A C++ toolbox to handle series for event-variant/time-variant (max,+) systems

Bertrand Cottenceau, Laurent Hardouin, Johannes Trunk

(EVENT|TIME)-VARIANT OPERATORS

B.COTTENCEAU, L.HARDOUIN, J.TRUNK

A C++ toolbox to handle series for event-variant/time-variant (max,+) systems

LARIS, University of Angers

– April 2019 –

CONTENTS

1 INTRODUCTION 5
1.1 Discrete event systems 5
1.1.1 Sequences, counter, dater 5
1.1.2 Synchronization 7
1.1.3 Operators 8
1.1.4 Semiring of operators 10
1.2 Timed Event Graphs and operators 13
1.3 Weighted Timed Event Graphs and operators 16
1.4 Timed Event Graphs with periodic holding times 19

1 INTRODUCTION

ETVO ((Event|Time)-Variant Operators) is a C++ library to handle the behaviour of a class of Discrete Event Systems (DES). This library contains a set of C++ classes to describe and to compute the formal series involved in the description of Timed Event Graphs (series in $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$), Weighted Timed Event Graphs (series in $\mathcal{E}[\![\delta]\!]$) and Timed Event Graphs with periodic holding times (series in $\mathcal{T}[\![\gamma]\!]$).

The variables of the formal series considered in this work can be assimilate to basic systems called *operators*. An operator is a mapping able to transform a signal. In the context of DES, a signal is for instance the list of events in a time axis.

A library called MinMaxGD already exists [6] to handle formal series in $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$. In MinMaxGD, series are well suited to describe time-invariant and event-invariant (min,+)/(max,+) systems. ETVO encompasses the library MinMaxGD and extends its set of classes to manage formal series for specific event-variant and time-variant systems. However, some similarities remain. For instance, as in $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$, the formal series in $\mathcal{E}[\![\delta]\!]$ and $\mathcal{T}[\![\gamma]\!]$ are still written in a standard form with an ultimate periodic pattern

$$s = p \oplus q(\gamma^{\nu}\delta^{\tau})^{*}.$$

This document gives an overview on DES considered and their modelling with ETVO library.

The reader can find complementary presentations in [4], [9], [11].

1.1 DISCRETE EVENT SYSTEMS

1.1.1 Sequences, counter, dater

We will first recall some features related to the modelling of Discrete Event Systems (DES). A DES is a dynamic system which is driven by the occurrence of punctual phenomena called *events*. An event reflects the moment when the system operates a state evolution. In a manufacturing system modelled as a DES, the events are for instance: the arrival of a part in a stock, the moment when a task is starting or ending, the moment when a resource is being

seized or released etc. Considering a system as a DES means that its evolution is described by a sequence of events along a time axis.

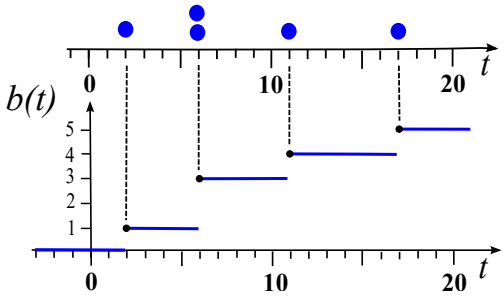

Figure 1: Sequence of events and counter function.

For instance in Fig.1, a sequence of events named b is given. Each occurrence of b is depicted by a big dot. This sequence of events can be described by a set of pairs (b_k, t_k) where $k \in \mathbb{N}$, $t_k, b_k \in \mathbb{Z}$, b_k denotes the k-th occurrence of event b and t_k its occurrence time. For the sequence in Fig.1, the occurrences of b are given by

$$\{b\} = \{(b_0, 2), (b_1, 6), (b_2, 6), (b_3, 11), (b_4, 17), ...\}.$$

The first occurrence of b is at date $t = 2$, then two simultaneous occurrences b appear at $t = 6$, etc. A sequence of events can be infinite,

$$\{a\} = \{(a_k, 2k+1)\} = \{(a_0, 1), (a_1, 3), (a_2, 5), ...\}_{k \in \mathbb{N}}$$

or finite

$$\{b\} = \{(b_0, 2), (b_1, 6), (b_2, 6), (b_3, 11), (b_4, 17), (b_5, +\infty)\},$$

which means that the sixth occurrence of b never occurs.

There is two alternative representations of event sequences. The first one is given by a *counter function*

$$\text{[counter function] } b(t) : \mathbb{Z} \to \mathbb{Z}, t \mapsto \text{ the number of events } b \text{ occurred before date } t.$$

Fig.1 depicts the counter function $b(t)$ associated to the occurrences of events b. Let us remark that a counter function is naturally monotonic.

The other representation, symmetrical, is called *dater function*. Such a function is defined by

$$\text{[dater function] } b(k) : \mathbb{Z} \to \mathbb{Z}, k \mapsto \text{ date of the k-th occurrence of event b.}$$

For the sequence $\{b\}$ depicted in Fig.1, the first values of $b(k)$ are $b(0) = 2, b(1) = 6$, etc.

Sequences of events, counter functions and dater functions all act as *signals* for DES since they encode the history of the occurrences of a given event. Therefore, the name 'signal' will be sometimes used in place of sequence, counter or dater.

Notation 1 (Sets of signals) *We denote by*

$$\Sigma_s \ : \ \textit{the set of event sequences} \tag{1}$$
$$\Sigma_c \ : \ \textit{the set of counter functions} \tag{2}$$
$$\Sigma_d \ : \ \textit{the set of dater functions} \tag{3}$$

1.1.2 Synchronization

In the DES that we consider, the synchronization is clearly the prevailing phenomenon. The synchronization of two sequences of events (denoted by the $\oplus$ operator) is a sequence of events. The synchronization can be expressed as follows: $\{a\}, \{b\} \in \Sigma_s$, then $\{c\} = \{a\} \oplus \{b\}$ means that each occurrence c_k is as soon as possible after a_k and b_k. More explicitely,

$$\{(c_k, \tau_k)\} = \{(a_k, t_k)\} \oplus \{(b_k, t'_k)\} = \{(c_k, \max(t_k, t'_k)\}$$

For the sequences given in Fig.2, we have $\{a\} = \{(a_0,1),(a_1,7),(a_2,10),(a_3,10),(a_4,14)\}$ and $\{b\} = \{(b_0,2),(b_1,4),(b_2,8),(b_3,13)\}$, therefore the synchonization of $\{a\}$ and $\{b\}$ is given by

$$\{c\} = \{(c_0,2),(c_1,7),(c_2,10),(c_3,13)\}.$$

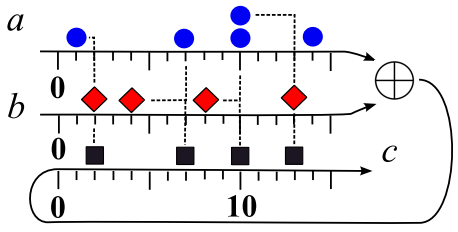

Figure 2: Synchronization of two sequences $\{a\} \oplus \{b\}$

When the signals are described as counter or dater functions, the synchronization is expressed as follows : $a, b \in \Sigma_c$

$$\text{[counter]},\ (a \oplus b)(t) = \min(a(t), b(t))$$

For the dater description, $a, b \in \Sigma_d$,

$$\text{[dater]},\ (a \oplus b)(k) = \max(a(k), b(k)).$$

It is important to remark that the synchronization is an idempotent operation on signals : $\forall a \in \Sigma, a \oplus a = a$.

1.1.3 Operators

In the ETVO library, the key feature is not really on the description of signals. The core of the library lies on the description of *systems* able to transform signals. A system which maps a signal to a signal is called an *operator*. For example in Fig.3, the system S can be considered as an operator able to transform the sequence $\{a\} = \{(a_0,1),(a_1,5),(a_2,9)\}$ into another sequence of events, namely $\{b\} = \{(b_0,3),(b_1,6),(b_2,6),(b_3,9)\}$. System S is an operator s.t. $S\{a\} = \{b\}$.

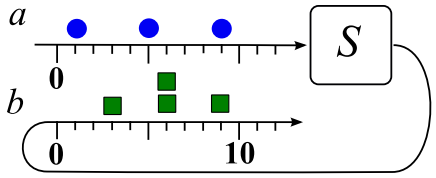

Figure 3: Operator S mapping $\{a\}$ to $\{b\} : \{b\} = S\{a\}$

Remark 1 *It is important to note that changing the type of signal does not change the nature of the operator considered. Whether the signal is a sequence in Σ_s or a counter function in Σ_c, the operator remains the same. The presentation given hereafter is not too formal. We will therefore allow ourselves to keep the same notation for an operator even if, for reasons of simplification, the signals handled will not always be of the same type along the presentation.*

The ETVO library provides a set of basic operators, only 5 basic operators, that can be composed to describe more complex systems. The most simple operators that are described in ETVO are the time-shift and the event-shift operator.

The *time-shift* of τ time units is an operator denoted δ^τ. The δ^τ operator transforms a sequence of events into a sequence where each events is time shifted. For $\{a\}, \{b\} \in \Sigma_s$, $\delta^\tau\{a\} = \{b\}$ is expressed as

$$\delta^\tau\{(a_k, t_k)\} = \{(b_k, t_k + \tau)\}.$$

We give an example in Fig.4, δ^3 operates a shift of 3 time units, say

$$\delta^3\{(a_0,1),(a_1,5),(a_2,5),(a_3,9)\} = \{(b_0,4),(b_1,8),(b_2,8),(b_3,12)\}.$$

When expressed as a mapping on counter functions, the δ^τ operator can be defined by :

$$\forall a \in \Sigma_c, \forall t, [\delta^\tau a](t) = a(t-\tau).$$

When expressed as a mapping on dater functions, the δ^τ operator can be defined by :

$$\forall a \in \Sigma_d, \forall k, [\delta^\tau a](k) = a(k) + \tau.$$

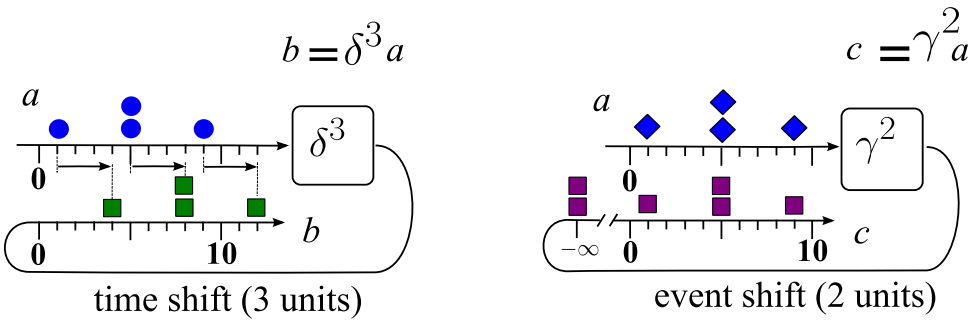

Figure 4: Time-shift/Event-shift : $\{b\} = \delta^3\{a\}$ and $\{c\} = \gamma^2\{a\}$

The *event-shift* of ν is another basic operator denoted γ^ν. The γ^ν operator produces a sequence of events where the event numbering is shifted. At any given time, the difference between the number of output and input events is fixed and equals to ν. Approximately, we have $\gamma^\nu\{(a_k, t_k)\} = \{(b_{k+\nu}, t_k)\}$. Operator γ^ν maps the event a_k to $b_{k+\nu}$. In order to be consistent, even if the input sequence has no event, ν events are produced by the γ^ν operator at date $-\infty$. Written differently, $\gamma^\nu\{(a_0, +\infty)\} = \{(b_0, -\infty), (b_1, -\infty), ..., (b_{\nu-1}, -\infty), (b_\nu, +\infty)\}$.

For the example given in Fig.4, the first two occurrences of b are at date $-\infty$,

$$\gamma^2\{(a_0,1),(a_1,5),(a_2,5),(a_3,9)\} = \{(b_0,-\infty),(b_1,-\infty),(b_2,1),(b_3,5),(b_4,5),(b_5,9)\}.$$

Operator γ^ν can be expressed as well as a mapping on counter or dater functions. When expressed as a mapping on counter functions:

$$\forall a \in \Sigma_c, [\gamma^\nu a](t) = a(t) + \nu.$$

When expressed as a mapping on dater functions : $a \in \Sigma_d$,

$$\forall a \in \Sigma_d, [\gamma^\nu a](k) = a(k - \nu).$$

1.1.4 Semiring of operators

All the basic operators handled in ETVO share the same property, they are all additive. For a, b being two signals, an operator S is said to be additive if

$$S(a \oplus b) = Sa \oplus Sb.$$

Clearly, operators γ^ν and δ^τ have this property. But we will introduce other basic operators satisfying this property as well. The set of additive operators can be endowed with an idempotent semiring structure as follows.

Notation 2 (Idempotent semiring of additive operators) *The set $\mathcal{O}$ of additive operators, endowed with the sum and the product given below, is an idempotent semiring: $h_1, h_2 \in \mathcal{O}$, $x \in \Sigma_c$,*

$$h_1 \oplus h_2 \triangleq \forall x, (h_1 \oplus h_2)(x) = h_1 x \oplus h_2 x \quad (4)$$

$$h_1.h_2 \triangleq \forall x, (h_1 h_2)(x) = h_1(h_2 x) \quad (5)$$

In this semiring, the neutral element for the addition is an operator denoted ε and the neutral element for the product is the identity operator denoted e, $\forall x \in \Sigma_c, e(x) = x$. The semiring $\mathcal{O}$ is not commutative, $h_1 h_2 \neq h_2 h_1$.

Notation 3 (Semiring $\mathcal{M}_{\mathbf{in}}^{\mathbf{ax}}[\![\gamma,\delta]\!]$) *The set of operators obtained by composing γ^ν, δ^τ and ε is a subsemiring of $\mathcal{O}$ denoted $\mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]$. The identity operator can be expressed $e = \gamma^0 = \delta^0$.*

The semiring $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$ was introduced in [3] and also detailed in [1]. It gives an algebraic framework to formally handle DES. The next theorem recalls some well-known facts on $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$ that can lead to a computation toolbox. A software library, called MinMaxGD [6], is available to handle rational computations in $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$.

The next theorem recalls some well-known equalities in $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$.

Theorem 1 *In $\mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]$, we have*

$$\gamma^n\delta^t = \delta^t\gamma^n \tag{6}$$
$$\gamma^n\gamma^{n'} = \gamma^{n+n'} \tag{7}$$
$$\delta^t\delta^{t'} = \delta^{t+t'} \tag{8}$$
$$\gamma^n \oplus \gamma^{n'} = \gamma^{\min(n,n')} \tag{9}$$
$$\delta^t \oplus \delta^{t'} = \delta^{\max(t,t')} \tag{10}$$

Because of (6), $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$ is a commutative idempotent semiring.

In addition to γ^n and δ^t, ETVO library introduces two basic operators for *event-variant* systems denoted μ_m (multiplier) and β_b (batch). The μ_m operator multiplies events. Each input event produces instantaneously m output events. For the example depicted in Fig.5, $\{a\} \in \Sigma_s$, $\{b\} = \mu_2\{a\}$,

$$\mu_2\{(a_0,1),(a_1,4),(a_2,7),...\} = \{(b_0,1),(b_1,1),(b_2,4),(b_3,4),(b_4,7),(b_5,7),...\}$$

Conversely, for the batch operator β_b, b input events are needed to produce one output event. We have, $\{a\} \in \Sigma_s$, $\{c\} = \beta_3\{a\}$,

$$\beta_3\{(a_0,1),(a_1,4),(a_2,7),...\} = \{(c_0,7),(c_1,15),...\}.$$

Operators μ_m and β_b can be expressed as mapping on counter functions as follows: $a \in \Sigma_c$

$$\forall a, [\mu_m a](t) = a(t) \times m, \qquad [\beta_b a](t) = \lfloor a(t)/b \rfloor.$$

Finally, the ETVO library introduces a supplementary operator for *time-variant* systems denoted Δ_T. The Δ_T operator is a synchronization on dates which are a multiple of T. For instance, the Δ_3 operator delays all the input events up to the next date in $3\mathbb{Z}$. All the output events are then synchronized on dates in $3\mathbb{Z}$.

In the example given Fig.6, we have $\{b\} = \Delta_3\{a\}$,

$$\Delta_3\{(a_0,1),(a_1,4),(a_2,6),(a_3,9),...\} = \{(b_0,3),(b_1,6),(b_2,6),(b_3,9),...\}.$$

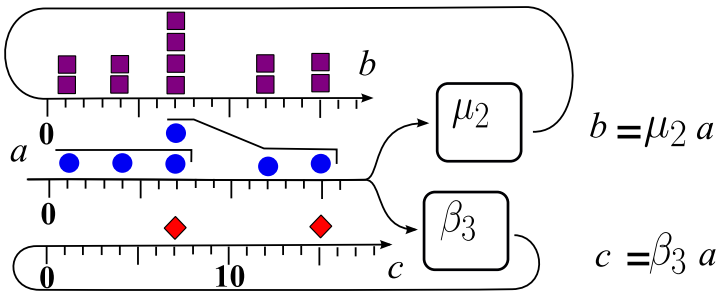

Figure 5: Event Muliplier/Batch : $\{b\} = \mu_2\{a\}$ and $\{c\} = \beta_3\{a\}$

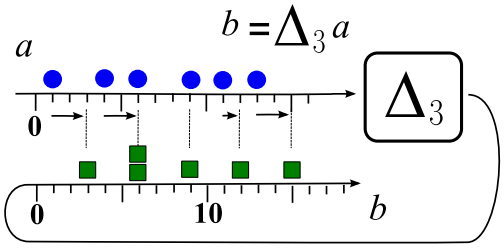

Figure 6: Date synchronization : $\{b\} = \Delta_3\{a\}$

The operator Δ_T can be expressed as a mapping on dater functions as follows: $a \in \Sigma_d$,

$$\forall a,\ [\Delta_T a](k) = \lceil a(k)/T \rceil \times T.$$

It is worth noticing that this operator can be interpreted as a time-variant time-shift operator. For instance, the gap between a_0 and b_0 is 2 time units whereas the gap is 0 time unit between a_2 and b_2.

In summary, the ETVO library introduces the 5 basic operators recalled below:

$$\begin{array}{rcl} \delta^{\tau} & : & \text{time-shift} \\ \gamma^{\nu} & : & \text{event-shift} \\ \mu_m & : & \text{event-multiplier} \\ \beta_b & : & \text{event-batch} \\ \Delta_T & : & \text{date synchronization} \end{array}$$

By considering only subsets of these operators, we obtain 3 different idempotent semirings, which are subsemirings of $\mathcal{O}$, that are usefull for DES modelling

$$\begin{array}{rcl} \mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!] & : & \text{semiring of sums and products in } \{\varepsilon,\gamma^{\nu},\delta^{\tau}\} \\ \text{[event-variant] } \mathcal{E}[\![\delta]\!] & : & \text{semiring of weight-balanced sums and products in } \{\varepsilon,\gamma^{\nu},\delta^{\tau},\mu_m,\beta_b\} \\ \text{[time-variant] } \mathcal{T}[\![\gamma]\!] & : & \text{semiring of sums and products in } \{\varepsilon,\gamma^{\nu},\delta^{\tau},\Delta_T\} \end{array}$$

The semiring $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$ is detailed in [3], [1] and the MinMaxGD toolbox to handle peridoic series is presented in [6]

The semiring $\mathcal{E}[\![\delta]\!]$ is presented in [4],[5],[9],[10].

The semiring $\mathcal{T}[\![\gamma]\!]$ is introduced in [11].

All these algebraic structures are more detailed in the PhD thesis of Johannes Trunk.

Proposition 1 *In semiring $\mathcal{O}$, operators γ^n, δ^t, μ_m, β_b, Δ_T satisfy:*

$$\begin{array}{rclrclrcll} \gamma^1\delta^1 & = & \delta^1\gamma^1 & \gamma^n\gamma^{n'} & = & \gamma^{n+n'} & \delta^t\delta^{t'} & = & \delta^{t+t'} & (f1) \\ \gamma^n \oplus \gamma^{n'} & = & \gamma^{\min(n,n')} & \delta^t \oplus \delta^{t'} & = & \delta^{\max(t,t')} & & & & (f2) \\ \mu_m\delta^1 & = & \delta^1\mu_m & \beta_b\delta^1 & = & \delta^1\beta_b & \beta_m\mu_m & = & e & (f3) \\ \Delta_T\gamma^1 & = & \gamma^1\Delta_T & \Delta_T\delta^T & = & \delta^T\Delta_T & & & & (f4) \\ \mu_m\mu_{m'} & = & \mu_{m\times m'} & \beta_b\beta_{b'} & = & \beta_{b\times b'} & & & & (f5) \\ \mu_m\gamma^1 & = & \gamma^m\mu_m & \gamma^1\beta_b & = & \beta_b\gamma^b & & & & (f6) \end{array}$$

1.2 TIMED EVENT GRAPHS AND OPERATORS

The formal series handled in ETVO are well suited to describe the behaviour of some subclasses of timed Petri nets. First, we give here a very short description of Timed Event Graphs. This graphical model is better explained in other references such as [1, chap.2],[2], [8, chap.7],[7].

A Timed Event Graph (TEG) is a timed Petri net - with P the set of places, T the set of transitions and $A \subset (P \times T) \cup (T \times P)$ the set of edges - such that each place has exactly one upstream and one downstream transition. A place $p_k \in P$ can have a positive holding time value $\tau \in \mathbb{N}$ and an initial marking denoted $M_0(p_k) \in \mathbb{N}$. The holding time is the minimal time a token needs to stay in a place before being able to cross a downstream transition.

We denote by $p^\bullet$ (resp. ${}^\bullet p$) the downstream (resp. upstream) transition of place p, and we denote by $t^\bullet$ (resp. ${}^\bullet t$) the set of downstream (resp. upstream) places of transition t. When one considers only the *earliest firing rule*, a transition t_j fires as soon as each place $p_l \in {}^\bullet t_j$

contains at least 1 available token. Then one token is removed from each place p_l, and one token is added to each place $p_k \in t_j^\bullet$. In fact, since the behaviour is at the earliest, all tokens that can contribute to the crossing of transitions are collected as soon as possible.

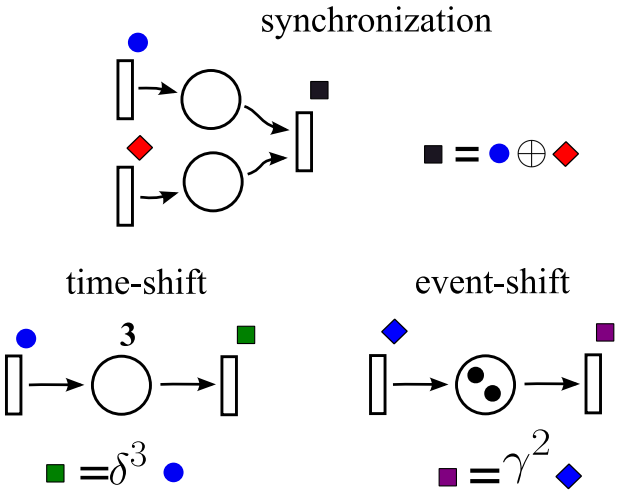

Figure 7: Elementary Timed Event Graphs

Timed Event Graphs provide a graphical representation of different dynamical phenomena. For a TEG, signals (sequence, counters, daters) are attached to transitions. The marking of the net characterizes the state and a transition firing is an event since it corresponds to the state evolution. A transition with two upstream places describes the synchronization of events. A place with a holding time leads to a time-shift between the upstream and the downstream transitions. Finally, the initial marking acts as a shift in the event numbering.

Therefore, the structure of a TEG can be translated into a block-diagram where only γ^ν and δ^τ operators are used and the synchronization of signals is denoted by the $\oplus$ symbol. For TEGs, the semiring considered for the computation is $\mathcal{M}_{\text{in}}^{\text{ax}}[\![\gamma,\delta]\!]$. From a practical point of view, this computation can be made by the MinMaxGD library.

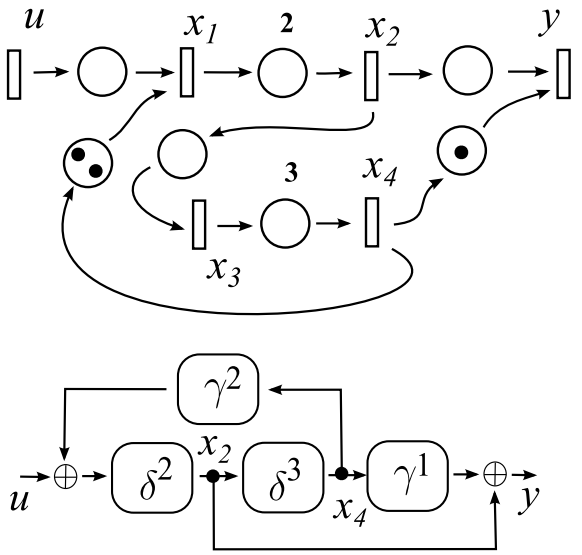

Figure 8: TEG decomposition in basic operators

In Fig.8, a TEG is depicted. Its decomposition in γ^{ν} and δ^{τ} operators is given as a block-diagram. The input corresponds to the signal u and the output is given by the signal y. All the signals are related to each other by the next relations,

$$\begin{aligned} x_1 &= u \oplus \gamma^2 x_4, \\ x_2 &= \delta^2 x_1, \\ x_3 &= x_2, \\ x_4 &= \delta^3 x_3, \\ y &= x_2 \oplus \gamma^1 x_4. \end{aligned}$$

We can describe this system in a matrix form where $x = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}'$ is a vector of signals,

$$\begin{cases} x &= Ax \oplus Bu \\ y &= Cx \end{cases}$$

with

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \gamma^2 \\ \delta^2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \delta^3 & \varepsilon \end{pmatrix}, B = \begin{pmatrix} e \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}, C = \begin{pmatrix} \varepsilon & e & \varepsilon & \gamma^1 \end{pmatrix}.$$

The behaviour of this system can be described by a rational expression $y = CA^*Bu = (CB \oplus CAB \oplus CA^2B...)u$. In this case, the computation gives

$$y = (\delta^2 \oplus \gamma^1\delta^5)(\gamma^2\delta^5)^*u.$$

The rational expression is an operator that describes how the input signal is transformed into the ouput signal by the system.By analogy with the classical system theory, this operator is the transfer function of the TEG.

This computation can be obtained by the MinMaxGD library included in ETVO. For this short example, the C++ script is given below.

```
// Simple C++ example to compute the transfer series of a TEG
#include "etvo.h"
using namespace std;   // namespace for cout object
using namespace etvo;  // namespace for ETVO classes
int main()
{
    // series corresponds to MinMaxGD series
        matrix<series> A(4,4), B(4, 1), C(1, 4);
        B(0,0)= gd(0,0);        //g0.d0=e
        C(0,1)= gd(0,0);
        C(0,3)= gd(1,0);        //g1.d0=g1
        A(0,3)= gd(2,0);
        A(1,0)= gd(0,2);        //g0.d2=d2
        A(2,1)= gd(0,0);
        A(3,2)= gd(0,3);

        matrix<series>  H = C * A.star() * B;
        cout << H(0,0) << endl;
        // output : (g0.d2+g1.d5).[g2.d5]*
}
```

1.3 WEIGHTED TIMED EVENT GRAPHS AND OPERATORS

A Weighted Timed Event Graph is a Timed Event Graph the edges of which have an integer weight. For $p_k \in P$ a place, the edge $t_i \rightarrow p_k$ (resp. $p_k \rightarrow t_o$) is valued by a strictly positive integer denoted $\omega_i(p_k)$ (resp. $\omega_o(p_k)$) (the weights of the edges). In order to avoid confusion with holding times, weights of edges are denoted between brackets, e.g. $\langle 2 \rangle$.

Moreover, $t_i \to p_k \to t_o$ defines an elementary path denoted π_k the gain of which is given by $\Gamma(\pi_k) \triangleq \omega_i(p_k)/\omega_o(p_k) \in \mathbb{Q}$.

The weights describe how many tokens are consumed/produced by each transition firing. When one considers only the *earliest firing rule*, a transition t_j fires as soon as each input place p_l of t_j contains at least $\omega_o(p_l)$ available token(s). Then $\omega_o(p_l)$ token(s) is(are) removed from each input place p_l of t_j, and $\omega_i(p_k)$ token(s) is(are) added to each output place p_k of t_j.

Fig.9 illustrates the effect of weights as basic operators. A weight on the output edge of a transition describes a multiplication of events. The μ_m operator can model this pehnomenon. Conversely, a weight on the input edge of a transition describes a batch operation which is modeled by a β_b operator.

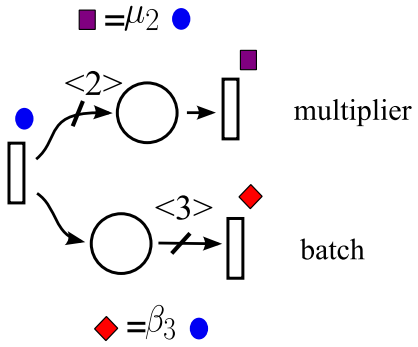

Figure 9: Elementary Weighted Timed Event Graphs

In Fig.10, a Weighted TEG is depicted. Its decomposition in basic operators is given as a block-diagram. All the relation between signals are expressed below :

$$\begin{aligned} x_1 &= \beta_2 u \oplus \beta_3 \gamma^5 \mu_2 x_4 \\ x_2 &= \delta^2 \mu_3 x_1 \\ x_3 &= x_2 \\ x_4 &= \beta_2 \delta^5 x_3 \\ y &= \beta_2 x_2 \oplus \gamma^1 x_4 \end{aligned}$$

We can describe this system in a matrix form where $x = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$,

$$\begin{cases} x = Ax \oplus Bu \\ y = Cx \end{cases}$$

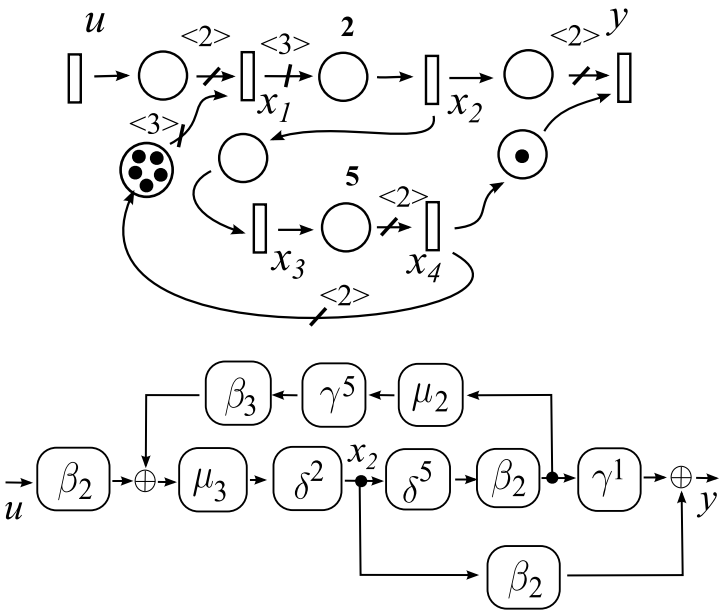

Figure 10: WB-TEG decomposition in basic operators

with

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \beta_3\gamma^5\mu_2 \\ \delta^2\mu_3 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \beta_2\delta^5 & \varepsilon \end{pmatrix}, B = \begin{pmatrix} \beta_2 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}, C = \begin{pmatrix} \varepsilon & \beta_2 & \varepsilon & \gamma^1 \end{pmatrix}$$

The behaviour of this system can be described by a rational expression $y = CA^*Buu$. For this example, the computation withthe ETVO library gives

$$y = \left((\mu_3\beta_4\gamma^2 \oplus \gamma^1\mu_3\beta_4)\delta^2 \oplus (\gamma^1\mu_3\beta_4\gamma^2 \oplus \gamma^2\mu_3\beta_4)\delta^7\right)(\gamma^2\delta^7)^*$$

For this short example, the C++ script is given below.

```
// Simple C++ example to compute the transfer series of a WB-TEG
#include "etvo.h"
using namespace std;
using namespace etvo;

int main()
{
```

```
        matrix<seriesEd> A(4,4), B(4,1), C(1,4);
        B(0, 0) = eb(2); //b2
        C(0, 1) = eb(2);
        C(0, 3) = eg(1); //g1
        A(0, 3) = eb(3)*eg(5)*em(2); //b3.g5.m2
        A(1, 0) = em(3)*ed(2); //m3.d2
        A(2, 1) = eg(0); //g0=e
        A(3, 2) = eb(2)*ed(5); //b2.d5
        matrix<seriesEd> H=C*A.star()*B;
        H(0, 0).toRight();
        cout << H(0,0) << endl;
//output=((m3.b4.g2+g1.m3.b4).d2+(g1.m3.b4.g2+g2.m3.b4).d7).[g2.d7]*
}
```

1.4 TIMED EVENT GRAPHS WITH PERIODIC HOLDING TIMES

ETVO can handle a basic time-variant operator denoted Δ_T. The Δ_T operator is a synchronization on dates in $T\mathbb{Z}$. With this operator, we can model delays changing as time changes. For instance, the Δ_3 operator can be considered as a delay δ^τ the value of which is time-variant. The Δ_3 operator adds no delay for input events occuring at dates in $3\mathbb{Z}$. For instance, $\Delta_3\{(a_0,0),(a_1,6)\} = \{(b_0,0),(b_1,6)\}$. For an input event occuring at dates in $3\mathbb{Z}+1$, the delay is 2 time units : $\Delta_3\{(a_0,1),(a_1,1),(a_2,4)\} = \{(b_0,3),(b_1,3),(b_2,6)\}$. And the delay is only 1 time unit for input events occuring at dates in $3\mathbb{Z}+2$. We summarize this with the notation $\Delta_3 = \delta^{\langle 0,2,1\rangle}$. The time-variant delays thus obtained are necessarily with a periodic sequence of values.

By considering DES with this kind of time-variant delays, we can model Timed Event Graphs with periodic holding times. This class of models is called Periodic Time-variant Event Graphs (PTEGs) in the thesis of J.Trunk. Each periodic time-variant delay can be obtained as a finite composition of fixed delays δ^τ and Δ_T operators.

In Fig.11, a PTEG is depicted as well as its decomposition in time-variant operators. As said before, $\delta^{\langle 0,2,1\rangle} = \Delta_3$. For the others time-variant holding times, we have the equivalence:

$$\begin{aligned} \delta^{\langle 1,0\rangle} &= \delta^1\Delta_2\delta^{-1} \\ \delta^{\langle 2,3,2\rangle} &= \delta^2\Delta_3\delta^{-2} \oplus \delta^1\Delta_3 \end{aligned}$$

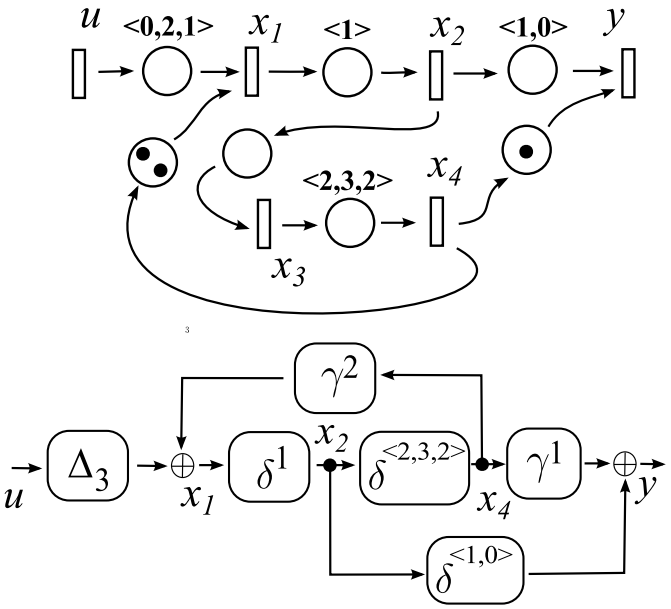

Figure 11: PTEG decomposition into operators

We can describe this system in a matrix form where $x = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}'$,

$$\begin{cases} x &= Ax \oplus Bu \\ y &= Cx \end{cases}$$

with

$$A = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \gamma^2 \\ \delta^1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \delta^{\langle 2,3,2 \rangle} & \varepsilon \end{pmatrix}, B = \begin{pmatrix} \Delta_3 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}, C = \begin{pmatrix} \varepsilon & \delta^{\langle 1,0 \rangle} & \varepsilon & \gamma^1 \end{pmatrix}.$$

The behaviour of this system can be described by a rational expression $y = Gu = CA^*Bu$. For this example, the computation with the ETVO library gives

$$y = \delta^{\langle 1,4,3,2,3,2 \rangle}\gamma^0 \oplus (\delta^{\langle 4,6,5 \rangle}\gamma^1 \oplus \delta^{\langle 5,8,7,6,7,6 \rangle}\gamma^2)(\delta^3\gamma^2)^* u.$$

For this short example, the C++ script is given below.

```
// Simple C++ example to compute the transfer series of a PTEG
```

```
#include "etvo.h"
using namespace std;
using namespace etvo;
int main(){
        matrix<seriesTg> A(4, 4), B(4, 1), C(1, 4);
        B(0, 0) = tD(3); //D3
        C(0, 1) = td({1,0}); //d<1,0>
        C(0, 3) = tg(1); //g1
        A(0, 3) = tg(2);
        A(1, 0) = td(1); //d1
        A(2, 1) = tg(0);
        A(3, 2) = td({2,3,2});
        matrix<seriesTg>  G=C*A.star()*B;
        G(0,0).toRight();
        cout << G(0,0).toStringAsDeltaVar() << endl;
//output=((d<1,4,3,2,3,2>.g0))+(d<4,6,5>.g1+d<5,8,7,6,7,6>.g2).[d3.g2]*
}
```

BIBLIOGRAPHY

[1] F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley and Sons, New York, 1992.

[2] G. Cohen. Analisis y control de sistemas de eventos discretos: de redes de petri temporizadas al algebra, 2001.

[3] G. Cohen, P. Moller, J.P. Quadrat, and M. Viot. Algebraic Tools for the Performance Evaluation of Discrete Event Systems. *IEEE Proceedings: Special issue on Discrete Event Systems*, 77(1):39–58, January 1989.

[4] B. Cottenceau, L. Hardouin, and J.-L. Boimond. Modeling and Control of Weight-Balanced Timed Event Graphs in Dioids. *IEEE Trans. on Autom. Cont.*, vol. 59:1219–1231, May 2014.

[5] Bertrand Cottenceau, Laurent Hardouin, and Johannes Trunk. Weight-balanced timed event graphs to model periodic phenomena in manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 14(4):1731–1742, 2017.

[6] L. Hardouin, B. Cottenceau, and M.Lhommeau. Minmaxgd, a toolbox to handle periodic series in semiring minmax[[g,d]]., 2013.

[7] Laurent Hardouin, Bertrand Cottenceau, Ying Shang, Jörg Raisch, et al. Control and state estimation for max-plus linear systems. *Foundations and Trends® in Systems and Control*, 6(1):1–116, 2018.

[8] B. Heidergott, G.J. Olsder, and J.van der Woude. *Max Plus at Work - Modelling and Analysis of Synchronized Systems - A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, Princeton, 2006.

[9] J. Trunk, B. Cottenceau, L. Hardouin, and J. Raisch. Model decomposition of weight-balanced timed event graphs in dioids: Application to control synthesis. In *IFAC World Congress*, Toulouse, France, 2017.

[10] Johannes Trunk, Bertrand Cottenceau, Laurent Hardouin, and Jörg Raisch. Output reference control for weight-balanced timed event graphs. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4839–4846. IEEE, 2017.

[11] Johannes Trunk, Bertrand Cottenceau, Laurent Hardouin, and Jörg Raisch. Model decomposition of timed event graphs under partial synchronization in dioids. *IFAC-PapersOnLine*, 51(7):198–205, 2018.