^h}\sigma^h + ... + t'_{\sigma^k}\sigma^k, t'_{\sigma^i} \geq t_{\sigma^i}\right) \vee (a_{i,j} = \epsilon)$, where $\sigma^i, ..., \sigma^k \subset \sigma$ and $b_{i,j} = t_\sigma \sigma$, then $\mathbf{A} \trianglelefteq \mathbf{B}$;
2. if $\theta_1(\mathbf{A}) \geq \theta_1(\mathbf{B})$, then $\mathbf{A} \trianglelefteq \mathbf{B}$.
3. for $i = 1$ until $N$, do:
   - **i.** if $(\phi_i(\mathbf{A}) \geq \phi_i(\mathbf{B}))$ or $(\phi_i(\mathbf{A}) = \epsilon \wedge \phi_i(\mathbf{B}) \neq \epsilon)$ or $(\phi_i(\mathbf{A}) = \epsilon \wedge \forall \phi_i(\mathbf{B}) = \epsilon)$, then $\mathbf{A} \trianglelefteq \mathbf{B}$.

The condition $\mathbf{A} \trianglelefteq \mathbf{B}$ implies the execution time of every sequence in $\mathbf{A}$ (string of events with its respective lifetimes) must be greater than or equal to the execution time of the same sequence in $\mathbf{B}$. In other words, given two timed languages $L(\mathbf{A})$ and $L(\mathbf{B})$ this condition also implies that $L(\mathbf{A}) \subseteq L(\mathbf{B})$, i.e., the execution time of a given timed language is always less than or equal to the execution time of any sub-language of that language (Alur e Dill, 1990; Alur e Dill, 1994; Alur, 1997). Equivalently,

$$(y|s_{\mathbf{A}}) \geq (y|s_{\mathbf{B}}), \tag{124}$$

where the lifetime of every event satisfies

$$t_{\sigma_{i_{\mathbf{A}}}} \geq t_{\sigma_{i_{\mathbf{B}}}}. \tag{125}$$

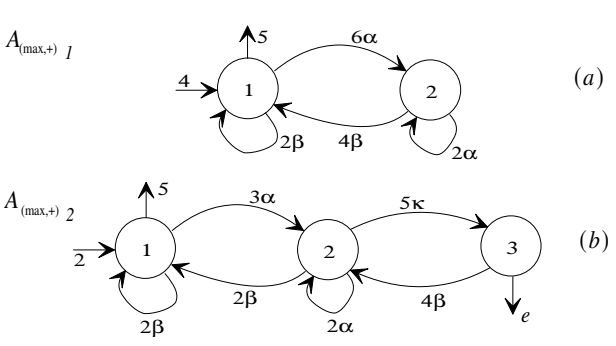


Figure 12: (max,+) automata for illustrating the use of the $\triangleleft$ operator.

**Example 21** The automaton shown in Figure 12(a) has the following matrix representation

$$\mathbf{A}_1 = \begin{bmatrix} 2\beta & 6\alpha & \epsilon \\ 4\beta & 2\alpha & \epsilon \\ \epsilon & \epsilon & \epsilon \end{bmatrix}, \quad \begin{array}{l} \Theta(\mathbf{A_1}) = \begin{bmatrix} 4 & \epsilon & \epsilon \end{bmatrix}, \\ \Phi(\mathbf{A_1}) = \begin{bmatrix} 5 & \epsilon & \epsilon \end{bmatrix}^T \end{array}$$

and the automaton shown in Figure 12(b) has the following matrix representation

$$\mathbf{A}_2 = \begin{bmatrix} 2\beta & 3\alpha & \epsilon \\ 2\beta & 2\alpha & 5\kappa \\ \epsilon & 4\beta & \epsilon \end{bmatrix}, \quad \begin{array}{l} \Theta(\mathbf{A}_2) = \begin{bmatrix} 2 & \epsilon & \epsilon \end{bmatrix}, \\ \Phi(\mathbf{A_2}) = \begin{bmatrix} 5 & \epsilon & e \end{bmatrix}^T \end{array}$$

Executing the Algorithm 6 we determine that $\mathbf{A}_1 \triangleleft \mathbf{A}_2$.

Besides the use of these operators, the proposed supervisor synthesis procedure requires that the desired specification $\mathbf{E}$ must be valid and controllable.

**Definition 32** A specification $\mathbf{E}$ for a given DES $\mathbf{A}$ is said valid if $\mathbf{E} \neq [\epsilon]$ and

$$\begin{array}{l} \forall e_{i,j}, f(t_\sigma) \geq f(t'_\sigma), f(t'_\sigma) \text{ lifetime of } \sigma \subset a_{i,j} \\ \theta_1(\mathbf{E}) \geq \theta_1(\mathbf{A}) \quad \text{and} \\ \phi_i(\mathbf{E}) \geq \phi_i(\mathbf{A}) \vee \phi_i(\mathbf{E}) = \epsilon, \forall \phi_i(\mathbf{A}) \neq \epsilon, \forall i = 1 \text{ to } N, \end{array} \tag{126}$$

where $[\epsilon]$ is the null matrix that all its elements equal to $\epsilon$ and if $\forall i, j, \sigma \subset e_{i,j}, \sigma \in \Sigma$.

The controllability condition is applied for valid a specification as

**Definition 33** A specification $\mathbf{E}$ for a given DES $\mathbf{A}$ is said controllable if

$$ACES(\mathbf{E}) = \mathbf{E}, \tag{127}$$

and

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) = \mathbf{E}. \tag{128}$$

The last condition can be employed to determine the existence of supervisor for a given DES.

**Lemma 1** *Given* $\mathbf{A}$ *(system) and* $\mathbf{E}$ *(specification) a supervisor* $\mathbf{S}$ *is defined if and only if*

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) = \mathbf{E}. \tag{129}$$

If the specification $\mathbf{E}$ satisfies the controllability condition then the supervisor $\mathbf{S}$ is determined by taking its trim component.

**Corollary 2** *Given* $\mathbf{A}$ *(system) and* $\mathbf{E}$ *(specification) the supervisor* $\mathbf{S}$ *is given by*

$$\mathbf{S} = TRIM(\mathbf{E}) \tag{130}$$

*if and only if*

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) = \mathbf{E}. \tag{131}$$

**Example 22** The automaton shown in Figure 13(a) has the following matrix representation

$$\mathbf{A} = \begin{bmatrix} \epsilon & (t^2\text{-}2t)\beta & 3t\mu \\ (t\text{-}2)\alpha & \epsilon & 4\mu \\ \epsilon & 3\mu & (t\text{-}1)\kappa \end{bmatrix}, \quad \Theta(\mathbf{A}) = \begin{bmatrix} 2 & \epsilon & \epsilon \end{bmatrix}, \quad \Phi(\mathbf{A}) = \begin{bmatrix} 3t & \epsilon & 1 \end{bmatrix}^T$$

where $\Sigma = \{\alpha, \beta, \kappa, \mu\}$, $\Sigma_{uc} = \{\kappa\}$ and $\Sigma_c = \{\alpha, \beta, \mu\}$. Considering that the specification given by

$$\mathbf{E} = \begin{bmatrix} \epsilon & (t^2\text{-}2t)\beta & (3t+1)\mu \\ (t^2\text{-}2)\alpha & \epsilon & 4\mu \\ \epsilon & 3\mu & (t\text{-}1)\kappa \end{bmatrix}, \quad \Theta(\mathbf{E}) = \begin{bmatrix} 4 & \epsilon & \epsilon \end{bmatrix}, \quad \Phi(\mathbf{E}) = \begin{bmatrix} 5t & \epsilon & 2 \end{bmatrix}^T$$

corresponding to the automaton shown in Figure 14(b), we have

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) = \mathbf{E},$$

where $\theta_1(\mathbf{E}) > \theta_1(\mathbf{A})$ ($\theta_2$ and $\theta_3$ are equal), $\phi_1(\mathbf{E}) = 5t$, $\phi_3(\mathbf{E}) = 2$ provided that $\mathbf{E}$ is controllable. Since $\mathbf{E}$ is also trim, the supervisor is given by $\mathbf{S} = TRIM(\mathbf{E}) = \mathbf{E}$.

For a given specification when $ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) \rhd \mathbf{E}$, it is required to determine the supremal controllable sub-language in order to synthesize the supervisor. This case will be discussed in the following:

**Definition 34** For a given $\mathbf{E}$ we can form the matrix

$$\mathbf{B}_{uc}^n = \mathbf{E} \otimes (\mathbf{A}_{uc})^{n-1}, \tag{132}$$

that is denoted the path matrix of order $n$. In this matrix the first element of every string is an element of $\mathbf{E}$ and all the others belong to $\mathbf{A}_{uc}$. The marked states vector is defined by

$$\phi_i(\mathbf{B}_{uc}^n) = \phi_i(\mathbf{E}) \circledast \phi_i(\mathbf{A}) \tag{133}$$

and the initial state vector by

$$\Theta(\mathbf{B}_{uc}^n) = \Theta(\mathbf{A}) \circledast \Theta(\mathbf{E}). \tag{134}$$

In a path matrix a given event $\sigma_{uc}$ belonging to $(a_{uc})_{i,j}$ always appears as the last event in the elements of $(b_{uc}^n)_{k,j}$. These terms of $\mathbf{B}_{uc}^n$ are preceded by $(b_{uc}^{n-1})_{k,i}$. Considering this we have that:

**Theorem 3** *If for a valid* $\mathbf{E}$ *and a given* $\mathbf{A}_{uc}$

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) \rhd \mathbf{E}, \tag{135}$$

*then the supremal controllable sublanguage* $\sup C(L)$ *will be determined recursively by:*

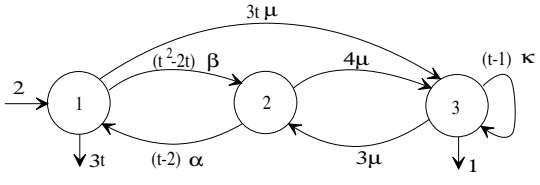


Figure 13: Automaton for illustrating the supervisor synthesis procedure.

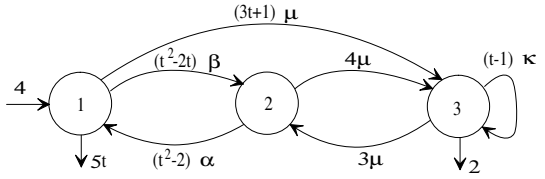


Figure 14: Controllable behavior specification.

1. *For* $n = 1$, $\mathbf{S}^1 = \mathbf{E}$.
2. *For* $n = n + 1$ *while* $(n \leq N) \wedge \exists \sigma_{uc} \notin \mathbf{E}$ *do*

$$\mathbf{B}_{uc}^n = \mathbf{E} \otimes (\mathbf{A}_{uc})^{n-1}$$
$$\mathbf{S}^n = \left[ s_{i,j}^n \right], \; s_{i,j}^n = \begin{cases} s_{i,j}^{n-1} & \text{if } \sigma^1 \sigma_{uc}^2 ... \sigma_{uc}^n \in \mathbf{B}_{uc}^n \wedge \sigma_{uc}^n \in \mathbf{E}; \\ \epsilon & \text{if } \sigma_{uc}^n \notin \mathbf{E} \wedge \sigma^1 \in \Sigma_c \end{cases}$$
$$\mathbf{S}^n = TRIM(\mathbf{S}^n)$$

   *The term* $\sigma_{uc}^n$ *is* $n$*-th event of a string of an element of* $\mathbf{B}_{uc}^n$ *that can be* $\sigma_{uc} \notin \mathbf{E}$.
3. *If* $(n > N) \wedge (\exists \sigma_{uc} \notin E$ *in* $\mathbf{S}^n)$ *then* $\mathbf{S} = [\epsilon]$.

## 5 SYNTHESIS ALGORITHM

Based on the previous formalism we can state the algorithm to synthesize the supervisor:

**Algorithm 7** Supervisor synthesis algorithm

1. if $\mathbf{E} \not\lhd \mathbf{A}$, construct $\mathbf{A}^{\#}$ and $\mathbf{E}^{\#}$ starting of $\mathbf{A}$ and $\mathbf{E}$, and do $\mathbf{E} \leftarrow \mathbf{E}^{\#}$ and $\mathbf{A} \leftarrow \mathbf{A}^{\#}$.
2. do $\mathbf{D} \leftarrow ACES(\mathbf{E} \oplus \mathbf{A}_{uc})$.
3. if $\mathbf{D} = \mathbf{E}$, do $\mathbf{S} \leftarrow TRIM(\mathbf{E})$ and stop.
4. if $\mathbf{D} \rhd \mathbf{E}$, do $n \leftarrow 1$:

   **a)** for $k = 1$ until $M$ ($M$ is the number of distinct elements between $\mathbf{A}_{uc}$ and $\mathbf{E}_{uc}$), do:

      **i.** $\mathbf{S}^n \leftarrow \mathbf{E}$, $xdif(k, n-1) \leftarrow i$ and $ydif(k, n-1) \leftarrow j$ (where $\mathbf{A}_{uc} \neq \mathbf{E}_{uc}$);

   **b)** compute $\mathbf{B}_{uc}^n$;

   **c)** for $k = 1$ until $M$, do:

      **i.** search the elements in

$$\mathbf{B}_{uc}^n(i, ydif(k, n-1)), \tag{136}$$

      where $\sigma_{uc}^n \notin \mathbf{E}$ ($\sigma_{uc}^n$ being the last element of the sequence).

      **(1)** if $\sigma_{uc}^n \notin \mathbf{E}$ and $\sigma^1 \in \Sigma_c$, do

$$\mathbf{S}^n(i, xdif(k, n-1)) \leftarrow \epsilon \tag{137}$$

      and compute

$$COACES(\mathbf{S}^n) \text{ and } ACES(\mathbf{S}^n); \tag{138}$$

**(2)** if $\sigma_{uc}^n \notin \mathbf{E}$ and $\sigma^1 \in \Sigma_{uc}$, do

$$xdif(k,n) \leftarrow i \tag{139}$$

and

$$ydif(k,n) \leftarrow ydif(k,n-1); \tag{140}$$

**d)** for $k = 1$ until $M$, do:

**i.** if

$$\mathbf{S}^n(xdif(k,n-1), ydif(k,n-1)) = \epsilon \tag{141}$$

$\forall k$, stop.

**ii.** else, do:

**(1)** $n \leftarrow n+1$;

**(2)** if $n \leq N$ then return to step $4.b$.

**(3)** else stop ($\mathbf{E}$ is not feasible).

## 6 ILLUSTRATIVE EXAMPLES

In this section we presented selected examples to illustrate the supervisor synthesis procedure as proposed in the present paper. The examples will focus the supervisor synthesis for un-timed automata, (max,+) automata and for TVA automata.

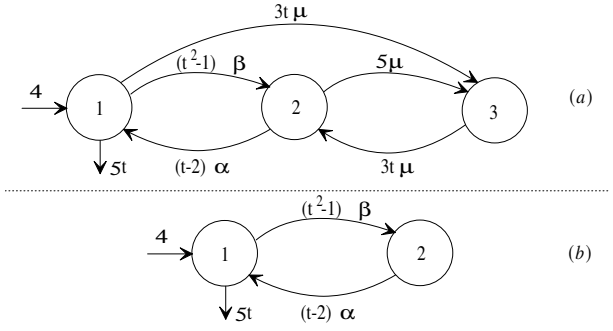


Figure 15: Time-varying automata: (a) Uncontrollable specification and (b) Supervisor.

**Example 23** Consider that for the TVA shown in Figure 13 the specification is given by

$$\mathbf{E} = \begin{bmatrix} \epsilon & (t^2\text{-}1)\beta & 3t\mu \\ (t\text{-}2)\alpha & \epsilon & 5\mu \\ \epsilon & 3t\mu & \epsilon \end{bmatrix}, \Theta(\mathbf{E}) = \begin{bmatrix} 4 & \epsilon & \epsilon \end{bmatrix}, \Phi(\mathbf{E}) = \begin{bmatrix} 5t \\ \epsilon \\ \epsilon \end{bmatrix}$$

The automaton for this specification is shown in Figure 15(a). The controllability test shows that

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) \rhd \mathbf{E}.$$

and then the synthesis procedure employing the path matrix must be employed. The first step is

$$\mathbf{S}^1 = \mathbf{E}.$$

Next we compute $\mathbf{B}_{uc}^2$

$$\mathbf{B}_{uc}^2 = \mathbf{E} \otimes \mathbf{A}_{uc} = \begin{bmatrix} \epsilon & \epsilon & \mu\kappa \\ \epsilon & \epsilon & \mu\kappa \\ \epsilon & \epsilon & \epsilon \end{bmatrix}$$

from where we can identify the terms

$$(b_{uc}^2)_{1,3} = \mu\kappa$$

and

$$(b_{uc}^2)_{2,3} = \mu\kappa,$$

that contain a controllable $\mu$ event followed by an uncontrollable $\kappa$ event that does not belong to $\mathbf{E}$. Since $\mu$ is controllable, we can set

$$s_{1,3}^2 = s_{2,3}^2 = \epsilon.$$

to avoid this sequence and, as a result the state 3 becomes unreachable, i.e.,

$$s_{3,2}^2 = s_{3,3}^2 = \epsilon.$$

The the supervisor can determined by taking the trim component of $\mathbf{S}^2$ as given by

$$\mathbf{S} = TRIM(\mathbf{S}^2) = \begin{bmatrix} \epsilon & (t^2\text{-}1)\beta & \epsilon \\ (t\text{-}2)\alpha & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon \end{bmatrix}, \quad \begin{matrix} \Theta(\mathbf{S}) = \begin{bmatrix} 4 & \epsilon & \epsilon \end{bmatrix}, \\ \Phi(\mathbf{S}) = \begin{bmatrix} 5t & \epsilon & \epsilon \end{bmatrix}^T \end{matrix}$$

and is shown in Figure 15(b).

**Example 24** Consider the Example 20 that is an extension of classical problem of a system consisting of machines and buffer. Consider that $\Sigma = \{\alpha_1, \alpha_2, \beta_1, \beta_2\}$, $\Sigma_{uc} = \{\beta_1, \beta_2\}$ and the extended specification $\mathbf{E}^{\#}$ shown in Figure 16. The controllability

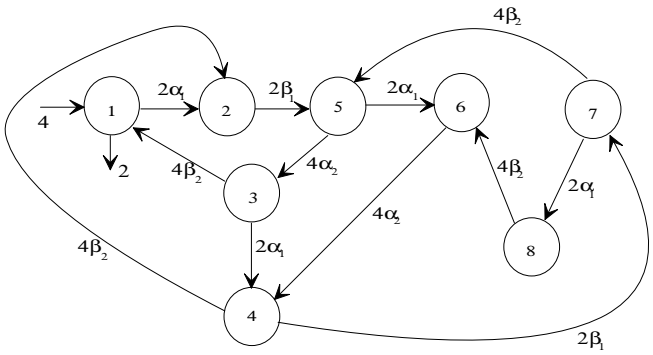


Figure 16: Timed behavior specification for the two machines and a buffer problem.

teste shows that

$$ACES(\mathbf{E}^{\#} \oplus \mathbf{A}^{\#}) \rhd \mathbf{E}^{\#}.$$

Note that $e_{6,9}^{\#}$ and $e_{8,11}^{\#}$ does not belong to $\mathbf{E}^{\#}$ and must not occur. Then, the synthesis procedure employing the path matrix must be employed. The first steps are to make $\mathbf{S}^1 = \mathbf{E}^{\#}$, $\Theta(\mathbf{S}^1) = \Theta(\mathbf{E}^{\#})$, $\Phi(\mathbf{S}^1) = \Phi(\mathbf{E}^{\#})$ and compute $\mathbf{B}_{uc}^2 = \mathbf{E}^{\#} \otimes \mathbf{A}_{uc}^{\#}$. Observing $\mathbf{B}_{uc}^2$ we identify $(b_{uc}^2)_{5,9} = 4\alpha_1\beta_1$, $(b_{uc}^2)_{7,11} = 4\alpha_1\beta_1$ and $(b_{uc}^2)_{8,9} = 6\beta_2\beta_1$ that are terms resulting from $e_{5,6}^{\#} \otimes (a_{uc}^{\#})_{6,9}$, $e_{7,8}^{\#} \otimes (a_{uc}^{\#})_{8,11}$

and $e^{\#}_{8,6} \otimes (a^{\#}_{uc})_{6,9}$, respectively. The inhibition of the controllable events that precedes $\beta_1$ in $s^1_{5,6}$ and $s^1_{7,8}$ make row 8 unreachable and consequently we does not need to examine the term $(b^2_{uc})_{8,9}$. In this case $S^2$ is given by

$$\mathbf{S}^2=\begin{bmatrix} \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon \\ 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon \\ \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 \\ \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & \underline{\epsilon} & \epsilon \\ \epsilon & \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon \end{bmatrix}$$

$$\Theta(\mathbf{S}^2)=\begin{bmatrix} 4 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{S}^2)=\begin{bmatrix} 2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}^T$$

By making $\mathbf{S}^2 = COACES(\mathbf{S}^2)$ followed by $\mathbf{S}^2 = ACES(\mathbf{S}^2)$ we eliminate the not co-reachable and not reachable states, respectively. The results is

$$\mathbf{S}^2=\begin{bmatrix} \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 & \epsilon & \epsilon \\ 4\beta_2 & \epsilon & \epsilon & 2\alpha_1 & \epsilon & \epsilon & \epsilon \\ \epsilon & 4\beta_2 & \epsilon & \epsilon & \epsilon & \epsilon & 2\beta_1 \\ \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 4\alpha_2 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 4\beta_2 & \epsilon & \epsilon \end{bmatrix}$$

$$\Theta(\mathbf{S}^2)=\begin{bmatrix} 4 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{S}^2)=\begin{bmatrix} 2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}^T$$

which is the supervisor shown in Figure 17. This supervisor makes sure that the events will be enabled after a certain delay such that the timed language of the supervised system will satisfy

$$L(\mathbf{S}||\mathbf{A}^{\#}) = L(\mathbf{S}). \tag{142}$$

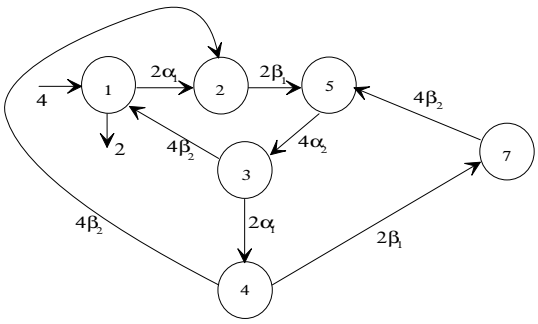


Figure 17: Timed supervisor.

**Example 25** Consider the automaton shown in Figure 18 where $\Sigma = \{\alpha, \beta, \kappa, \eta, \lambda, \mu\}$ and $\Sigma_{uc} = \{\alpha, \lambda\}$. The incidence matrices $\mathbf{A}$ and $\mathbf{A}_{uc}$ are

$$\mathbf{A}=\begin{bmatrix} \epsilon & \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \lambda & \beta & \kappa & \mu & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \kappa+\mu & \epsilon & \epsilon & \eta \\ \epsilon & \epsilon & \epsilon & \alpha & \beta & \epsilon & \epsilon \\ \epsilon & \epsilon & \mu & \lambda & \epsilon & \alpha & \eta \\ \alpha & \mu & \epsilon & \eta & \kappa & \beta+\lambda & \epsilon \\ \epsilon & \mu & \eta+\lambda & \epsilon & \epsilon & \epsilon & \alpha+\beta \end{bmatrix}$$

$$\Theta(\mathbf{A})=\begin{bmatrix} e & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{A})=\begin{bmatrix} \epsilon & e & \epsilon & \epsilon & \epsilon & \epsilon & e \end{bmatrix}^T$$

and

$$\mathbf{A}_{uc}=\begin{bmatrix} \epsilon & \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \lambda & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \alpha & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \lambda & \epsilon & \alpha & \epsilon \\ \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \lambda & \epsilon \\ \epsilon & \epsilon & \lambda & \epsilon & \epsilon & \epsilon & \alpha \end{bmatrix}$$

$$\Theta(\mathbf{A}_{uc})=\Theta(\mathbf{A})$$
$$\Phi(\mathbf{A}_{uc})=\Phi(\mathbf{A})$$

For each marked state there is an output arc valued with $\phi_{q_m} = e$; the initial state is an input arc valued with $\theta_q = e$ and for all the remaining arcs its transition functions are $f_\sigma(t) = e$. The specification is given by

$$\mathbf{E}=\left[\begin{array}{ccccccc|c} \epsilon & \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \lambda & \epsilon & \kappa & \epsilon & \epsilon & \epsilon & \epsilon & e \\ \epsilon & \epsilon & \epsilon & \mu & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \alpha & \beta & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \alpha & \eta & \epsilon \\ \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \lambda & \epsilon & \epsilon \\ \epsilon & \epsilon & \lambda & \epsilon & \epsilon & \epsilon & \alpha & e \end{array}\right]$$

$$\Theta(\mathbf{E})=\begin{bmatrix} e & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix},$$
$$\Phi(\mathbf{E})=\begin{bmatrix} \epsilon & e & \epsilon & \epsilon & \epsilon & \epsilon & e \end{bmatrix}^T$$

and corresponds to the automaton shown in Figure 19. It is ease to see that $\mathbf{E}$ is valid and $\mathbf{E} \lhd \mathbf{A}$, however the controllability test fails since

$$ACES(\mathbf{E} \oplus \mathbf{A}_{uc}) \rhd \mathbf{E}.$$

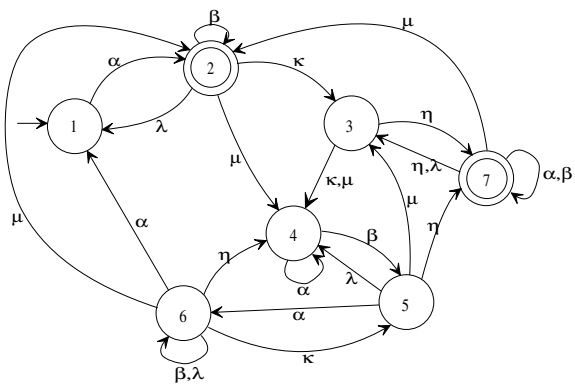


Figure 18: Un-timed automaton.

The term $(a_{uc})_{5,4} = \lambda$ does not belong to $\mathbf{E}$. Then, the synthesis procedure based on the path matrix must be employed, i.e., make $\mathbf{S}^1 = \mathbf{E}$ and compute $\mathbf{B}^2_{uc} = \mathbf{E} \otimes \mathbf{A}_{uc}$

$$\mathbf{B}^2_{uc}= \begin{bmatrix} \alpha\lambda & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \lambda\alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \mu\alpha & \epsilon & \epsilon & \epsilon \\ \epsilon & \eta\alpha & \epsilon & \alpha\alpha + \underline{\beta\lambda} & \epsilon & \beta\alpha & \epsilon \\ \alpha\alpha & \epsilon & \eta\lambda & \epsilon & \epsilon & \alpha\lambda & \eta\alpha \\ \lambda\alpha & \alpha\alpha & \epsilon & \epsilon & \epsilon & \lambda\lambda & \epsilon \\ \epsilon & \epsilon & \alpha\lambda & \epsilon & \epsilon & \epsilon & \alpha\alpha \end{bmatrix}$$

with $\Theta(\mathbf{B}^2_{uc}) = \Theta(\mathbf{A}) = \Theta(\mathbf{E})$, $\Phi(\mathbf{B}^2_{uc}) = \Phi(\mathbf{A}) = \Phi(\mathbf{E})$, to determine what events must be disabled to generate the language restricted by the supervisor. In this case we are interested in disabling the event that has made the controllability test to fail. Observing $\mathbf{B}^2_{uc}$ we see that $(b^2_{uc})_{4,4}$ has the sequence $\beta\lambda$, that leads the automaton, through the controllable event $\beta$, from state $4$ to state $5$, and then back to state $4$ through the uncontrollable event $\lambda$ that does belong to $\mathbf{E}$. Since $\beta$ is controllable we can set $s^1_{4,5} = \epsilon$ to avoid that sequence and the completing $\mathbf{S}^2$. By making $\mathbf{S}^2 = COACES(\mathbf{S}^2)$ we eliminate the non co-reachable states to obtain

$$\mathbf{S}^2{=}COACES\left(\left[\begin{array}{ccccccc|c} \epsilon & \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \lambda & \epsilon & \kappa & \epsilon & \epsilon & \epsilon & \epsilon & e \\ \epsilon & \epsilon & \epsilon & \mu & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \alpha & \underline{\epsilon} & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \alpha & \eta & \epsilon \\ \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \lambda & \epsilon & \epsilon \\ \epsilon & \epsilon & \lambda & \epsilon & \epsilon & \epsilon & \alpha & e \end{array}\right]\right)$$

or

$$\mathbf{S}^2{=}\left[\begin{array}{ccccccc|c} \epsilon & \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \lambda & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & e \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \alpha & \eta & \epsilon \\ \alpha & \epsilon & \epsilon & \epsilon & \epsilon & \lambda & \epsilon & \epsilon \\ \epsilon & \epsilon & \lambda & \epsilon & \epsilon & \epsilon & \alpha & e \end{array}\right].$$

Now computing $\mathbf{S}^2 = ACES(\mathbf{S}^2)$ we eliminate all unreachable states and obtain the supervisor

$$\mathbf{S}{=}ACES(\mathbf{S}^2){=}\begin{bmatrix} \epsilon & \alpha \\ \lambda & \epsilon \end{bmatrix}$$

as it is shown in Figure 20. Note that the final supervisor was presented with dimension 2, indeed it has same dimension 7, the same of $\mathbf{S}^2$. However, all the elements of rows and columns from $k = 3, ..., 7$ are all equal to $\epsilon$. This examples shows that the un-timed synthesis is a special case of the general time-varying supervisory control problem.

## 7 CONCLUSIONS

We have provided the framework for studying time-varying DES problems. The proposed time-varying automaton allows to model systems where the event lifetime varies during its execution. Such type of timed discrete event systems cannot be modelled by (max,+) automata. The paper has also provided a synthesis procedure that is based on the standard algorithm of supervisory control theory. However, the use of a non-traditional algebraic structure (dioid) and the representation of system and its specification in matrix format were introduced. This fact has allowed to achieve, with the proposed formalism, the synthesis of both un-timed supervisor, fixed lifetimes supervisor and time-varying lifetimes supervisors with the same basic algorithm.

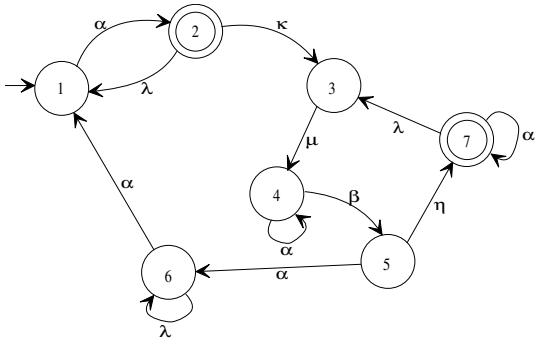


Figure 19: Un-timed behavior specification.

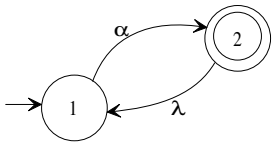


Figure 20: Un-timed supervisor.

For the general case the order of complexity of the proposed supervisor synthesis algorithm is $O(N^4)$ where $N$ is the dimension of the incidence matrix. However, if the specification incidence matrix is a sub-matrix of the system incidence matrix the order of complexity is reduced to $O(N^2)$. When the event lifetimes of the system model are fixed the complexity of the proposed algorithm is the same as that of the Ramadge and Wonham approach.

The proposed framework allows to deal with more general issues in the design of supervisory control problems as illustrated by the selected examples presented in the paper. Indeed, by the same basic synthesis procedure we can design finite state supervisors for un-timed DES, fixed lifetime DES, time-varying DES that can be either cyclic or acyclic. Thus, the proposed approach provides an extension of timed automata for representing more general classes of real-time problems like communication protocols.

## REFERENCES


Alur, R. (1997). Timed automata, *Proc. NATO-ASI Summer School, Antalaya, Turkey* .

Alur, R. e Dill, D. (1990). Automata for modeling real-time systems, *Proc. 17th International Colloquim on Automata, Languages and Programming, Lectures Notes on Computer Science, New York: Springer Verlag* **443**: 322–335.

Alur, R. e Dill, D. (1994). A theory of timed automata, *Theoretical Computer Science* **126**(2): 183–235.

Alur, R. e Dill, D. (1995). Automata-theoretic verification of real-time systems, *Technical report*, Computing Science Research (Bell Labs) and Computer Science Department (Stanford University).

Alur, R. e Henzinger, T. (1992). Back to the future: Towards a theory of timed regular languages, *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pp. 177–186.

Asarin, E. (1998). *Equations on Timed languages*, Hybrid Systems: Computation and Control, Springer-Verlag.

Berstel, J. e Reutenauer, C. (1988). *Rational Series and their Languages*, Springer.

Brandin, B. e Wonham, W. (1994). Supervisory control of timed discrete-event systems, *IEEE Transactions on Automatic Control* **39**(2): 329–342.

Cofer, D. e Garg, V. (1996). Supervisory control of real time discrete-event systems using lattice theory, *IEEE Transactions on Automatic Control* **41**(2): 199–209.

Fribourg, L. (1998). *A Closed-Form Evaluation for Extended Timed Automata*, Research Report LSV-98-2, Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan, France.

Klimann, I. (1999). *Langages, Séries et Contrôle de Trajectoires*, PhD thesis, l'Université Denis Diderot - Paris 7.

Lawford, M. (1997). *Model Reduction of Discrete Real-Time Systems*, PhD thesis, University of Toronto.

Ostroff, J. e Wonham, W. (1990). A framework for real-time discrete event control, *IEEE Transactions on Automatic Control* **35**(4): 386–397.

Ramadge, P. e Wonham, W. (1982). Supervision of discrete event processes, *Procedings of 21st Conference on Decision and Control* pp. 1228–1229.

Ramadge, P. e Wonham, W. (1987a). On the supremal controllable sublanguage of a given language, *SIAM Journal of Control and Optimization* **25**(3): 637–659.

Ramadge, P. e Wonham, W. (1987b). Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization* **25**(1): 206–230.

Saksena, M. e Selic, B. (1999). Real-time software design - state of the art and future challenges, *IEEE Canadian Review* **2**(32): 5–8.

Tripakis, S. (1998). *L'Analyse Formelle Des Systémes Temporisés En Pratique*, PhD thesis, L'Université Joseph Fourier.