AN ALGEBRA OF TIME-CONSUMING COMPUTATIONS [1]

Józef Winkowski
Instytut Podstaw Informatyki PAN
01-237 Warszawa, ul. Ordona 21, Poland

Abstract

Structures for representing time-consuming computations are introduced and operations of composing such structures sequentially and in parallel are defined. Thus a monoidal category satisfying certain specific axioms of associativity and symmetry is obtained which is generated by a set of indecomposable time-consuming computations and in which the equality of morphisms is provable from axioms. Within this category one can represent behaviours of timed Petri nets.

Key words:
computation, time-consuming computation, symmetry, interchange, sequential composition, parallel composition, monoidal category, timed Petri net, timed computation.

1 Introduction

In [W 80] processes in safe Petri nets have been represented by labelled partially ordered sets of a sort and operations of composing such processes sequentially and in parallel have been introduced. In [W 82] the algebras of processes thus obtained have been characterized as partially monoidal categories satisfying some specific axioms. In [DMM 89] and [DMM 91] similar but more advanced results have been presented for arbitrary Place/Transition nets.

In this paper we are interested in time-consuming computations as those in timed Petri nets with tokens carrying information. We want to describe such computations by specifying the tokens which take part in a computation, how these tokens cause each other, and the delays with which tokens are obtained from their causal ancestors. We use for this purpose mathematical structures obtained by modifying the concept of [DMM 91] of concatenable computations of standard Petri nets. However, in contrast to [DMM 91], we introduce and study these structures as independent objects, that is without deriving them from Petri nets.

[1] This work has been supported by KBN - the Polish State Committee for Scientific Research (Grant No. 2 2047 92 03). It has been published in 1992 as the report 722 of the Institute of Computer Science of the Polish Academy of Sciences.

Our aim is to adopt to time-consuming computations the operations defined in [DMM 91] on concatenable computations and the respective results. We consider arbitrary finite time-consuming computations and operations of composing such computations sequentially and in parallel. With these operations the considered computations constitute a symmetric strict monoidal category. We show that each finite time-consuming computation can be obtained by combining computations which cannot be decomposed into simpler components. Consequently, the category of finite time-consuming computations is generated by the set of indecomposable finite time-consuming computations. The sets of time-consuming computations of concrete timed Petri nets constitute subcategories of this category generated by the respective subsets of indecomposable computations, or parts of such subcategories. Finally, by applying the argumentation of [DMM 91] it can be shown that the equality of finite time-consuming computations is provable.

2 Time-consuming computations

Time-consuming computations can be viewed as real-time processes which absorb and produce tokens with data. The tokens occurring in a computation of this type constitute a set X. The causal relation between tokens and the respective delays can be reflected by a function $d : X \times X \to \{-\infty\} \cup [0, +\infty)$, where $d(x, y) \in [0, +\infty)$ is the delay between x and y whenever y is a (direct or indirect) causal successor of x, and $d(x, y) = -\infty$ whenever y is not a causal successor of x. Each token $x \in X$ has a value $e(x)$ from a set V of values. For every value $v \in V$, the initial tokens with this value can be arranged into a sequence s_v and the terminal tokens into a sequence t_v. The respective formalization is as follows.

2.1. Definition. A *delay function* on a set X is a mapping $d : X \times X \to \{-\infty\} \cup [0, +\infty)$ such that:

(df1) $d(x, x) = 0$,

(df2) $x \neq y$ implies $d(x, y) = -\infty$ or $d(y, x) = -\infty$,

(df3) $d(x, y) + d(y, z) \leq d(x, z)$. □

Due to (df1) - (df3) the relation $\leq$, where $x \leq y$ stands for $d(x, y) \neq -\infty$, is a partial order on X (the order *induced* by d) with a subset X_{min} of minimal elements and a subset X_{max} of maximal elements.

2.2. Definition. A delay function d on X is said to be *proper* if it satisfies the following two conditions:

(pdf1) each maximal chain of X ordered by the order induced by d has a member in each maximal antichain,

(pdf2) for each pair (x, y) such that $x \leq y$ there exists a maximal chain Z from x to y such that $d(x, y) = d(x, z_1) + d(z_1, z_2) + d(z_2, y)$ for all $z_1, z_2 \in Z$ such that $z_1 \leq z_2$. □

The property (pdf1), known in the theory of Petri nets as K-density, is assumed in order to make the maximal antichains suitable for representing the possible global states of the respective time-consuming computation. It has the following two consequences.

2.3. Proposition. If a delay function d on X satisfies (pdf1) then for each maximal antichain $Y \subseteq X$ the order induced on X by d is the transitive closure of the union of the restrictions of this order to the subsets $\downarrow Y = \{x \in X : x \leq y \text{ for some } y \in Y\}$ and $\uparrow Y = \{x \in X : x \leq y \text{ for some } y \in Y\}$. $\Box$

It suffices to show that $x \leq y$ with $x \in \downarrow Y$ and $y \in \uparrow Y$ implies $x \leq z$ and $z \leq y$ for some $z \in Y$. To this end it suffices to take a maximal chain containing x and y and its member belonging to Y. $\Box$

2.4. Proposition. If a delay function d on X satisfies (pdf1) then the set of maximal antichains of X with the partial order defined by assuming $Y \sqsubseteq Y'$ whenever each $x \in X$ has an upper bound $y \in Y'$ is a lattice. $\Box$

Proof: We define $Y \sqcup Y'$ as the set of all $x \in X$ of the form $max(ZY, ZY')$, where Z is a maximal chain and ZY, ZY' denote the unique members of this chain in Y and in Y', respectively. Then $Y \sqcup Y'$ cannot contain two different x,y such that $x \leq y$ (since each maximal chain containing x and y may have at most one member in $Y \sqcup Y'$) and each $x \in X$ must be comparable with $max(ZY, ZY') \in Y \sqcup Y'$ for each maximal chain Z containing x. Hence $Y \sqcup Y'$ is the largest upper bound of Y and Y', as required. Similarly for the greatest lower bounds. $\Box$

A maximal chain Z as in the condition (pdf2) is called a *critical chain* from x to y. The existence of such a chain is assumed in order to guarantee that the delay between x and y such that y follows x cannot be reduced without reducing some delays between intermediate elements. That it really guarantees such a property can be derived easily from the following fact.

2.5. Proposition. If a delay function d on X satisfies (pdf2) and $Z \subseteq X$ is a critical chain from x to y then

$$d(x, y) = d(x, z_1) + d(z_1, z_2) + ... + d(z_{n-1}, z_n) + d(z_n, y)$$

for each subchain $x \leq z_1 \leq z_2 \leq ... \leq z_{n-1} \leq z_n \leq y$ of Z. $\Box$

Proof: For subchains containing 2, 3, or 4 elements the equality reduces to that in (pdf2). Assuming that it holds true for subchains containing not more than $n + 1$ elements and considering for a subchain $x \leq z_1 \leq z_2 \leq ... \leq z_{n-1} \leq z_n \leq y$ the quantities

$$\begin{aligned} l &= d(x, z_1) + ... + d(z_{i-2}, z_{i-1}) \\ m &= d(z_{i-1}, z_{i+1}) \\ u &= d(z_{i+1}, z_{i+2}) + ... + d(z_n, y) \\ L &= d(x, z_1) + ... + d(z_{i-2}, z_{i-1}) + d(z_{i-1}, z_i) \end{aligned}$$

$$\begin{aligned} m_1 &= d(z_{i-1}, z_i) \\ m_2 &= d(z_i, z_{i+1}) \\ U &= d(z_i, z_{i+1}) + d(z_{i+1}, z_{i+2}) + ... + d(z_n, y) \end{aligned}$$

we obtain $l+m_1+U = l+m+u = L+m_2+u$. Hence $m_1+U = m+u$ and $L+m_2 = m+l$, which implies $m = m_1 + m_2$ and thus $d(x, y) = l + m_1 + m_2 + u$, as required. □

2.6. Definition. Given a set V of values, a (finite) *time-consuming computation* with values from V is $\mathcal{X} = (X, d, e, s, t)$, where X is a finite set (of *tokens*), d is a proper delay function on X, e is a mapping from X to V (a *valuation* which assigns a value to each token), $s = (s_v : v \in V)$ is a family of enumerations of the sets $e^{-1}(v) \cap X_{min}$ (the *initial arrangement* of tokens), and $t = (t_v : v \in V)$ is a family of enumeration of the sets $e^{-1}(v) \cap X_{max}$ (the *terminal arrangement* of tokens). □

By an enumeration of a set we mean here a sequence of elements of this set in which each element of this set occurs exactly once. Sometimes we use subscripts, $X_{\mathcal{X}}$, $d_{\mathcal{X}}$, $e_{\mathcal{X}}$, $s_{\mathcal{X}}$, $t_{\mathcal{X}}$, in order to avoid a confusion.

Two time-consuming computations of the timed Petri net in figure 1 are shown in figure 2 in the form of graphs. Nodes represent tokens. They are labelled by the corresponding values, where each value specifies the place in which a token resides. Directed edges annotated with numbers represent delays between tokens. These delays result from the durations of transitions which are specified in the corresponding boxes. The arrows between terminal nodes illustrate the respective terminal arrangements of tokens (the initial arrangements are trivial since there are not two initial tokens with the same value). It is worth noticing that the presented time-consuming computations are essentially different in any reasonable sense even though each of them can be obtained from the other by swap operations as in [BD 87].

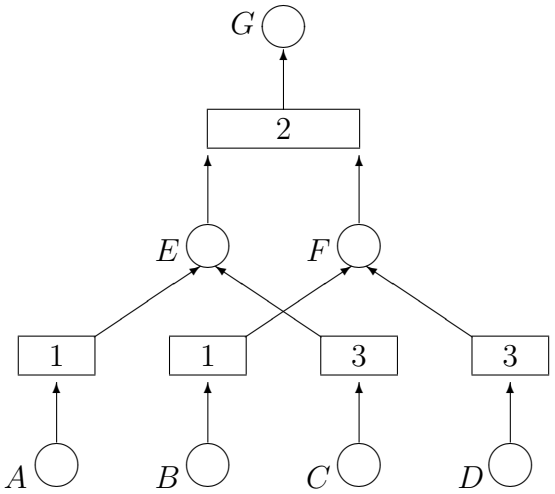

Figure 1

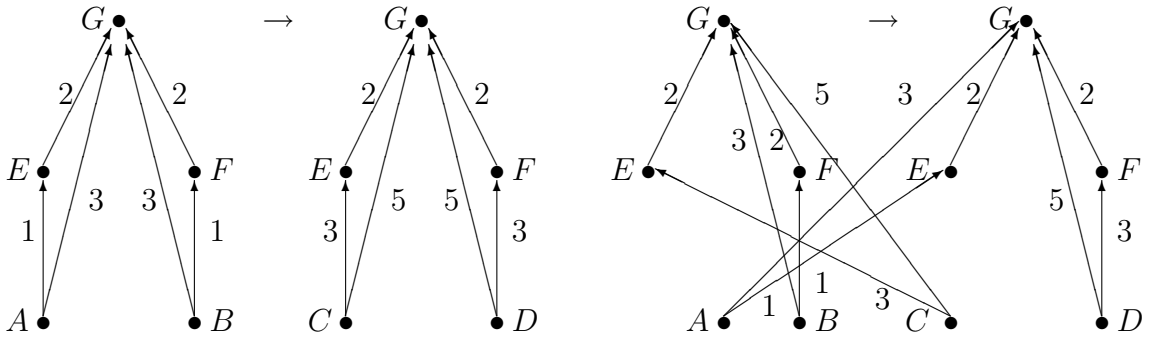

Figure 2

According to 2.6, the restriction of a time-consuming computation $\mathcal{X}$ to X_{min} with the initial and the terminal arrangements of tokens given by s, and the restriction of $\mathcal{X}$ to X_{max} with the initial and the terminal arrangements of tokens given by t, are time-consuming computations. We call them respectively the *source* and the *target* of $\mathcal{X}$ and denote by $source(\mathcal{X})$ and $target(\mathcal{X})$. If $X = X_{min} = X_{max}$ then $\mathcal{X}$ reduces to two (possibly different) arrangements of tokens belonging to X (or, equivalently, to a rearrangement of X). We call such reduced time-consuming computations *symmetries*. Symmetries with the source equal to the target are said to be *idle*. The idle symmetry with the empty set of tokens is said to be *empty*.

To each n-tuple $(\mathcal{X}_1, ..., \mathcal{X}_n)$ of idle symmetries $\mathcal{X}_i = (X_i, d_i, e_i, s_i, t_i)$ and each permutation p of the sequence $1, 2, ..., n$ there corresponds a symmetry $\mathcal{X} = (X, d, e, s, t)$, where X is a disjoint union of (suitable copies of) $X_1, ..., X_n$, $d(x, y) = d_i(x, y)$ for $x, y \in X_i$ and $d(x, y) = -\infty$ for the remaining x, y, $e(x) = e_i(x)$ for $x \in X_i$, $s_v = s_{1v}...s_{nv}$ (the concatenation of $s_{1v}, ..., s_{nv}$), and $t_v = t_{p(1)v}...t_{p(n)v}$ (the concatenation of $t_{p(1)v}, ..., t_{p(n)v}$). We denote such a symmetry by $I_p(\mathcal{X}_1, ..., \mathcal{X}_n)$ and, after [DMM 91], call symmetries of this type *interchanges*.

Being finite by definition, the time-consuming computations considered in this paper are determined uniquely by their reducts called *skeletons*.

2.7. Proposition. Each time-consuming computation $\mathcal{X} = (X, d, e, s, t)$ is determined uniquely by the structure $skeleton(\mathcal{X}) = (X, d_{red}, e, s, t)$, where d_{red} is the restriction of d to the subset $P \subseteq X \times X$ of pairs (x, y) such that $d(x, y) \neq -\infty$ and there is no z with $d(x, z) \neq -\infty$ and $d(z, y) \neq -\infty$. □

Proof: The proposition follows from the fact that P is an acyclic graph such that $x \leq y$ whenever in P there is a directed path from x to y, and such that $d(x, y)$ is the sum of the delays corresponding to the edges of such a path and this sum is maximal in the set of sums corresponding to the possible paths from x to y. □

The skeleton of a time-consuming computation may be regarded as an acyclic graph with a number $d(x, y)$ corresponding to each edge (x, y) and with the respective arrange-

ments of initial and terminal nodes. For example, for the time-consuming computations in figure 2 we have the skeletons shown in figure 3.

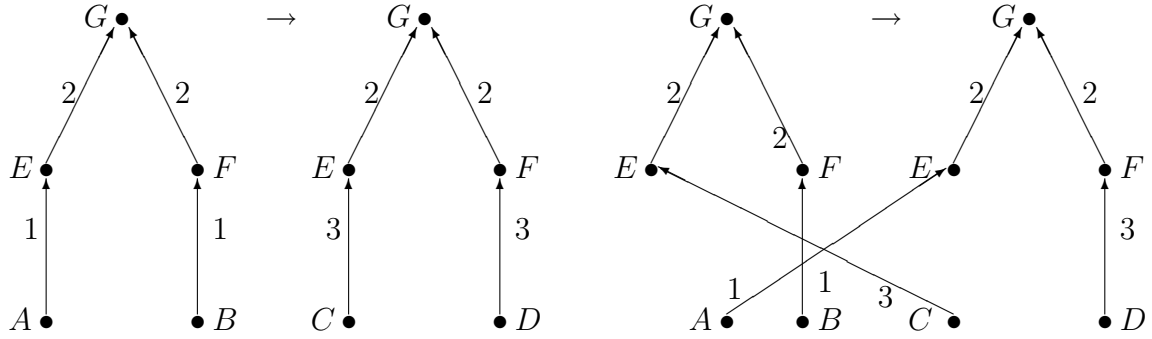

Figure 3

By ignoring in a time-consuming computation the tokens which are neither initial nor terminal we get a structure which specifies the delays with which terminal tokens are obtained from initial ones. Formally, we have the following obvious fact.

2.8. Proposition. For each time-consuming computation $\mathcal{X} = (X, d, e, s, t)$ the structure $\mathcal{X}' = (X_{min} \cup X_{max}, d', e', s, t)$, where $d' = d \mid X_{min} \times X_{max}$ and $e' = e \mid X_{min} \cup X_{max}$, is a time-consuming computation. We call it the *input-output reduct* of $\mathcal{X}$ and denote by $inout(\mathcal{X})$. □

The tokens of a time-consuming computations of the form $\mathcal{X}' = inout(\mathcal{X})$ are either minimal or maximal or both. The time-consuming computations with this property coincide with those satisfying the equality $inout(\mathcal{X}') = \mathcal{X}'$. As they specify the delays between initial and terminal tokens of computations, we call them *delay tables*.

Time-consuming computations are considered up to isomorphisms which are defined as follows.

2.9. Definition. An *isomorphism* from a time-consuming computation $\mathcal{X} = (X, d, e, s, t)$ to a time-consuming computation $\mathcal{X}' = (X', d', e', s', t')$ is a bijection $f : X \to X'$ such that $d'(f(x), f(y)) = d(x, y)$, $m'(f(x)) = m(x)$, $s'_v(i) = f(s_v(i))$, $t'_v(j) = f(t_v(j))$ for all $x, y \in X$, i not exceeding the length of s_v, j not exceeding the length of t_v, and $v \in V$. □

If such an isomorphism exists then we say that $\mathcal{X}$ and $\mathcal{X}'$ are *isomorphic* and consider them as particular instances of the same object, called an *abstract* time-consuming computation and written as $[\mathcal{X}]$ or $[\mathcal{X}']$. The source and the target of $[\mathcal{X}]$ are defined respectively as $source([\mathcal{X}]) = [source(\mathcal{X})]$ and $target([\mathcal{X}]) = [target(\mathcal{X})]$. In a similar manner we define an abstract interchange as $I_p([\mathcal{X}_1], ..., [\mathcal{X}_n]) = [I_p(\mathcal{X}_1, ..., \mathcal{X}_n)]$.

By $Act(V)$ we denote the set of abstract time-consuming computations with values from V. Each $\xi \in Atc(V)$ which is an abstract idle symmetry of the form $\xi = [\mathcal{X}]$ with

$\mathcal{X} = (X, d, e, s, t)$, $X = X_{min} = X_{max}$ and $s = t$, can be regarded as a multiset or as a formal sum $\sum(\mid s_v \mid \cdot v : v \in V) = \sum(\mid t_v \mid \cdot v : v \in V)$, where for each sequence u by $\mid u \mid$ we denote the length of this sequence. If $X = \emptyset$ then ξ can be regarded as the empty multiset 0.

Finally, by $Del(V)$ we denote the subset of those abstract time-consuming computations $\xi \in Act(V)$ which are isomorphism classes of delay tables (abstract delay tables).

3 Operations

Time-consuming computations can be composed sequentially and in parallel. The respective operations can be derived in a natural manner from the internal structure of such computations. This structure is determined by decompositions of time-consuming computations with the aid of maximal antichains and partitions into independent components. Formal definitions are as follows.

3.1. Proposition. Let $\mathcal{X} = (X, d, e, s, t)$ be a time-consuming computation, Y a maximal antichain of X, and $u = (u_v : v \in V)$ a family of enumerations of the sets $e^{-1}(v) \cap Y$. The restriction of $\mathcal{X}$ to $\downarrow Y$ with the terminal arrangement given by u is a time-consuming computation, written as $head_{Y,u}(\mathcal{X})$. Similarly for the restriction of $\mathcal{X}$ to $\uparrow Y$ with the initial arrangement given by u, written as $tail_{Y,u}(\mathcal{X})$. $\square$

Proof: It suffices to notice that Y divides maximal chains into maximal ones and critical chains into critical ones. $\square$

By $Heads_Y(\mathcal{X})$ and $Tails_Y(\mathcal{X})$ we denote respectively the set of time-consuming computations $head_{Y,u}(\mathcal{X})$ and the set of time-consuming computations $tail_{Y,u}(\mathcal{X})$.

3.2. Proposition. Let $\mathcal{A}$ and $\mathcal{B}$ be time-consuming computations such that $source(\mathcal{B})$ is isomorphic to $target(\mathcal{A})$. There exist a time-consuming computation $\mathcal{Z}$ and a maximal antichain U of $X_{\mathcal{Z}}$ such that $\mathcal{A}$ is isomorphic to a member of $Heads_U(\mathcal{Z})$ and $\mathcal{B}$ is isomorphic to a member of $Tails_U(\mathcal{Z})$. This computation is determined uniquely up to isomorphism by $\mathcal{A}$ and $\mathcal{B}$. $\square$

Proof: Due to the existence of an isomorphism f from $source(\mathcal{B})$ to $target(\mathcal{A})$ we can find isomorphic copies of $\mathcal{A}$ and $\mathcal{B}$ such that $(s_{\mathcal{B}})_v(i) = (t_{\mathcal{A}})_v(i)$ for all i and v such that either side is defined, and these are the only common elements of $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$. Then we can substitute these copies for $\mathcal{A}$ and $\mathcal{B}$ and define:

$$X_{\mathcal{Z}} = X_{\mathcal{A}} \cup X_{\mathcal{B}}$$

$$U = X_{\mathcal{A}} \cap X_{\mathcal{B}}$$

$$d_{\mathcal{Z}}(x, y) = \begin{cases} d_{\mathcal{A}}(x, y) & \text{for } x, y \in X_{\mathcal{A}} \\ d_{\mathcal{B}}(x, y) & \text{for } x, y \in X_{\mathcal{B}} \\ max(d_{\mathcal{A}}(x, u) + d_{\mathcal{B}}(u, y) : u \in U) & \text{for } x \in X_{\mathcal{A}}, y \in X_{\mathcal{B}} \\ -\infty & \text{for the remaining } x, y \in X_{\mathcal{Z}} \end{cases}$$

$$e_{\mathcal{Z}}(x) = \begin{cases} e_{\mathcal{A}}(x) & \text{for } x \in X_{\mathcal{A}} \\ e_{\mathcal{B}}(x) & \text{for } x \in X_{\mathcal{B}} \end{cases}$$

$$s_{\mathcal{Z}} = s_{\mathcal{A}}$$

$$t_{\mathcal{Z}} = t_{\mathcal{B}}.$$

It is straightforward to verify that the structure $\mathcal{Z} = (X_{\mathcal{Z}}, d_{\mathcal{Z}}, e_{\mathcal{Z}}, s_{\mathcal{Z}}, t_{\mathcal{Z}})$ is a time-consuming computation as required. □

Proposition 3.2 allows us to formulate the following definition.

3.3. Definition. Let $\alpha, \beta \in Atc(V)$ be abstract time-consuming computations such that $source(\beta) = target(\alpha)$ and let $\zeta \in Atc(V)$ be the unique abstract time-consuming computation such that α has an instance in $Heads_U(\mathcal{Z})$ and β has an instance in $Tails_U(\mathcal{Z})$ for an instance $\mathcal{Z}$ of ζ and a maximal antichain U of $X_{\mathcal{Z}}$. We call ζ the *sequential composition* of α and β and write it as $\alpha;\beta$. □

3.4. Proposition. Let $\mathcal{X} = (X, d, e, s, t)$ be a time-consuming computation and $p = (X', X'')$ a partition of X into subsets X', X'' such that:

(1) each s_v is a concatenation of an enumeration s'_v of $e^{-1}(v) \cap X_{min} \cap X'$ and an enumeration s''_v of $e^{-1}(v) \cap X_{min} \cap X''$,

(2) each t_v is a concatenation of an enumeration t'_v of $e^{-1}(v) \cap X_{max} \cap X'$ and an enumeration t''_v of $e^{-1}(v) \cap X_{max} \cap X''$,

(3) X' and X'' are *independent* in the sense that they are disjoint and x', x'' are incomparable whenever $x' \in X'$ and $x'' \in X''$.

The restriction of $\mathcal{X}$ to X' with the initial arrangement given by $s' = (s'_v : v \in V)$ and the terminal arrangement given by $t' = (t'_v : v \in V)$ is a time-consuming computation, written as $left_p(\mathcal{X})$. Similarly, the restriction of $\mathcal{X}$ to X'' with the initial arrangement given by $s'' = (s''_v : v \in V)$ and the terminal arrangement given by $t'' = (t''_v : v \in V)$ is a time-consuming computation, written as $right_p(\mathcal{X})$. □

Proof: From the construction.

A partition p as in this proposition is called a *splitting* of $\mathcal{X}$.

3.5. Proposition. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary time-consuming computations. There exist a time-consuming computation $\mathcal{Z}$ and a splitting p of $X_{\mathcal{Z}}$ such that $\mathcal{A}$ is isomorphic to $left_p(\mathcal{Z})$ and $\mathcal{B}$ is isomorphic to $right_p(\mathcal{Z})$. This computation is determined uniquely up to isomorphism by $\mathcal{A}$ and $\mathcal{B}$. □

Proof: There exist isomorphic copies of $\mathcal{A}$ and $\mathcal{B}$ such that $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are disjoint. We can substitute these copies for $\mathcal{A}$ and $\mathcal{B}$ and define:

$$X_{\mathcal{Z}} = X_{\mathcal{A}} \cup X_{\mathcal{B}}$$

$$p = (X_{\mathcal{A}}, X_{\mathcal{B}})$$

$$d_{\mathcal{Z}}(x,y) = \begin{cases} d_{\mathcal{A}}(x,y) & \text{for } x,y \in X_{\mathcal{A}} \\ d_{\mathcal{B}}(x,y) & \text{for } x,y \in X_{\mathcal{B}} \\ -\infty & \text{for the remaining } x,y \in X_{\mathcal{Z}} \end{cases}$$

$$e_{\mathcal{Z}}(x) = \begin{cases} e_{\mathcal{A}}(x) & \text{for } x \in X_{\mathcal{A}} \\ e_{\mathcal{B}}(x) & \text{for } x \in X_{\mathcal{B}} \end{cases}$$

$$(s_{\mathcal{Z}})_v = (s_{\mathcal{A}})_v (s_{\mathcal{B}})_v \text{ for all } v \in V$$

$$(t_{\mathcal{Z}})_v = (t_{\mathcal{A}})_v (t_{\mathcal{B}})_v \text{ for all } v \in V.$$

It is straightforward to verify that the structure $\mathcal{Z} = (X_{\mathcal{Z}}, d_{\mathcal{Z}}, e_{\mathcal{Z}}, s_{\mathcal{Z}}, t_{\mathcal{Z}})$ is a time-consuming computation as required. □

Due to 3.5 we can formulate the following definition.

3.6. Definition. Let $\alpha, \beta \in Atc(V)$ be arbitrary abstract time-consuming computations and let $\zeta \in Atc(V)$ be the unique abstract time-consuming computation such that $left_p(\mathcal{Z})$ is an instance of α and $right_p(\mathcal{Z})$ is an instance of β for an instance $\mathcal{Z}$ of ζ and a splitting p of $\mathcal{Z}$. We call ζ the *parallel composition* of α and β and write it as $\alpha \otimes \beta$. □

Thus, taking into account the results of sections 2 and 3, we obtain the following operations on $Atc(V)$: $() \mapsto 0$, $(\alpha) \mapsto source(\alpha)$, $(\alpha) \mapsto target(\alpha)$, $(\alpha, \beta) \mapsto \alpha;\beta$, $(\alpha, \beta) \mapsto \alpha \otimes \beta$.

Referring to these operations we can define similar operations on the set $Del(V)$ of abstract delay tables, say $() \mapsto 0'$, $(\alpha) \mapsto source'(\alpha)$, $(\alpha) \mapsto target'(\alpha)$, $(\alpha, \beta) \mapsto \alpha;'\beta$, $(\alpha, \beta) \mapsto \alpha \otimes' \beta$. It suffices to take $0' = 0$, $source'(\alpha) = source(\alpha)$, $target'(\alpha) = target(\alpha)$, $\alpha \otimes' \beta = \alpha \otimes \beta$, and define the sequential composition of two abstract delay tables α and β as $\alpha;'\beta = inout(\alpha;\beta)$.

4 Properties of operations

The introduced operations on abstract time-consuming computations enjoy properties which can be summarized as follows.

4.1. Theorem. When equipped with the operations $() \mapsto 0$, $(\alpha) \mapsto source(\alpha)$, $(\alpha) \mapsto target(\alpha)$, $(\alpha, \beta) \mapsto \alpha;\beta$, $(\alpha, \beta) \mapsto \alpha \otimes \beta$, the set $Atc(V)$ is a strict monoidal (morphisms-only) category, that is:

(1) $(Atc(V), source, target, ;)$ is a category with abstract idle symmetries playing the role of objects and their identities and each abstract time-consuming computation α playing the role of a morphism from $source(\alpha)$ to $target(\alpha)$,

(2) $(Atc(V), \otimes, 0)$ is a monoid such that $\alpha;\beta \otimes \gamma;\delta = (\alpha \otimes \gamma);(\beta \otimes \delta)$ whenever $source(\beta) = target(\alpha)$ and $source(\delta) = target(\gamma)$.

Moreover, the following *coherence axiom* is satisfied for all $\alpha_1, ..., \alpha_n$ with $source(\alpha_1) = u_1, ..., source(\alpha_n) = u_n$, $target(\alpha_1) = v_1, ..., target(\alpha_n) = v_n$ and each permutation p of the sequence $1, ..., n$:

(3) $I_p(u_1, ..., u_n); \alpha_{p(1)} \otimes ... \otimes \alpha_{p(n)} = \alpha_1 \otimes ... \otimes \alpha_n; I_p(v_1, ..., v_n)$.

Similarly for the set $Del(V)$ of abstract delay tables and operations $() \mapsto 0' = 0$, $(\alpha) \mapsto source'(\alpha) = source(\alpha)$, $(\alpha) \mapsto target'(\alpha) = target(\alpha)$, $(\alpha, \beta) \mapsto \alpha;'\beta = inout(\alpha; \beta)$, $(\alpha, \beta) \mapsto \alpha \otimes' \beta = \alpha \otimes \beta$. The correspondence $\alpha \mapsto inout(\alpha)$ is a functor and a homomorphism with respect to the parallel composition. □

Proof: The mentioned properties follow directly from the definitions of operations. □

Due to (3) the family of interchanges $I_*(u_1, u_2)$, where $*$ denotes the permutation $1 \mapsto 2$, $2 \mapsto 1$, is a natural transformation from the bifunctor $(\alpha_1, \alpha_2) \mapsto \alpha_1 \otimes \alpha_2$ to $(\alpha_1, \alpha_2) \mapsto \alpha_2 \otimes \alpha_1$ such that:

$$I_*(u_1, u_2); I_*(u_2, u_1) = u_1 \otimes u_2$$

$$(I_*(u_1, u_2) \otimes u_3); (u_2 \otimes I_*(u_1, u_3)) = I_*(u_1, u_2 \otimes u_3).$$

Consequently, the considered strict monoidal category is symmetric.

Now we shall show that each abstract time-consuming computation can be obtained by composing sequentially and in parallel computations which cannot further be decomposed (atoms) and symmetries. To this end we exploit a particular property of finite, K-dense, partially ordered sets (cf. [W 80]).

4.2. Proposition. Given a time-consuming computation $\mathcal{X} = (X, d, e, s, t)$, there exists a set Z of subsets of X such that:

(1) each $z \in Z$ ordered by the restriction of the order induced by the delay function consists of two disjoint subsets: a subset z_{min} of minimal elements and a subset z_{max} of maximal elements, each minimal element comparable with each maximal element and vice-versa,

(2) each $x \in X - (X_{min} \cup X_{max})$ belongs to exactly two $z, z' \in Z$ and then $z \neq z'$ and either $x \in z_{max} \cap z'_{min}$ or $x \in z'_{max} \cap z_{min}$,

(3) each $x \in X_{min} - X_{max}$ belongs to exactly one $z \in Z$ and then $x \in z_{min}$,

(4) each $x \in X_{max} - X_{min}$ belongs to exactly one $z' \in Z$ and then $x \in z'_{max}$,

(5) there is no $z \in Z$ containing elements of $X_{min} \cap X_{max}$. □

Proof: Let $Y_0 \sqsubseteq Y_1 \sqsubseteq ... \sqsubseteq Y_{n-1} \sqsubseteq Y_n$ be a maximal chain of maximal antichains of X. We define Z as the set of subsets of the form $(Y_i - Y_{i+1}) \cup (Y_{i+1} - Y_i)$. Due to the maximality of this chain each $x \in Y_i - Y_{i+1}$ is comparable with each $y \in Y_{i+1} - Y_i$ (since otherwise between Y_i and Y_{i+1} there would be a maximal antichain containing x and y and it would

be different from Y_i and Y_{i+1}). Hence (1) holds true. The remaining properties follow directly from the construction. □

The proposition just proved suggests a decomposition of a time-consuming computation $\mathcal{X}$ into components corresponding to restrictions of $\mathcal{X}$ to members of Z or to one-element sets. Such components can be defined formally as follows.

4.3. Definition. A time-consuming computation $\mathcal{X} = (X, d, e, s, t)$ and the corresponding abstract time-consuming computation $[\mathcal{X}]$ are said to be *prime* if $X = X_{min} \cup X_{max}$, $X_{min} \cap X_{max} = \emptyset$, and every $x \in X_{min}$ is comparable with every $y \in X_{max}$ and vice-versa. Similarly, a time-consuming computation is said to be a *one-element* computation if X is a one-element set. Finally, the prime and one-element computations are said to be *elementary.* □

This definition implies that elementary time-consuming computations are indecomposable and thus atomic.

4.4. Theorem. Each time-consuming computation can be obtained by composing sequentially and in parallel symmetries and elementary computations. □

Proof: Let $\mathcal{X} = (X, d, e, s, t)$ be a time-consuming computation, $Y_0 \sqsubseteq Y_1 \sqsubseteq ... \sqsubseteq Y_{n-1} \sqsubseteq Y_n$ a maximal chain of maximal antichains of X and Z the set of subsets of X as in 4.2. There exists a symmetry π_1 rearranging the initial tokens such that for each $v \in V$ the members of $e^{-1}(v) \cap Y_0 \cap Y_1$ precede those of $e^{-1}(v) \cap (Y_0 - Y_1)$ and the orders of members in $e^{-1}(v) \cap Y_0 \cap Y_1$ are consistent with an enumeration $y_{11}y_{12}...y_{1i_1}$ of $Y_0 \cap Y_1$. Besides, there exists an arrangement t_1 of tokens in Y_1 which is identical with the terminal arrangement of π_1 in $Y_0 \cap Y_1$ and such that for each $v \in V$ the members of $e^{-1}(v) \cap Y_0 \cap Y_1$ precede those of $e^{-1}(v) \cap (Y_1 - Y_0)$. The restriction of $\mathcal{X}$ to $\uparrow Y_0 \cap \downarrow Y_1$ with the initial arrangement given by the terminal arrangement of π_1 and the terminal arrangement given by t_1 is a time-consuming computation $\mathcal{X}_1$. Now $\xi_1 = [\mathcal{X}_1]$ can be represented as the parallel composition

$$\xi_1 = \sigma_{11} \otimes ... \otimes \sigma_{1i_1} \otimes \tau_1$$

where σ_{11},...,σ_{1i_1},τ_1 correspond to the respective restrictions of $\mathcal{X}$ to the subsets $\{y_{11}\}$,..., $\{y_{1i_1}\}$, $(Y_0 - Y_1) \cup (Y_1 - Y_0)$. Thus we obtain a decomposition of ξ_1 into the parallel composition of one-element computations σ_{11},...,σ_{1i_1} and of a prime computation τ_1. Similarly, for Y_1, Y_2 we can define a symmetry rearranging the terminal tokens of $\mathcal{X}_1$, the corresponding restriction $\mathcal{X}_2$ of $\mathcal{X}$, and a representation of $\xi_2 = [\mathcal{X}_2]$ as an appropriate parallel composition

$$\xi_2 = \sigma_{21} \otimes ... \otimes \sigma_{2i_2} \otimes \tau_2,$$

and so on until reaching

$$\xi_n = \sigma_{n1} \otimes ... \otimes \sigma_{ni_n} \otimes \tau_n.$$

Finally, we define π_{n+1} as the symmetry rearranging the terminal tokens of ξ_n to the terminal arrangement of $\mathcal{X}$.

Thus we obtain a sequence

$$\pi_1, \xi_1, \pi_2, \xi_2, ..., \pi_n, \xi_n, \pi_{n+1}$$

such that

$$\pi_1; \xi_1; \pi_2; \xi_2; ...; \pi_n; \xi_n; \pi_{n+1}$$

is defined and equal to $[\mathcal{X}]$ as required. □

In general, there are many ways of representing a time-consuming computation as a result of composing sequentially and in parallel its components (atomic or not). Each way corresponds to an expression which is built of symbols of component computations and symbols of operations. However, following [DMM 91], we can show that every two expressions which represent the same time-consuming computation in terms of its atomic components can be proved equivalent w.r. to the equivalence generated by the axioms of strict monoidal categories and the coherence axiom.

Formally, we assume that the elementary time-consuming computations belonging to $Atc(V)$ are represented by some symbols, where the symbols of one-element computations are arranged into a totally ordered set S, and the symbols of prime computations are arranged into a totally ordered set T. Then we define Ex, the set of *expressions*, as the lest set which contains S, T, (symbols of) the symmetries belonging to $Atc(V)$, and strings $\xi; \xi'$ and $\xi \otimes \xi'$ with $\xi \in Ex$ and $\xi' \in Ex$ (in order to avoid a confusion also brackets may be used and a priority of ; over $\otimes$ may be assumed).

Each expression $\xi \in Ex$ represents an abstract time-consuming computation, called the *value* of ξ. For a symbol of a one-element computation or a prime computation or a symmetry this value is just the abstract time-consuming computation represented by this symbol. For an expression of the form $\xi'; \xi''$ or $\xi' \otimes \xi''$ the value of ξ is respectively the sequential or the parallel composition of the values of ξ' and ξ''. Conversely, for the decomposition of an expression $\xi = \xi'; \xi''$ or $\xi = \xi' \otimes \xi''$ into its subexpressions ξ' and ξ'' we have a unique decomposition of the represented time-consuming computation into parts represented by ξ' and ξ''. Thus a uniquely determined part of a time-consuming computation $\mathcal{X}$ there corresponds to each occurrence φ of an expression η in an expression ξ which represents $\mathcal{X}$. We call such a part the *instance* of η in $\mathcal{X}$ for φ.

Finally, the equivalence of expressions in Ex is defined as the least equivalence relation $\sim$ such that $\xi \sim \xi'$ whenever it follows from the axioms of strict monoidal categories and the coherence axiom that the morphisms represented by ξ and ξ' coincide.

The provability of equivalence of expressions representing the same time-consuming computation can be shown with the aid of maximally parallel representations of time-consuming computations.

4.5. Proposition. By applying the axioms of strict monoidal categories and the coherence axiom one can reduce each expression from Ex to a *maximally parallel* form:

$$\pi_1; \sigma_{11} \otimes ... \otimes \sigma_{1m_1} \otimes \tau_{11} \otimes ... \otimes \tau_{1n_1}; ...; \pi_r; \sigma_{r1} \otimes ... \otimes \sigma_{rm_r} \otimes \tau_{r1} \otimes ... \otimes \tau_{rn_r}; \pi_{r+1}$$

where π_i denote symmetries, σ_{ij} denote one-element computations, τ_{ik} denote prime computations,and the following conditions are satisfied:

(1) in each sequential component of the form

$$\xi_i = \sigma_{i1} \otimes ... \otimes \sigma_{im_i} \otimes \tau_{i1} \otimes ... \otimes \tau_{in_i}$$

the order of one-element parallel components respects the order in S and the order of prime parallel components respects the order in T,

(2) there is no symmetry ϱ_i, $i \in \{2, ..., r\}$, from $target(\xi_{i-1})$ to $source(\xi_i)$ such that all the initial tokens of the instance of some τ_{ik} in a time-consuming computation $\mathcal{X}_i$ represented by $\xi_{i-1}; \varrho_i; \xi_i$ can be found among the tokens of instances of $\sigma_{i-1,1}, ..., \sigma_{i-1,m_{i-1}}$ in $\mathcal{X}_i$. □

Proof: The proposition follows from the fact that if there exist ϱ_i and τ_{ik} which contradict to (2) then one can move τ_{ik} from ξ_i to ξ_{i-1} by composing ξ_{i-1} with a symmetry π'_i and the expression

$$\sigma_{i1} \otimes ... \otimes \sigma_{im_i} \otimes \tau'_{i1} \otimes ... \otimes \tau'_{in_i}$$

where $\tau'_{ik} = source(\tau_{ik})$ and $\tau'_{ij} = \tau_{ij}$ for the remaining j, and by arranging the components of the computation thus obtained according to the orders in S and T. □

The result about provability of the fact that two expressions represent the same abstract time-consuming computation is as follows.

4.6. Theorem. If ξ and ξ' are two expressions representing the same abstract time-consuming computation then these two expressions are equivalent, that is $\xi \sim \xi'$. □

Proof: Due to 4.5 we may assume without a loss of generality that ξ and ξ' are maximally parallel representations of an abstract time-consuming compotation $[\mathcal{X}]$. Moreover, the sequential components of ξ and ξ' have the same instances $\mathcal{X}_i$ in $\mathcal{X}$. Consequently,

$$\xi = \pi_1; \xi_1; ...; \pi_r; \xi_r; \pi_{r+1}$$

and

$$\xi' = \pi'_1; \xi_1; ...; \pi'_r; \xi_r; \pi'_{r+1},$$

where ξ_i denote $[\mathcal{X}_i]$, π_i, π'_i denote symmetries, and we have to show that these two representations of $[\mathcal{X}]$ are equivalent. The proof is by induction on the number of sequential components.

For $r = 1$ we have to prove that $\pi; \xi; \varrho$ and $\pi'; \xi; \varrho'$ are equivalent whenever they represent the same abstract time-consuming computation $[\mathcal{X}]$. We replace $\pi; \xi; \varrho$ by $\pi^{-1}\pi; \xi; \varrho\varrho^{-1} = \xi$ and in $\pi'; \xi; \varrho'$ we substitute $\pi^{-1}; \pi'$ for π' and $\varrho'; \varrho^{-1}$ for ϱ'. Then we notice that the equality of values of ξ and $\pi'; \xi; \varrho'$ implies that π' and ϱ' are mutually inverse interchanges corresponding to a permutation of the set of parallel components of ξ as in the coherence axiom. Thus $\pi; \xi; \varrho$ and $\pi'; \xi; \varrho'$ are equivalent.

In the case of $r > 1$ we define:

$$\eta = \pi_1; \xi_1; ...; \pi_{r-1}; \xi_{r-1}; \pi_r$$

$$\eta' = \pi'_1; \xi_1; ...; \pi'_{r-1}; \xi_{r-1}; \pi'_r$$

$$\zeta = \xi_r; \pi_{r+1}$$

$$\zeta' = \xi_r; \pi'_{r+1}$$

and find a symmetry π such that $\eta; \pi$ and $\eta'; \pi$ represent the same abstract time-consuming computation and $\pi^{-1}; \zeta$ and $\pi^{-1}; \zeta'$ represent the same abstract time-consuming computation. Then $\eta; \pi$ and $\eta'; \pi$ are equivalent by inductive hypothesis and $\pi^{-1}; \zeta$ and $\pi^{-1}; \zeta'$ are equivalent as in the case $r = 1$. □

4.7. Corollary. The input-output reduct of a time-consuming computation given by an expression ξ can be obtained by composing the input-output reducts of elementary time-consuming computations occurring in ξ with the aid of operations on delay tables corresponding to those on time-consuming computations as specified by ξ. $\square$

5 Applications to timed Petri nets

The monoidal category $Atc(V)$ has been defined without relating the time-consuming computations which are its members to any particular system. In the case of a concrete system we have to do with a concrete set V of values and with a concrete part of $Atc(V)$. In particular, for timed Petri nets V must be chosen such that the positions of tokens and the data they carry can be represented, and computations must be constructed from components corresponding to presences of tokens in places, rearrangements of tokens, and executions of transitions. The way of constructing computations from their components is always the same: by composing components sequentially and in parallel. Only the criteria of qualifying the results of constructions as computations of a given net depend on the type of this net.

For a timed Petri net N without a distinguished initial marking, where $N = (P, A, F, \delta)$ with a set P of unbounded places, a set A of transitions, a flow relation $F \subseteq P \times A \cup A \times P$, and a function δ which assigns durations to transitions, computations can be defined as arbitrary compositions of symmetries, one-element computations corresponding to presences of tokens in places of N, and prime computations corresponding to executions of transitions of N (cf. figures 1 and 2). It suffices to take for values all $p \in P$, where each value represents the presence of a token in the respective place p, and use symmetries which depend on the net through these values only. The presence of a token x in a place p can be defined as a one-element computation $\hat{p}$ such that:

$$X_{\hat{p}} = \{x\}$$

$$d_{\hat{p}}(x, x) = 0$$

$$e_{\hat{p}}(x) = p$$

$$(s_{\hat{p}})_p = (t_{\hat{p}})_p = x$$

$$(s_{\hat{p}})_q = (t_{\hat{p}})_q = \emptyset \text{ for } q \neq p.$$

An execution of a transition a can be represented as a prime computation $\hat{a}$ such that:

$$X_{\hat{a}} = X' \cup X'',$$

where X', X'' are disjoint and such that there exist bijections $f : X' \to Ft$ and $g : X'' \to tF$,

$$d_{\hat{a}}(x, y) = \begin{cases} 0 & \text{for } x = y \\ \delta(a) & \text{for } x \in X' \quad \text{and } y \in X'' \\ -\infty & \text{for the remaining } x, y \in X_{\hat{a}} \end{cases}$$

$$e_{\hat{a}}(x) = \begin{cases} p & \text{for } x \in X' \quad \text{with} \quad f(x) = p \\ q & \text{for } x \in X'' \quad \text{with} \quad g(x) = q \end{cases}$$

$(s_{\hat{a}})_p$ is any enumeration of the subset $\{x \in X' : f(x) = p\}$, and $(t_{\hat{a}})_q$ is any enumeration of $\{x \in X'' : g(x) = q\}$.

Any computation of N can be represented as a result of composing symmetries and one-element and prime computations (cf. 4.4). Conversely, each result of composing elementary computations of N and symmetries rearranging tokens of such computations is a computation of N since no restrictions are imposed on the coexistence of tokens in places of N and on the concurrency of executions of transitions of N.

In the case of nets with places of finite capacities only those compositions of elementary computations and symmetries can be taken into account in which each maximal antichain has in each place a number of tokens that does not exceed the capacity of this place. Similarly for nets with other restrictions of this type.

For a timed Petri net N with an initial marking μ, where $N = (P, A, F, \delta)$ as before and $\mu : P \times (-\infty, +\infty) \to \{0, 1, ...\}$ specifies the numbers of tokens appearing in given places at given instants of time, a typical problem is to see how the possible time-consuming computations originated by μ are displayed on a time scale. The respective descriptions, called *timed computations*, can be obtained by finding for each time-consuming computation ξ thus originated a function which assigns the corresponding appearance times to its tokens.

More precisely, a time-consuming computation ξ of N is said to be *originated* by μ if there exist an instance $\mathcal{X} = (X, d, e, s, t) \in \xi$ and a function $\vartheta_0 : X_{min} \to (-\infty, +\infty)$ such that, for each $p \in P$ and each $u \in (-\infty, +\infty)$, $\mu(p, u)$ is the number of tokens $x \in X_{min}$ with $e(x) = p$ and $\vartheta_0 = u$. For such a computation ξ we define the timed computation corresponding to μ, ξ, ϑ_0 as $(\mathcal{X}, \vartheta)$, where ϑ is the function assigning to each $x \in X$ the appearance time given by the formula:

$$\vartheta(x) = max(\vartheta_0(m) + d(m, x) : m \in X_{min}).$$

Timed computations may be considered up to isomorphisms which preserve their structure, that is as *abstract* timed computations.

It is important to realize that not all potential timed computations originated by a marking μ have a chance to be executed. This phenomenon may occur when some solutions of conflicts between transitions become possible later than some others and thus are excluded by an earlier choice. In particular, it is a consequence of the principle "first enabled - first chosen" which we usually assume implicitly when dealing with timed systems.

The timed computations which can really be executed starting from μ can be characterized as those members of the set of all potential timed computations originated by μ which cannot be *dominated* by any other member of this set, where ζ is said to be dominated by ζ' if there exist a timed computation ζ'' originated by μ and instances $(\mathcal{Z}, \lambda) \in \zeta$, $(\mathcal{Z}', \lambda') \in \zeta'$, $(\mathcal{Z}'', \lambda'') \in \zeta''$, such that $X_{\mathcal{Z}''} = X_{\mathcal{Z}} \cap X_{\mathcal{Z}'}$, $\lambda'' = \lambda \mid X_{\mathcal{Z}''} = \lambda' \mid X_{\mathcal{Z}''}$, and in $X_{\mathcal{Z}'} - X_{\mathcal{Z}''}$ there is a token x' with the appearance time $\lambda'(x')$ earlier than the appearance times $\lambda(x)$ of all tokens in $X_{\mathcal{Z}} - X_{\mathcal{Z}''}$. For example, for the initial marking which consists of single tokens appearing at the time instant 0 in A, B, C, D, the timed computation corresponding to the first time-consuming computation in figure 2 dominates that corresponding to the second one since it produces one of the tokens in G earlier.

Note that the possibility of having a timed computation executed is a property which depends on the existence of other timed computations which could dominate it.

References

[BD 87] Best, E., Devillers, R., *Sequential and Concurrent Behaviour in Petri Net Theory*, Theoretical Computer Science, 55, 1987, pp.87-136

[DMM 89] Degano, P., Meseguer, J., Montanari, U., *Axiomatizing Net Computations and Processes*, in the Proceedings of 4th LICS Symposium, IEEE, 1989, pp.175-185

[DMM 91] Degano, P., Meseguer, J., Montanari, U., *Axiomatizing the Algebra of Net Computations and Processes*, Università degli Studi di Pisa, Dipartimento di Informatica, Technical Report: TR-1/91

[W 80] Winkowski, J., *Behaviours of Concurrent Systems*, Theoretical Computer Science, 12, 1980, pp. 39-60

[W 82] Winkowski, J., *An Algebraic Description of System Behaviours*, Theoretical Computer Science, 21, 1982, pp.315-340

[W 92] Winkowski, J., *An Algebra of Time-Consuming Computations*, Institute of Computer Science of the Polish Academy of Sciences, Technical Report 722, December 1992