Chapter 7

Process Petri Nets with Time Stamps and Their Using in Project Management

Ivo Martiník

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76769

Abstract

Process Petri nets with time stamps (PPNTS) are the newly introduced class of low-level Petri nets, whose definition and the properties are the main topic of this chapter; they generalize the properties of Petri net processes in the area of design, modeling and verification of generally parallel systems with the discrete time. Property-preserving Petri net process algebras (PPPAs) were originally designed for the specification and verification of manufacturing systems. PPPA does not need to verify composition of Petri net processes because all their algebraic operators preserve the specified set of the properties. These original PPPAs are generalized for the class of the PPNTSs in this chapter. The new COMP, SYNC and JOIN algebraic operators are defined for the class of PPNTS and their chosen properties are proved. With the support of these operators, the PPNTSs can be extended also to the areas of project management and the determination of the project critical path with the support of the critical path method (CPM). The new CPNET subclass of PPNTS class is defined in this chapter. It is specially designed for the generalization of the CPM activity charts and their properties. This fact is then demonstrated on the simple project example and its critical path and other property specifications.

Keywords: process Petri nets with time stamps, property-preserving Petri net process algebras, critical path method, discrete time, property preservation, parallel systems modeling

1. Introduction

There are currently a number of formally defined classes of **Petri nets** [1, 2] available for modeling of generally parallel systems. When studying distributed parallel programming systems, real-time systems, economic systems and many other types of systems, it plays a role modeling of the time variables associated with individual system events, the duration of the

studied activities, the time history of the modeled system and many other time characteristics. Special classes of Petri nets were introduced for the modeling of these types of systems with discrete time and their properties were studied in detail. **Time Petri nets** and **timed Petri nets** [3, 4] are currently the two most important classes of low-level Petri nets that use the concept of discrete time in their definition. Other classes of low-level Petri nets with discrete time are introduced and discussed for instance in [5–7]. It can be stated that most of the currently studied classes of Petri nets with discrete time use only the relative time variables usually related to the specific marking of the given Petri net. This fact can then cause difficulties, for example, in modeling complex time-synchronized distributed systems in which an external time source is usually available and individual components of this system must be synchronized with this external time source.

Process Petri nets (PPN) [8] were primarily introduced as a special subclass of classic low-level Petri nets for their using in the area of workflow management. PPN is a continuous Petri net that include within the set of all its places the unique input place, the unique output place and a finite set of so-called resource places which may contain, along with the input place, the tokens in the entry marking of the given PPN. These tokens located in the entry marking of PPN at the resource places usually represent the permanent resources of the modeled system. The given PPN can pass into its exit marking that is reachable from its entry marking by performing the final sequence of the transition firings. The tokens of the PPN's exit marking may be then located only at its single output place and also at its resource places.

Process Petri nets with time stamps (PPNTS) are the newly introduced class of low-level Petri nets whose definition and the properties are the main topics of this chapter. PPNTS generalize the properties of PPNs in the area of design, modeling and verification of generally parallel systems with the discrete time.

Property-preserving Petri net process algebras (PPPA) [9] were originally designed for the specification and verification of manufacturing systems. PPPA has four types of operators: extensions, compositions, refinements and reductions. All operators can preserve about 20 PPN's properties (some of them under additional conditions), such as liveness, boundedness, reversibility, RC-property, traps, siphons, proper termination, and so on. PPPA does not need to verify composition of PPNs because all their algebraic operators preserve the specified set of the properties. Hence, if the source PPNs satisfy the desirable properties, each of the composite PPN, including the PPN that models the resulting system itself, also satisfies these properties. These original PPPA are generalized for the class of the PPNTS in this chapter and their properties of proper-formed, well-formed and pure-formed PPNTS are then newly introduced. The new **COMP**, **SYNC** and **JOIN** algebraic operators are defined for the class of PPNTS and their chosen properties are proved.

With the support of these operators, the PPNTS can be extended also to the areas of the project management and the determination of the project critical path with the support of the critical path method (CPM) [10]. The new CPPNET subclass of PPNTS class is then defined in this chapter to represent pure-formed time-dependent processes. It is specially designed for the generalization of the CPM activities charts and their properties. This fact is then demonstrated on the simple project example and its critical path and other properties specification.

http://dx.doi.org/10.5772/intechopen.76769

This chapter is arranged into the following sections: Section 2 explains the base term of this chapter, that is, process Petri nets with time stamps and introduces the terms of proper-formed, well-formed and pure-formed PPNTS; Section 3 discusses algebraic operators **COMP** and **SYNC** defined over the class of PPNTS and their main properties; Section 4 then introduces the special subclass CPPNET of the PPNTS class and explains its use in the area of the project management to represent pure-formed time-dependent processes with using of PPNTSs and to find critical paths for these processes similarly as in the case of the well-known critical path method (CPM). Finally, Section 5 gives the conclusions of the research to conclude the chapter.

2. Process Petri nets with time stamps and their properties

Let $\boldsymbol{N}$ denote the set of all natural numbers, $\boldsymbol{N} := \{1, 2, \ldots\}$; $\boldsymbol{N}_0$ the set of all non-negative integer numbers, $\boldsymbol{N}_0 := \{0, 1, 2, \ldots\}$; $\varnothing$ the empty set; $|A|$ the cardinality of the given set A; $\mathscr{P}(A)$ denotes the family of all the subsets of the given set A; f: $A \to B$ a function on a domain A to a codomain B; $\vdash$ the logical negation operator. Let $(A \subset \boldsymbol{N}_0) \wedge (\exists n \in \boldsymbol{N}: |A| = n) \wedge (A \neq \varnothing)$; then $max(A) := x$, where $(x \in A) \wedge (\forall y \in A: x \geq y)$. **Multiset** M over a nonempty set S is a function M: $S \to \boldsymbol{N}_0$. The non-negative number $M(a) \in \boldsymbol{N}_0$, where $a \in S$, denotes the number of occurrences of the element a in the multiset M. The multiset M over a nonempty set S will be represented by the notation $M := [a^{M(a)}, b^{M(b)}, c^{M(c)}, \ldots] = [a, \ldots, a, b, \ldots, b, c, \ldots, c, \ldots]$, where $S := \{a, b, c, \ldots\}$. Notation S_{MS} then denotes the class of all the multisets over the set S.

Definition 1. Let A be a nonempty set. By the (nonempty finite) **sequence** σ over the set A we understand a function σ: $\{1, 2, \ldots, n\} \to A$, where $n \in \boldsymbol{N}$. Function ε: $\varnothing \to A$ is called the **empty sequence** on the set A. We usually represent the sequence σ: $\{1, 2, \ldots, n\} \to A$ by the notation $\sigma = <a_1, a_2, \ldots, a_n>$ of the elements of the set A, where $a_i = \sigma(i)$ for $1 \leq i \leq n$. Empty sequence ε: $\varnothing \to A$ on the set A we usually represent by the notation $\varepsilon = <>$. We denote the set of all finite (and possible empty) sequences over the set A by the notation A_{SQ}.

If $\sigma = <a_1, a_2, \ldots, a_n>$ and $\tau = <b_1, b_2, \ldots, b_m>$ are the finite sequences, where $\sigma \in A_{SQ}$, $\tau \in A_{SQ}$, $n \in \boldsymbol{N}$, $m \in \boldsymbol{N}$, then by the **concatenation of the sequences** σ and τ, denoted by $\sigma{+}{+}\tau$, we understand the finite sequence $\sigma{+}{+}\tau := < a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m>$. The following functions are defined:

i. *length*: $A_{SQ} \to \boldsymbol{N}_0$, so that: $length(\sigma) := n$, $length(\varepsilon) := 0$,

ii. *elements*: $A_{SQ} \to \mathscr{P}(A)$, so that: $elements(\sigma) := \{a \mid \exists i, 1 \leq i \leq n: a = \sigma(i)\}$, $elements(\varepsilon) := \varnothing$,

iii. *prefix*: $A_{SQ} \times \boldsymbol{N}_0 \to A_{SQ}$, so that:

$prefix(<a_1, a_2, \ldots, a_n>, m) := < a_1, a_2, \ldots, a_m>$, if $m \leq n$,

$prefix(<a_1, a_2, \ldots, a_n>, m) := < a_1, a_2, \ldots, a_n>$, if $m > n$,

$prefix(\varepsilon, m) := \varepsilon$,

iv. *suffix*: $A_{SQ} \times \mathbf{N}_0 \rightarrow A_{SQ}$, so that:

$suffix(<a1, a2, \ldots, a_n>, m) := < a_{m+1}, a_{m+2}, \ldots, a_n>$, *if* $m < n$,

$suffix(<a1, a2, \ldots, a_n>, m) := \varepsilon$, *if* $m \geq n$,

$suffix(\varepsilon, m) := \varepsilon$,

v. *create*: $\mathbf{N} \times A \rightarrow A_{SQ}$, so that: $create(n, a) := < a, a, \ldots, a>$, where $length(<a, a, \ldots, a>) = n$,

vi. *sort*: $(\mathbf{N}_0)_{SQ} \rightarrow (\mathbf{N}_0)_{SQ}$, so that: $sort(\sigma) := \rho$,

where $(\rho = <b_1, b_2, \ldots, b_n>) \wedge (b_1 \leq b_2 \leq \ldots \leq b_n) \wedge ([a_1, a_2, \ldots, a_n] = [b_1, b_2, \ldots, b_n])$.

We use the following subsets of the set $(\mathbf{N}_0)_{SQ}$:

- $\mathbf{N}^{\#} := \{\sigma \in (\mathbf{N}_0)_{SQ} \mid (\sigma = \varepsilon) \vee ((\sigma = <a_1, a_2, \ldots, a_n>) \wedge (a_1 \leq a_2 \leq \ldots \leq a_n)), n \in \mathbf{N}\}$,
- $\mathbf{N}^0 := \{\sigma \in (\mathbf{N}_0)_{SQ} \mid (\sigma = \varepsilon) \vee ((\sigma = <0, 0, \ldots, 0>) \wedge (length(\sigma) = n)), n \in \mathbf{N}\}$.

Thus, the elements of the set $\mathbf{N}^{\#}$ constitute the empty sequence ε and all the finite ascending ordered sequences σ consisting of non-negative integer numbers. Similarly, the elements of the set $\mathbf{N}^0$ then form an empty sequence ε and all the sequences in the form <0, 0, …, 0 > of any finite length.

Definition 2. Net *NET* is an ordered triple $NET := (P, T, A)$, where P is finite nonempty set of **places**, T is finite set of **transitions**, $P \cap T = \varnothing$, and A is finite set of **arcs**, $A \subseteq (P \times T) \cup (T \times P)$. □

The given net *NET* is then described with a bipartite graph containing a finite nonempty set P of places used for expressing of the conditions of a modeled process (we usually use circles for their representation), a finite set T of transitions describing the changes in the modeled process (we usually draw them in the form of rectangles) and a finite set A of arcs being principally oriented while connecting the place with the transition or the transition with the place and we usually draw them as lines with arrows.

Some commonly used notations for the nets are $\bullet y = \{x \mid (x, y) \in A\}$ for the **preset** and $y\bullet = \{x \mid (y, x) \in A\}$ for the **postset** of a net node y (i.e., place or transition). A **path** of a net $NET := (P, T, A)$ is a nonempty sequence $<x_1, \ldots, x_k>$ of net nodes, where $k \in \mathbf{N}$, which satisfies $(x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k) \in A$. A path of the net $NET := (P, T, A)$ leading from its node x to its node y is a **circuit** if $(y, x) \in A$. We denote the set of all the circuits of the net *NET* by $\boldsymbol{CIRCUITS}_{NET}$. Net $NET' := (P', T', A')$ is a **subnet** of the net $NET := (P, T, A)$ if $(P' \subseteq P) \wedge (T' \subseteq T) \wedge (A' = A \cap ((P' \times T') \cup (T' \times P')))$. Net *NET* is **connected** if and only if it is not composed of two or more disjoint and nonempty subnets.

Definition 3. Process net with time stamps (PNTS) *PNTS* is an ordered tuple $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$, where

i. (P, T, A) is the **connected net**, $\forall t \in T$: $(\bullet t \neq \varnothing) \wedge (t\bullet \neq \varnothing)$,

ii. AF: $(P \times T) \cup (T \times P) \rightarrow \mathbf{N}_0$ is the **arc function**,

$AF(x, y) > 0 \Leftrightarrow (x, y) \in A$, $AF(x, y) = 0 \Leftrightarrow (x, y) \notin A$, *where* $x, y \in P \cup T$,

http://dx.doi.org/10.5772/intechopen.76769

iii. TP is the **transition priority function**, TP: $T \rightarrow \mathbf{N}$,

iv. TI: $(T \times P) \rightarrow \mathbf{N}_0$ is the **time interval function**,

v. IP is the **input place**, $(IP \in P) \wedge (\bullet IP = \emptyset) \wedge (\forall p \in (P \setminus (\{IP\} \cup RP))$: $\bullet p \neq \emptyset)$,

vi. OP is the **output place**, $(OP \in P) \wedge (OP\bullet = \emptyset) \wedge (\forall p \in (P \setminus (\{OP\} \cup RP))$: $p\bullet \neq \emptyset)$,

vii. RP is the set of **resource places**, $(RP \subseteq (P \setminus \{IP, OP\})) \wedge (\forall t \in T$: $\neg(\bullet t \subseteq RP))$.

The class of all the PNTS will be denoted by ***PNTS***. □

The given PNTS $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ is represented by the **connected net** (P, T, A) with nonempty preset and postset of each of its transitions t; the **arc function** AF assigning each arc with a natural number (such number has the default value of 1, if not explicitly indicated in the PNTS diagram) expressing the number of removed or added tokens from or to the place associated with that arc when firing a particular transition; **transition priority function** TP assigns with each transition the natural number value expressing its priority (with the default value of 1); the **time interval function** TI assigns to each arc of the type (*transition*, *place*) a non-negative integer d expressing the minimum time interval during which the token has to remain in the *place* instead of being able to participate in the next firing of some transition and it thus determines the so-called **time marking** of the given PNTS (the value d associated with the respective arc is given in the format $+d$ in the PNTS diagram); the **input place** IP is the only one nonresource place of PNTS $PNTS$ with no input arc(s); the **output place** OP is the only one nonresource place of PNTS $PNTS$ with no output arc(s); the finite set RP of **resource places** is used for expressing conditions of a modeled process containing some initial resources and we use circles with the double line for their representation.

Definition 4. Let $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ be the PNTS. Then:

i. **marking** M of the PNTS $PNTS$ is a function M: $P \rightarrow \mathbf{N}_0$,

ii. **time marking** m of the PNTS $PNTS$ is a function m: $P \rightarrow \mathbf{N}^{\#}$,

where $\forall p \in P$: $|M(p)| = length(m(p))$,

iii. variable $\tau \in \mathbf{N}_0$ is the **net time** of the PNTS $PNTS$,

iv. **state** S of the PNTS $PNTS$ is an ordered triple $S := (M, m, \tau)$,

v. transition $t \in T$ is **enabled** in the state $S := (M, m, \tau)$ of the PNTS $PNTS$ that is denoted by t ***en*** S, if $\forall p \in \bullet t$: $(M(p) \geq AF(p, t)) \wedge (\forall n \in elements(prefix(m(p), AF(p, t)))$: $n \leq \tau))$,

vi. **firing of the transition** $t \in T$ results in changing the state $S := (M, m, \tau)$ of the PNTS $PNTS$ into its state $S' := (M', m', \tau)$ that is denoted by $S\ [t\rangle\ S'$, where $\forall p \in P$:

- $M'(p) := M(p) - AF(p, t) + AF(t, p)$,
- $m'(p) := sort(suffix(m(p), AF(p, t))$++$create(\tau + TI(t, p), AF(t, p)))$,

vii. **elapsing of time interval** $\delta \in \mathbf{N}$ results in changing the state $S := (M, m, \tau)$ of PNTS $PNTS$ into its state $S' := (M, m, \tau + \delta)$, where $\forall t \in T$: $\neg(t$ ***en*** $(M, m, \tau))$, that is denoted by $S\ [\delta\rangle\ S'$, so that:

$(\forall t \in T\, \forall n \in \boldsymbol{N}, 1 \le n < \delta : \neg(t\ \boldsymbol{en}\ (M, m, \tau + n))) \wedge (\exists t \in T : t\ \boldsymbol{en}\ (M, m, \tau + \delta))$,

viii. if the transitions $t_1, t_2, \ldots, t_n \in T$ are enabled in the state $S := (M, m, \tau)$ of PNTS *PNTS* (i.e., $(t_1\ \boldsymbol{en}\ S) \wedge (t_2\ \boldsymbol{en}\ S) \wedge \ldots \wedge (t_n\ \boldsymbol{en}\ S)$) we say that these transitions are **enabled in parallel** in the state S that is denoted by $\{t_1, t_2, \ldots, t_n\}\ \boldsymbol{en}\ S$,

ix. finite nonempty sequence $\sigma := t_1\ t_2\ \ldots\ t_n$ of the transitions $t_1, t_2, \ldots, t_n \in T$ for which the following is valid in the state $S_1 := (M_1, m_1, \tau_1)$ of PNTS *PNTS*:

- $(M_1, m_1, \tau_1)\ [t_1\rangle\ (M_2, m_2, \tau_1)\ [t_2\rangle\ \ldots\ [t_n\rangle\ (M_{n+1}, m_{n+1}, \tau_1)$,
- $\forall t \in T: \neg(t\ \boldsymbol{en}\ (M_{n+1}, m_{n+1}, \tau_1))$,

is called **step** σ in the given state S_1 of PNTS *PNTS* and it is denoted by.

$(M_1, m_1, \tau_1)\ [\sigma\rangle\ (M_{n+1}, m_{n+1}, \tau_1)$,

x. finite nonempty sequence ρ of steps and time intervals elapsing that represents the following finite sequence

$(M_1, m_1, \tau_1)\ [\sigma_1\rangle\ (M_2, m_2, \tau_1)\ [\delta_1\rangle \ldots (M_{n+1}, m_{n+1}, \tau_n)\ [\delta_n\rangle\ (M_{n+1}, m_{n+1}, \tau_{n+1})$

of the state changes of PNTS *PNTS* is the sequence $\rho := \sigma_1\ \delta_1\ \sigma_2\ \delta_2\ \ldots\ \sigma_n\ \delta_n$ of steps $\sigma_1, \sigma_2, \ldots, \sigma_n$ and time intervals elapsing $\delta_1, \delta_2, \ldots, \delta_n$,

xi. we say the state S' of PNTS *PNTS* is reachable from its state S if there exists the finite sequence $\rho := \sigma_1\ \delta_1\ \sigma_2\ \delta_2\ \ldots\ \sigma_n\ \delta_n$ of steps $\sigma_1, \sigma_2, \ldots, \sigma_n$ and time intervals elapsing $\delta_1, \delta_2, \ldots, \delta_n$ such that $S\ [\sigma_1\ \delta_1\ \sigma_2\ \delta_2\ \ldots\ \sigma_n\ \delta_n\rangle\ S'$; the set of all the reachable states of PNTS *PNTS* from its state S is denoted by $[S\rangle$; the set of all the finite sequences $\rho := \sigma_1\ \delta_1\ \sigma_2\ \delta_2\ \ldots\ \sigma_n\ \delta_n$ associated with all the reachable states $S' \in [S\rangle$ is denoted by $[S\rangle\rangle$, that is,

$[S\rangle\rangle := \{\sigma_1\ \delta_1\ \sigma_2\ \delta_2 \ldots\ \sigma_n\ \delta_n \mid \exists S' \in [S\rangle : S\ [\sigma_1\ \delta_1\ \sigma_2\ \delta_2 \ldots\ \sigma_n\ \delta_n\rangle\ S', n \in \boldsymbol{N}\}$,

xii. the set of all the states $S := (M, m, \tau)$ of PNTS *PNTS* is denoted by $\boldsymbol{S}$,

xiii. the set of all the markings M associated with the set $\boldsymbol{S}$ of all the states of PNTS *PNTS* is denoted by $\boldsymbol{M}$, that is, $\boldsymbol{M} := \{M \mid (S = (M, m, \tau)) \wedge (S \in \boldsymbol{S})\}$,

xiv. **static state** $S_s := (M_s, m_s, \tau_s)$ of PNTS *PNTS* is every of its states where

$\forall p \in P \setminus RP : (M_s(p) = 0) \wedge (m_s(p) = <>)$,

xv. the set of all the static states $S_s := (M_s, m_s, \tau_s)$ of PNTS *PNTS* is denoted by $\boldsymbol{S_s}$,

xvi. the set of all the static markings M_s associated with the set $\boldsymbol{S_s}$ of all the static states of PNTS *PNTS* is denoted by $\boldsymbol{M_s}$, that is, $\boldsymbol{M_s} := \{M_s \mid (S_s := (M_s, m_s, \tau_s)) \wedge (S_s \in \boldsymbol{S_s})\}$,

xvii. the function $\xi: \boldsymbol{M} \to \boldsymbol{M_s}$ which assigns to each marking $M \in \boldsymbol{M}$ of a given PNTS *PNTS* the associated static marking $M_s \in \boldsymbol{M_s}$ is defined as follows:

- $\forall p \in RP: \xi(M(p)) := M(p)$,
- $\forall p \in P \setminus RP: \xi(M(p)) := 0$,

xviii. entry state $S_e := (M_e, m_e, \tau_e)$ of PNTS *PNTS* is every of its states where

- $k \in N$: $(Me(IP) = k) \wedge (length(me(IP)) = k)$,
- $\forall p \in P \setminus (RP \cup \{IP\})$: $(Me(p) = 0) \wedge (me(p) = <>)$,
- $\forall p \in RP$: $(Me(p) \geq 0) \wedge (me(p) \in N^0)$,

xix. the set of all the entry states $S_e := (M_e, m_e, \tau_e)$ of PNTS *PNTS* is denoted by $\boldsymbol{S_e}$,

xx. exit state $S_x := (M_x, m_x, \tau_x)$ of PNTS *PNTS* that is reachable from its entry state $S_e := (M_e, m_e, \tau_e)$ is every of its states where

- $S_x \in [S_e\rangle$,
- $M_x(OP) = M_e(IP)$,
- $\forall p \in P \setminus (RP \cup \{OP\})$: $(M_e(p) = 0) \wedge (m_e(p) = <>)$,

xxi. the set of all the exit states $S_x := (M_e, m_e, \tau_e)$ of PNTS *PNTS* that are reachable from its entry state $S_e := (M_e, m_e, \tau_e)$ is denoted by $[S_e\rangle_x$,

xxii. the set of all the exit states S_x of PNTS *PNTS* that are reachable from all its entry states $S_e \in \boldsymbol{S_e}$ is denoted by $\boldsymbol{S_x}$. □

The above established concepts are demonstrated in a simple example of the PNTS *PNTS*1 := (*P*, *T*, *A*, *AF*, *TP*, *TI*, *IP*, *OP*, *RP*) that is shown in **Figure 1**, where *P* := {**IP**, **P1**, **R1**, **OP**}, *T* := {**T1**, **T2**, **T3**}, *A* := {(**IP**, **T1**), (**IP**, **T2**), (**T1**, **P1**), (**T2**, **P1**), (**R1**, **T1**), (**P1**, **T3**), (**T3**, **R1**), (**T3**, **OP**)}, *AF* := {((**IP**, **T1**), 1), ((**IP**, **T2**), 1), ((**T1**, **P1**), 1), ((**T2**, **P1**), 2), ((**R1**, **T1**), 1), ((**P1**, **T3**), 1), ((**T3**, **R1**), 1), ((**T3**, **OP**), 1)}, *TP* := {(**T1**, 2), (**T2**, 1), (**T3**, 1)}, *TI* := {((**T1**, **P1**), 3), ((**T2**, **P1**), 3), ((**T3**, **R1**), 1), ((**T3**, **OP**), 4)}, *IP* := **IP**, *OP* := **OP**, *RP* := {**R1**}.

PNTS *PNTS*1 is in its entry state $S_e := (M_e, m_e, \tau_e)$, where marking $M_e := (M_e(\mathbf{IP}), M_e(\mathbf{P_1}), M_e(\mathbf{R1}), M_e(\mathbf{OP})) = (2, 0, 2, 0)$, time marking $m_e := (m_e(\mathbf{IP}), m_e(\mathbf{P_1}), m_e(\mathbf{R1}), m_e(\mathbf{OP})) = (<0, 2>, <>, <0, 0>, <>)$ and net time $\tau_e = 0$ (i.e., $\tau_e = \tau$). Static marking $M_s \in \boldsymbol{M_s}$ associated with the entry marking M_e (see (xiv) and (xvii) of Definition 4) has the value $M_s := \xi(M_e) = (0, 0, 2, 0)$.

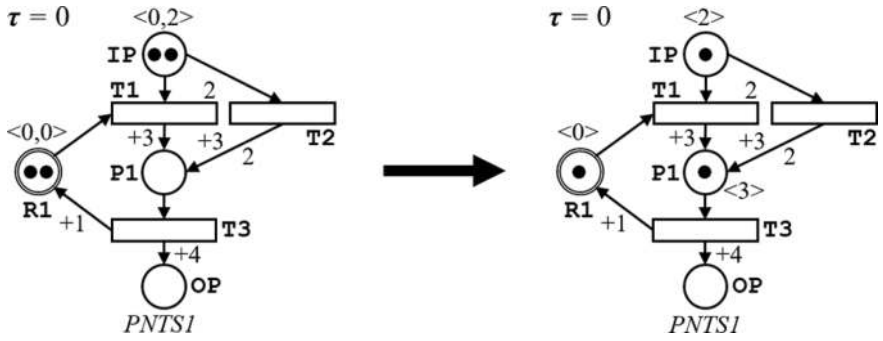

Figure 1. Firing of transition **T1** in PNTS *PNTS*1.

Time marking m of any PNTS expresses the current time state of the modeled system using the final (or empty) ascending ordered sequences of non-negative integers (i.e., elements of the set $\boldsymbol{N}^{\#}$) associated with each of its places. The individual values of the time marking m associated with the arbitrary place p of the given PNTS in its state S, informally said, represent the values of the net time τ at which the respective token can first participate in the firing of selected enabled transition t of the given PNTS.

The transitions **T1** and **T2** are enabled in the entry state S_e because (see (v) of Definition 4):

- $\forall p \in \bullet\mathbf{T1}$: $(2 = M(\mathbf{IP}) \geq AF(\mathbf{IP}, \mathbf{T1}) = 1) \wedge (2 = M(\mathbf{R1}) \geq AF(\mathbf{R1}, \mathbf{T1}) = 1) \wedge (\forall n \in elements(prefix(m(\mathbf{IP}), AF(\mathbf{IP}, \mathbf{T1}))) = elements(prefix(<0, 2>, 1)) = elements(<0>) = \{0\}: 0 \leq 0) \wedge (\forall n \in elements(prefix(m(\mathbf{R1}), AF(\mathbf{R1}, \mathbf{T1}))) = elements(prefix(<0, 0>, 1)) = elements(<0>) = \{0\}: 0 \leq 0)$,
- $\forall p \in \bullet\mathbf{T2}$: $(2 = M(\mathbf{IP}) \geq AF(\mathbf{IP}, \mathbf{T1}) = 1) \wedge (\forall n \in elements(prefix(m(\mathbf{IP}), AF(\mathbf{IP}, \mathbf{T2}))) = elements(prefix(<0, 2>, 1)) = elements(<0>) = \{0\}: 0 \leq 0)$.

When enabling individual transitions of the given PNTS so-called **conflicts** can originate in its certain markings (or **conflict transitions**). At the enabling of the transitions t_1 and t_2 of the given PNTS in its state S the conflict occurs, if both transitions t_1 and t_2 have at least one input place, each of the transitions t_1 and t_2 is individually enabled in the state S, but the transitions t_1 and t_2 are not enabled in parallel in the state S (see (viii) of Definition 4) and enabling of one of them will prevent enabling of the other (i.e., $(\bullet t_1 \cap \bullet t_2 \neq \emptyset) \wedge (t_1 \; \boldsymbol{en} \; S) \wedge (t_2 \; \boldsymbol{en} \; S) \wedge \neg(\{t_1, t_2\} \; \boldsymbol{en} \; S)$). The term of conflict transitions can be obviously easily generalized for the case of a finite set $t_1, t_2, \ldots, t_n$, $n \in \boldsymbol{N}$ of the transitions of the given PNTS.

The transitions **T1** and **T2** in the entry state S_e of PNTS *PNTS*1 are conflict transitions because the time marking $m_e(\mathbf{IP}) = <0, 2>$ (i.e., only one token of the entry marking $M_e(\mathbf{IP})$ may participate in the firing of the transition **T1** or **T2** in the net time $\tau_e = 0$). When solving such transitions conflict we therefore follow the rule which determines, informally said, that from the set of conflict transitions the one will be enabled, whose value of the transition priority function *TP* is the highest. If such transition from the set of conflict transitions does not exist, the given conflict would have to be solved by other means. The transition **T1** is then enabled in the entry state S_e on the basis of that rule in our studied example (because $TP(\mathbf{T1}) = 2$ and $TP(\mathbf{T2}) = 1$).

Firing of the transition **T1** changes the entry state $S_e := (M_e, m_e, \tau_e)$ of the PNTS *PNTS*1 into its state $S_1 := (M_1, m_1, \tau_e)$ (i.e., $S_e \; [\mathbf{T1}\rangle \; S_1$—see **Figure 1**), where (see (vi) of Definition 4):

- $M_1(\mathbf{IP}) := M_e(\mathbf{IP}) - AF(\mathbf{IP}, \mathbf{T1}) = 2 - 1 = 1$,
- $m_1(\mathbf{IP}) := sort(suffix(m_e(\mathbf{IP}), AF(\mathbf{IP}, \mathbf{T1}))) = sort(suffix(<0, 2>, 1)) = sort(<2>) = <2>$,
- $M_1(\mathbf{P1}) := M_e(\mathbf{P1}) + AF(\mathbf{T1}, \mathbf{P1}) = 0 + 1 = 1$,
- $m_1(\mathbf{P1}) := sort(create(\tau + TI(\mathbf{T1}, \mathbf{P1}), AF(\mathbf{T1}, \mathbf{P1}))) = sort(create(0 + 3, 1)) = sort(create(3, 1)) = sort(<3>) = <3>$,
- $M_1(\mathbf{R1}) := M_e(\mathbf{R1}) - AF(\mathbf{R1}, \mathbf{T1}) = 2 - 1 = 1$,
- $m_1(\mathbf{R1}) := sort(suffix(m_e(\mathbf{R1}), AF(\mathbf{R1}, \mathbf{T1}))) = sort(suffix(<0, 0>, 1)) = sort(<0>) = <0>$.

http://dx.doi.org/10.5772/intechopen.76769

There is no enabled transition in the state $S_1 := (M_1, m_1, \tau_1)$ and it is necessary to perform the time interval elapsing with the value of $\delta = 2$. This will change the state $S_1 := (M_1, m_1, \tau_e)$ into the state $S_2 := (M_1, m_1, \tau_1)$, where $\tau_1 := \tau_e + \delta = 2$ (i.e., S_1 [2⟩ S_2). It can be easily shown that transition **T1** in the state S_2 is enabled and firing of this transition changes the state $S_2 := (M_1, m_1, \tau_1)$ of the PNTS *PNTS*1 into its state $S_3 := (M_2, m_2, \tau_1)$ (i.e., S_2 [**T1**⟩ S_3), where $M_2 := (M_2(\mathbf{IP}), M_2(\mathbf{P1}), M_2(\mathbf{R1}), M_2(\mathbf{OP})) = (0, 2, 0, 0)$ and $m_2 := (m_2(\mathbf{IP}), m_2(\mathbf{P1}), m_2(\mathbf{R1}), m_2(\mathbf{OP})) = (\langle\rangle, \langle 3, 5\rangle, \langle\rangle, \langle\rangle)$.

It can then be easily verified that S_3 [1⟩ S_4 [**T3**⟩ S_5 [2⟩ S_6 [**T3**⟩ S_x, where:

- $S_4 = (M_2, m_2, \tau_2) = ((0, 2, 0, 0), (\langle\rangle, \langle 3, 5\rangle, \langle\rangle, \langle\rangle), 3)$,
- $S_5 = (M_3, m_3, \tau_2) = ((0, 1, 1, 1), (\langle\rangle, \langle 5\rangle, \langle 4\rangle, \langle 7\rangle), 3)$,
- $S_6 = (M_3, m_3, \tau_3) = ((0, 1, 1, 1), (\langle\rangle, \langle 5\rangle, \langle 4\rangle, \langle 7\rangle), 5)$,
- $S_x = (M_4, m_4, \tau_3) = ((0, 0, 2, 2), (\langle\rangle, \langle\rangle, \langle 4, 6\rangle, \langle 7, 9\rangle), 5)$.

There are no enabled transitions in the exit state $S_x := (M_4, m_4, \tau_3)$ of PNTS *PNTS*1 that is reachable from the entry state $S_e := (M_e, m_e, \tau_e)$ (see (xx) of Definition 4) and there is also no time interval elapsing value δ in this state that enables any of the transitions.

The set $\boldsymbol{AM_s} \subseteq \boldsymbol{M_s}$ of all the **allowed static markings** of the given PNTS *PNTS*, informally said, expresses how many tokens may be located in its individual resource places if PNTS *PNTS* be in its (now no longer arbitrary) allowed entry state $AS_e \in \boldsymbol{AS_e}$, where $\boldsymbol{AS_e} \subseteq \boldsymbol{S_e}$. For instance, the set $\boldsymbol{AM_s}$ of the PNTS *PNTS*1 (see **Figure 1**) can be defined as $\boldsymbol{AM_s} := \{(0, 0, k, 0) \mid k \in \boldsymbol{N}\}$, that is, there must be at least one token in the resource place **R1** (and of course at least one token in the input place **IP**) in any allowed entry state $AS_e \in \boldsymbol{AS_e}$.

Definition 5. Let $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ be a PNTS, $\boldsymbol{AM_s} \subseteq \boldsymbol{M_s}$ be the set of all of its allowed static markings and $\boldsymbol{AS_e} := \{AS_e \mid (AS_e = (AM_e, am_e, 0)) \wedge (\xi(AM_e) \in \boldsymbol{AM_s})\}$ be the set of all of its allowed entry states. Then:

i. *PNTS* is ***k*-bounded** PNTS if

$$\forall AS_e \in \boldsymbol{AS_e}\ \exists k \in \boldsymbol{N}_0 \forall p \in P\ \forall S \in [AS_e\rangle, S{:=}(M, m, \tau) : M(p) \leq k,$$

ii. *PNTS* is **proper-formed** PNTS if

$$\forall AS_e \in \boldsymbol{AS_e} : (\forall S \in [AS_e\rangle \exists S_x \in [AS_e\rangle_x : S_x \in [S\rangle) \wedge (\exists n \in \boldsymbol{N} : |[AS_e\rangle\rangle| = n),$$

iii. proper-formed *PNTS* is **well-formed** PNTS if

$$\forall AS_e \in \boldsymbol{AS_e} \forall S_x \in [AS_e\rangle_x, S_x{:=}(M_x, m_x, \tau_x) : \xi(M_x) \in \boldsymbol{AM_s},$$

iv. well-formed *PNTS* is **pure-formed** PNTS if

$$\forall AS_e \in \boldsymbol{AS_e} \forall S_x \in [AS_e\rangle_x, S_x{:=}(M_x, m_x, \tau_x) : \xi(AM_e) = \xi(M_x).$$ □

PNTS is **proper-formed** PNTS if for any of its state S that is reachable from any allowed entry state $AS_e \in \boldsymbol{AS_e}$ there exists its output state S_x that is also reachable from its allowed entry state

$AS_e \in \boldsymbol{AS_e}$ such that the output state S_x is also reachable from the state S (i.e., $\forall S \in [AS_e\rangle\ \exists S_x \in [AS_e\rangle_x$: $S_x \in [S\rangle$). Furthermore, the cardinality of the set $[AS_e\rangle\rangle$ of all the sequences $\rho := \sigma_1\ \delta_1\ \sigma_2\ \delta_2\ \ldots\ \sigma_n\ \delta_n$ associated with all the reachable states $S \in [AS_e\rangle\rangle$ must be finite (i.e., $(\exists n \in \boldsymbol{N}$: $|\ [AS_e\rangle\rangle| = n$).

Proper-formed *PNTS* is **well-formed** PNTS if for any of its allowed entry state $AS_e \in \boldsymbol{AS_e}$ and for any of its exit state $S_x \in [AS_e\rangle_x$, where $S_x := (M_x, m_x, \tau_x)$, it is true that the exit static marking $\xi(M_x)$ of all its resource places **is an element** of the set $\boldsymbol{AM_s}$ of all its allowed static markings if PNTS *PNTS* be in its allowed entry state $AS_e \in \boldsymbol{AS_e}$ (i.e., $\forall AS_e \in \boldsymbol{AS_e}\ \forall S_x \in [AS_e\rangle_x$, $S_x := (M_x, m_x, \tau_x)$: $\xi(M_x) \in \boldsymbol{AM_s}$).

Well-formed *PNTS* is **pure-formed** PNTS if for any of its allowed entry state $AS_e \in \boldsymbol{AS_e}$, where $AS_e := (AM_e, am_e, \tau_e)$, and for any of its exit state $S_x \in [AS_e\rangle_x$, where $S_x := (M_x, m_x, \tau_x)$, it is true that the exit static marking $\xi(M_x)$ of all its resource places **is equal to** the entry static marking $\xi(AM_e)$ of all its resource places that is associated with the allowed entry state AS_e (i.e., $\forall AS_e \in \boldsymbol{AS_e}\ \forall S_x \in [AS_e\rangle_x$, $S_x := (M_x, m_x, \tau_x)$: $\xi(AM_e) = \xi(M_x)$).

For instance, if the set $\boldsymbol{AM_s}$ of the PNTS *PNTS*1 (see **Figure 1**) is defined as:

i. $\boldsymbol{AM_s} := \{(0, 0, k, 0) \mid k \in \boldsymbol{N}\}$ (i.e., there must be at least one token in the resource place **R1** in any allowed entry state $AS_e \in \boldsymbol{AS_e}$), then it can be shown that PNTS *PNTS*1 is *k*-bounded, proper-formed, well-formed and pure-formed PNTS,

ii. $\boldsymbol{AM_s} := \{(0, 0, 0, 0)\}$ (i.e., there may not be any token in the resource place **R1** in any allowed entry state $AS_e \in \boldsymbol{AS_e}$), then it can be shown that PNTS *PNTS*1 is *k*-bounded, proper-formed, but not well-formed or pure-formed PNTS (see for instance the sequence

$$((1,0,0,0),(<0>,<>,<>,<>),0)\ [\mathbf{T2}\rangle\ ((0,2,0,0),(<>,<3,3>,<>,<>),0)\ [3\rangle.$$
$$((0,2,0,0),(<>,<3,3>,<>,<>),3)\ [\mathbf{T3\ T3}\rangle\ ((0,0,2,2),(<>,<>,<4,4>,<7,7>),3),$$
where $\xi(M_x) = \xi((0,0,2,2)) = (0,0,2,0) \notin \{(0,0,0,0)\} = \boldsymbol{AM_s}$).

Lemma 1. If *PNTS* is proper-formed PTNS then *PNTS* is *k*-bounded PNTS.

Proof. Clear. PNTS *PNTS* := (*P*, *T*, *A*, *AF*, *TP*, *TI*, *IP*, *OP*, *RP*) is a connected net that contains the finite set *T* of the transitions. Then the finite number of tokens will be added to each of the places $p \in P$ by firing each of the transitions $t \in T$. The number of states $S \in [AS_e\rangle$ for any allowed entry state $AS_e \in \boldsymbol{AS_e}$ must be also finite because *PNTS* is proper-formed PNTS (i.e., $\exists n \in \boldsymbol{N}$: $|[AS_e\rangle\rangle| = n$). From these facts then immediately follows that in any state $S \in [AS_e\rangle$ the finite number of tokens must be placed in any place $p \in P$, where any final number of tokens is placed in the input place *IP* in the entry state AS_e. From these facts then immediately follows that $\forall AS_e \in \boldsymbol{AS_e}\ \exists k \in \boldsymbol{N}_0\ \forall p \in P\ \forall S \in [AS_e\rangle$, $S := (M, m, \tau) : M(p) \le k$. □

Definition 6. Process Petri net with time stamps (PPNTS) *PPNTS* is an ordered couple *PPNTS* := (*PNTS*, S_e), where *PNTS* := (*P*, *T*, *A*, *AF*, *TP*, *TI*, *IP*, *OP*, *RP*) is the PNTS and $S_e \in \boldsymbol{S_e}$ is the entry state of PNTS *PNTS*. The class of all PPNTSs will be denoted by ***PPNTS***. □

http://dx.doi.org/10.5772/intechopen.76769

3. Algebraic operators COMP and SYNC and their properties

We study the issue of transforming PNTS through precisely defined binary operator **COMP** and n-ary operator **SYNC** over the class ***PNTS*** and we also examine the preservation of individual PNTS's properties when applying each of these operators. Formal enrollment of an application of generally n-ary operator **OP** whose operands are the PNTS $PNTS_1$, $PNTS_2$, …, $PNTS_n$ ($n \in \boldsymbol{N}$) and whose application requires the specification of values of k formal parameters ($k \in \boldsymbol{N}$) par_1, par_2, … par_k, will be denoted by the expression.

$$PNTS := [PNTS_1, PNTS_2, \ldots, PNTS_n].\mathbf{OP}(par_1, par_2, \ldots, par_k),$$

where $PNTS$ is the resulting PNTS.

Definition 7. Let $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$ and $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$ be the PNTSs. Let $\boldsymbol{AM_{s1}} := \{(AM_{s1}(IP_1), AM_{s1}(p1_1), \ldots, AM_{s1}(p1_n), AM_{s1}(r1_1), \ldots, AM_{s1}(r1_m), AM_{s1}(OP_1)) \mid P_1 := \{p1_1, \ldots, p1_n, r1_1, \ldots, r1_m\}, RP_1 := \{r1_1, \ldots, r1_m\}, n \in \boldsymbol{N}, m \in \boldsymbol{N}\}$ be the set of all the allowed static markings of $PNTS_1$, $\boldsymbol{AM_{s2}} := \{(AM_{s2}(IP_2), AM_{s2}(p2_1), \ldots, AM_{s2}(p2_k), AM_{s2}(r2_1), \ldots, AM_{s2}(r2_h), AM_{s2}(OP_2)) \mid P_2 := \{p2_1, \ldots, p2_k, r2_1, \ldots, r2_h\}, RP_2 := \{r2_1, \ldots, r2_h\}, k \in \boldsymbol{N}, h \in \boldsymbol{N}\}$ be the set of all the allowed static markings of $PNTS_2$.

Cartesian product $\boldsymbol{AM_{s1}} \otimes \boldsymbol{AM_{s2}}$ is then the following set:

$$\begin{aligned}\boldsymbol{AM_{s1}} \otimes \boldsymbol{AM_{s2}} := \{&(AM_{s1}(IP_1), AM_{s1}(p1_1), \ldots, AM_{s1}(p1_n), AM_{s1}(r1_1), \ldots, AM_{s1}(r1_m), AM_{s1}(OP_1),\\ &AM_{s2}(IP_2), AM_{s2}(p2_1), \ldots, AM_{s2}(p2_k), AM_{s2}(r2_1), \ldots, AM_{s2}(r2_h), AM_{s2}(OP_2)) \mid .\\ &(AM_{s1}(IP_1), AM_{s1}(p1_1), \ldots, AM_{s1}(p1_n), AM_{s1}(r1_1), \ldots, AM_{s1}(r1_m), AM_{s1}(OP_1)) \in \boldsymbol{AM_{s1}} \wedge .\\ &(AM_{s2}(IP_2), AM_{s2}(p2_1), \ldots, AM_{s2}(p2_k), AM_{s2}(r2_1), \ldots, AM_{s2}(r2_h), AM_{s2}(OP_2)) \in \boldsymbol{AM_{s2}}\}.\end{aligned}$$

PNTS $PNTS_1$ and $PNTS_2$ are disjoint and we denote this fact by $PNTS_1 \measuredangle PNTS_2$ if.

$$(P_1 \cap P_2 = \varnothing) \wedge (T_1 \cap T_2 = \varnothing). \qquad \square$$

Definition 8. The function **COMP**: $\boldsymbol{PNTS} \times \boldsymbol{PNTS} \rightarrow \boldsymbol{PNTS}$ **of nets composition** is defined as follows: if $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$ and $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$ be the arbitrary PNTSs, $PNTS_1 \measuredangle PNTS_2$, t be an arbitrary transition, where $(t \notin T_1) \wedge (t \notin T_2)$, $ti \in \boldsymbol{N}_0$, then $PNTS := [PNTS_1, PNTS_2].\mathbf{COMP}(t, ti)$, where PNTS $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ fulfills the following:

i. $P := P_1 \cup P_2$,

ii. $T := T_1 \cup T_2 \cup \{t\}$,

iii. $A := A_1 \cup A_2 \cup \{(OP_1, t), (t, IP_2)\}$,

iv. $AF := AF_1 \cup AF_2 \cup \{((OP_1, t), 1), ((t, IP_2), 1)\}$,

v. $TP := TP_1 \cup TP_2 \cup \{(t, 1)\}$,

vi. $TI := TI_1 \cup TI_2 \cup \{(t, IP_2), ti)\}$,

vii. $IP := IP_1$,

viii. $OP := OP_2$,

ix. $RP := RP_1 \cup RP_2$. □

Symbolic representation of PNTS $[PNTS_1, PNTS_2]$.**COMP**(t, ti) can be seen in **Figure 2**.

Lemma 2. Let $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$ and $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$ be two arbitrary PNTS, $PNTS_1 \angle PNTS_2$, t be an arbitrary transition, $(t \notin T_1) \wedge (t \notin T_2)$, $ti \in \mathbf{N}_0$, $\boldsymbol{AM}_{s1}$ and $\boldsymbol{AM}_{s2}$ be the sets of all the allowed static markings of $PNTS_1$ and $PNTS_2$. Let $PNTS := [PNTS_1, PNTS_2]$.**COMP**(t, ti).

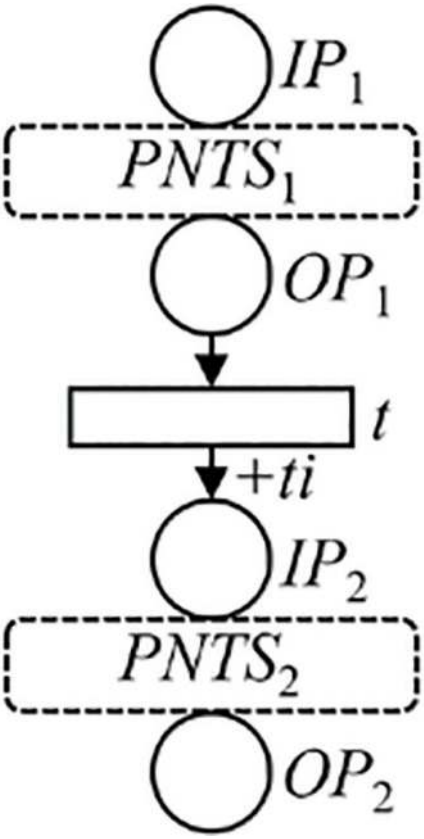

Figure 2. Symbolic representation of PNTS $[PNTS_1, PNTS_2]$.**COMP**(t, ti).

If $PNTS_1$ and $PNTS_2$ are proper-formed, resp. well-formed, resp. pure-formed, PNTS and $\boldsymbol{AM}_s = \boldsymbol{AM}_{s1} \otimes \boldsymbol{AM}_{s2}$ be the set of all the allowed static markings of PNTS $PNTS$, then also resulting $PNTS$ is proper-formed, resp. well-formed, resp. pure-formed, PNTS.

Proof. Clear, it directly follows from Definition 5, Definition 7 and Definition 8. □

Definition 9. The function **SYNC**: $\boldsymbol{PNTS} \times \boldsymbol{PNTS} \times \ldots \times \boldsymbol{PNTS} \rightarrow \boldsymbol{PNTS}$ **of synchronous nets composition** is defined as follows: if $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$, $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$, …, $PNTS_n := (P_n, T_n, A_n, AF_n, TP_n, TI_n, IP_n, OP_n, RP_n)$, be the arbitrary PNTSs, $\forall i, 1 \le i \le n, \forall j, 1 \le j \le n$: $i \ne j \Rightarrow PNTS_i \angle PNTS_j$, where $n \in \boldsymbol{N}$, pi and po be the arbitrary places, $(pi \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (po \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (pi \ne po)$, ti and to be the arbitrary transitions, $(ti \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (to \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (ti \ne to)$, $af1 \in \boldsymbol{N}$, $af2 \in \boldsymbol{N}$, …, $afn \in \boldsymbol{N}$, $ti1 \in \boldsymbol{N}_0$, $ti2 \in \boldsymbol{N}_0$, …, $tin \in \boldsymbol{N}_0$, $tio \in \boldsymbol{N}_0$, then

$$PNTS := [PNTS_1, PNTS_2, \ldots, PNTS_n].\mathbf{SYNC}(pi, po, ti, to, af1, \ldots, afn, ti1, \ldots, tin, tio),$$

where PNTS $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ fulfills the following:

i. $P := P_1 \cup P_2 \cup \ldots \cup P_n \cup \{pi, po\}$,

ii. $T := T_1 \cup T_2 \cup \ldots \cup T_n \cup \{ti, to\}$,

iii. $A := A_1 \cup A_2 \cup \ldots \cup A_n \cup \{(pi, ti), (ti, IP_1), \ldots, (ti, IP_n), (OP_1, to), \ldots, (OP_n, to), (to, po)\}$,

iv. $AF := AF_1 \cup AF_2 \cup \ldots \cup AF_n \cup \{((pi, ti), 1), ((ti, IP_1), af1), \ldots, ((ti, IP_n), afn), ((OP_1, to), af1), \ldots, ((OP_n, to), afn), ((to, po), 1)\}$,

v. $TP := TP_1 \cup TP_2 \cup \ldots \cup TP_n \cup \{(ti, 1), (to, 1)\}$,

vi. $TI := TI_1 \cup TI_2 \cup \ldots \cup TI_n \cup \{((ti, IP_1), ti1), \ldots, ((ti, IP_n), tin), ((to, po), tio)\}$,

vii. $IP := pi$,

viii. $OP := po$,

ix. $RP := RP_1 \cup RP_2 \cup \ldots \cup RP_n$. □

Symbolic representation of PNTS $[PNTS_1, PNTS_2, \ldots, PNTS_n].\mathbf{SYNC}(pi, po, ti, to, af1, \ldots, afn, ti1, \ldots, tin, tio)$ can be seen in **Figure 3**.

Lemma 3. Let $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$, $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$, …, $PNTS_n := (P_n, T_n, A_n, AF_n, TP_n, TI_n, IP_n, OP_n, RP_n)$ be arbitrary PNTSs, $\forall i, 1 \le i \le n, \forall j, 1 \le j \le n$: $i \ne j \Rightarrow PNTS_i \angle PNTS_j$, where $n \in \boldsymbol{N}$, pi and po be arbitrary places, $(pi \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (po \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (pi \ne po)$, ti and to be arbitrary transitions, $(ti \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (to \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (ti \ne to)$, $af1 \in \boldsymbol{N}$, $af2 \in \boldsymbol{N}$, …, $afn \in \boldsymbol{N}$, $ti1 \in \boldsymbol{N}_0$, $ti2 \in \boldsymbol{N}_0$, …, $tin \in \boldsymbol{N}_0$, $tio \in \boldsymbol{N}_0$ and $\boldsymbol{AM}_{s1}$, $\boldsymbol{AM}_{s2}$, …, $\boldsymbol{AM}_{sn}$ be the sets of all the allowed static markings of $PNTS_1$, $PNTS_2$, …, $PNTS_n$. Let $PNTS := [PNTS_1, PNTS_2, \ldots, PNTS_n].\mathbf{SYNC}(pi, po, ti, to, af1, \ldots, afn, ti1, \ldots, tin, tio)$.

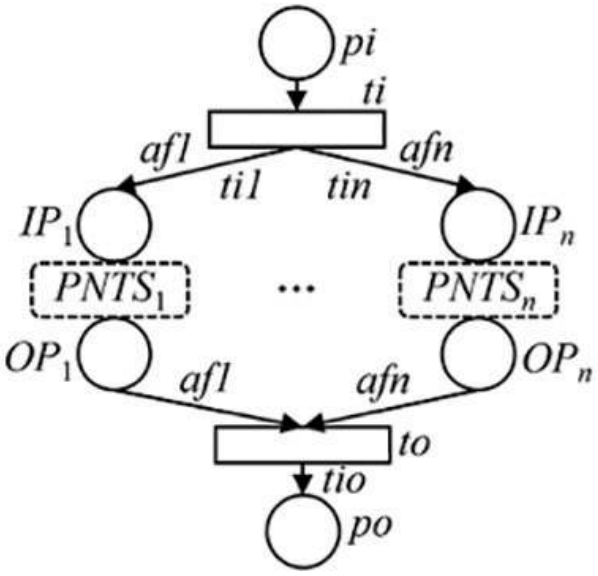

Figure 3. Symbolic representation of PNTS [$PNTS_1$, $PNTS_2$, …, $PNTS_n$].**SYNC**(*pi*, *po*, *ti*, *to*, *af*1, …, *afn*, *ti*1, …, *tin*, *tio*).

If $PNTS_1$, $PNTS_2$, …, $PNTS_n$ are proper-formed, resp. well-formed, resp. pure-formed, PNTS and $\boldsymbol{AM}_s = \boldsymbol{AM}_{s1} \otimes \boldsymbol{AM}_{s2} \otimes \ldots \otimes \boldsymbol{AM}_{sn}$ is the set of all the allowed static markings of PNTS *PNTS*, then also *PNTS* is proper-formed, resp. well-formed, resp. pure-formed, PNTS.

Proof. Clear, it directly follows from Definition 5, Definition 7 and Definition 9. □

4. Process Petri nets with time stamps and their applications in project management area

Critical Path Method (CPM) is a method used in modeling and project management that was developed at the end of 1950s and that is commonly used for all the types of projects including software development [10]. The CPM is the most widely used method of so-called network analysis, even though it is designed to analyze the time consumption of only deterministic projects, that is, projects where the duration of each of their activities is exactly known, including all their sub-activities.

The basis for using CPM is to create a project model that includes:

- the list of all activities needed to complete the project,
- the time duration of each activity that is constant,
- the dependencies between the project activities,

http://dx.doi.org/10.5772/intechopen.76769

A critical path is then a designation for a sequence of activities whose time duration directly affects the time duration of the entire project. The activities that make up the critical path are then referred to as critical activities. There may be several critical paths in the project. When managing the project, a sequence of activities within a given network chart describing this project that increases the longest total time duration of a project is called its critical path. The critical path within the network chart can be used to determine the shortest time required to complete the project. The application of the CPM method can therefore determine which activities within the studied project are "critical" (i.e., activities on the longest path in the network chart describing the project) and which activities may be delayed in the execution of the project without increasing its total time.

The special class $\boldsymbol{CPNET} \subset \boldsymbol{PNTS}$ of PNTS is introduced in the following paragraphs to represent network chart used in the CPM method through PNTS. Special unary operator **JOIN** that is required in the definition of the class ***CPNET*** is introduced first.

Definition 10. The function **JOIN**: $\boldsymbol{PNTS} \rightarrow \boldsymbol{PNTS}$ **of net transition joining** is defined as follows: if $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$ be the arbitrary PNTS, $p \notin P_1$ be the arbitrary place, $t1$ and $t2$ be the arbitrary transitions, $(t1 \neq t2) \wedge (t1 \in T_1) \wedge (t2 \in T_1)$, $ti \in \mathbf{N}_0$, then $PNTS := PNTS_1.\mathbf{JOIN}(p, t1, t2, ti)$, where PNTS $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$ fulfills the following:

i. $P := P_1 \cup \{p\}$,

ii. $T := T_1$,

iii. $A := A_1 \cup \{(t1, p), (p, t2)\}$,

iv. $AF := AF_1 \cup \{((t1, p), 1), ((p, t2), 1)\}$,

v. $TP := TP_1 \cup \{(t, 1)\}$,

vi. $TI := TI_1 \cup \{(t1, p), ti)\}$,

vii. $IP := IP_1$,

viii. $OP := OP_1$,

ix. $RP := RP_1$. □

Symbolic representation of the unary operator **JOIN** application over the PNTS $PNTS_1$ can be seen in **Figure 4**.

Definition 11. Let $PNTS_1 := (P_1, T_1, A_1, AF_1, TP_1, TI_1, IP_1, OP_1, RP_1)$, $PNTS_2 := (P_2, T_2, A_2, AF_2, TP_2, TI_2, IP_2, OP_2, RP_2)$, …, $PNTS_n := (P_n, T_n, A_n, AF_n, TP_n, TI_n, IP_n, OP_n, RP_n)$, where $n \in \mathbf{N}$, be the arbitrary PNTSs, $\forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n: i \neq j \Rightarrow PTSN_i \measuredangle PTSN_j$. The class $\boldsymbol{CPNET} \subset \boldsymbol{PNTS}$ then contains the following PNTSs:

i. if p be an arbitrary place, $p \notin (P_1 \cup P_2 \cup \ldots \cup P_n)$, then PNTS $BASE_p \in \boldsymbol{CPNET}$, where $BASE_p := (\{p\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, p, p, \emptyset\}$,

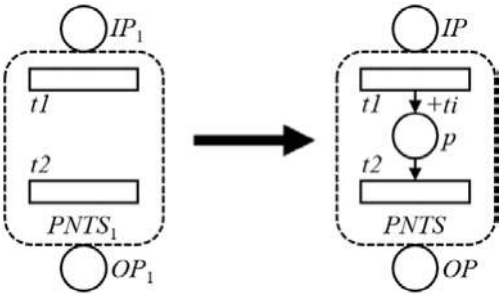

Figure 4. Symbolic representation of PNTS $PNTS := PNTS_1$.**JOIN**(p, $t1$, $t2$, ti).

ii. if $PNTS_1 \in \boldsymbol{CPNET}$, $PNTS_2 \in \boldsymbol{CPNET}$, t be an arbitrary transition, where $(t \notin T_1) \wedge (t \notin T_2)$, $ti \in \boldsymbol{N}_0$, then also $[PNTS_1, PNTS_2]$.**COMP**(t, ti) $\in \boldsymbol{CPNET}$,

iii. if $PNTS_1, PNTS_2, \ldots, PNTS_n \in \boldsymbol{CPNET}$, pi and po be the arbitrary places, $(pi \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (po \notin P_1 \cup P_2 \cup \ldots \cup P_n) \wedge (pi \neq po)$, ti and to be the arbitrary transitions, $(ti \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (to \notin T_1 \cup T_2 \cup \ldots \cup T_n) \wedge (ti \neq to)$, $ti1 \in \boldsymbol{N}_0$, $ti2 \in \boldsymbol{N}_0$, …, $tin \in \boldsymbol{N}_0$, $tio \in \boldsymbol{N}_0$, then also PNTS $PNTS \in \boldsymbol{CPNET}$, where

$$PNTS := [PNTS_1, PNTS_2, \ldots, PNTS_n].\mathbf{SYNC}(pi, po, ti, to, 1, \ldots, 1, ti1, ti2, \ldots, tin, tio),$$

iv. if $PNTS_1 \in \boldsymbol{CPNET}$, $p \notin P_1$ be an arbitrary place, $t1$ and $t2$ be the arbitrary transitions, $(t1 \neq t2) \wedge (t1 \in T_1) \wedge (t2 \in T_1)$, $ti \in \boldsymbol{N}_0$ and $af \in \boldsymbol{N}$, then also $PNTS \in \boldsymbol{CPNET}$, where $PNTS := (P, T, A, AF, TP, TI, IP, OP, RP)$, such that:

- $PNTS := PNTS_1.\boldsymbol{JOIN}(p, t1, t2, ti)$,
- $\vdash(\exists\, x_1\, x_2 \ldots x_n \in \mathbf{CIRCUITS}_{PNTS}: (x_1 \in \mathrm{T}) \wedge (x_n \in P) \wedge (n \in N))$. □

Four simple PNTSs $BASE_{\mathbf{P_1}}$, $CPNET1$, $CPNET2$ and $CPNET3$ that are the members of the class ***CPNET*** can be seen in **Figure 5**, where:

- $CPNET1 := [[BASE_{\mathbf{P_2}}, BASE_{\mathbf{P3}}].\mathbf{COMP}(\mathbf{T2}, 2), BASE_{\mathbf{P4}}].\mathbf{COMP}(\mathbf{T3}, 5)$,
- $CPNET2 := [[BASE_{\mathbf{P6}}, BASE_{\mathbf{P8}}].\mathbf{COMP}(\mathbf{T6}, 2), BASE_{\mathbf{P7}}]$

  $.\mathbf{SYNC}(\mathbf{P5}, \mathbf{P9}, \mathbf{T5}, \mathbf{T7}, 1, 1, 1, 6, 3)$,
- $CPNET3 := [ANET2, ANET3, BASE_{\mathbf{P_1}}].\mathbf{SYNC}(\mathbf{IP}, \mathbf{OP}, \mathbf{T1}, \mathbf{T8}, 1, 1, 1, 4, 1, 8, 2)$

  $.\mathbf{JOIN}(\mathbf{P10}, \mathbf{T3}, \mathbf{T5}, 4)$.

  $.\mathbf{JOIN}(\mathbf{P11}, \mathbf{T3}, \mathbf{T6}, 5)$.

Note also that the PNTS $CPNET3$ does not contain any circuit as it is required in the (4) of Definition 11.

Lemma 4. Let $PNTS \in \boldsymbol{CPNET}$ be an arbitrary PNTS. Then $PNTS$ is pure-formed PNTS.

http://dx.doi.org/10.5772/intechopen.76769

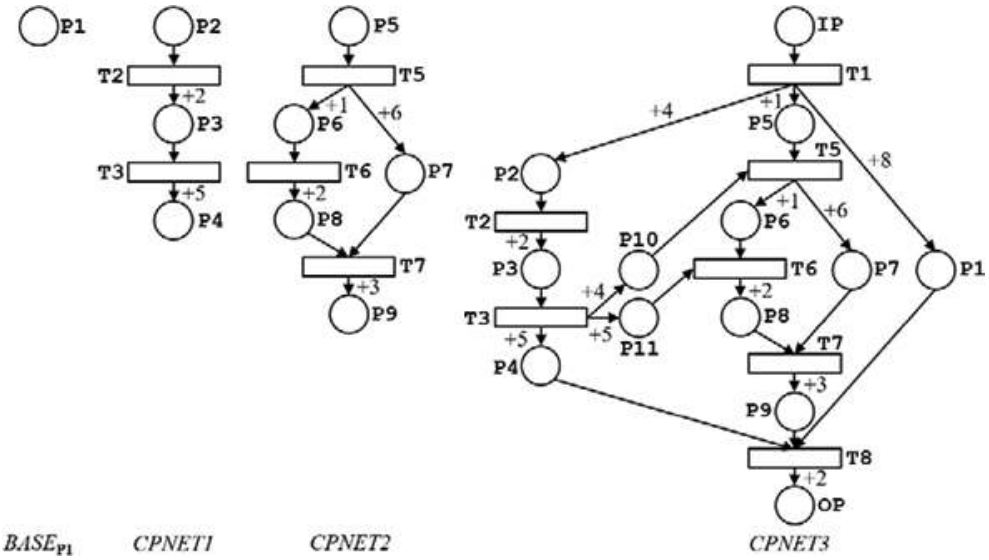

Figure 5. PNTSs $BASE_{\mathbf{P1}}$, *CPNET*1, *CPNET*2 and *CPNET*3.

Proof. Clear, it directly follows from Definition 5, Definition 7, Definition 10 and Definition 11 and from the fact that any $PNTS \in \boldsymbol{CPNET}$ does not contain any resource place (i.e., if the given *PNTS* is proper-formed PNTS, then it is also immediately well-formed and pure-formed PNTS). Furthermore, it is also clear that if we allow the existence of a circuit within the PNTS *PNTS* (see (iv) of Definition 11), there is always the danger of a deadlock in such a PNTS and *PNTS* is in this case neither a proper-formed PTSN. See, for instance, PNTS *CPNET*4 in its entry state S_e in **Figure 6**, where *CPNET*4 := *CPNET*3.**JOIN**(**P12**, **T7**, **T2**, 6). It is true that *CPNET*4 $\notin$ ***CPNET*** because there exists for instance the circuit.

$$\mathbf{T2\ P3\ T3\ P10\ T5\ P6\ T6\ P8\ T7\ P12} \in \boldsymbol{CIRCUITS}_{CPNET4}.$$

It is also clear that after the firing of the transition **T1** in the entry state S_e of the PNTS *CPNET*4 no one transition will be enabled for any value of the net time τ in this PNTS (i.e., there exists the deadlock marking in this PNTS) and thus the *CPNET*4 is not even proper-formed PNTS. □

Definition 12. The class $\boldsymbol{CPPNET} \subset \boldsymbol{PPNTS}$ contains all the PPNTSs $PPNTS := (PNTS, S_e)$ where $PNTS \in \boldsymbol{CPNET}$ and $S_e := ((1, 0, \ldots, 0), (<0>, <>, \ldots, <>), 0)$. □

A_n example of a simple process is presented in the following paragraphs the characteristics of which will be studied with using of the PPNTS from the CPPNET class. The studied process is described in the following table of activities (see **Table 1**):

The CPM chart of the abovementioned process comprising the activities listed in **Table 1** is shown in **Figure 7** where:

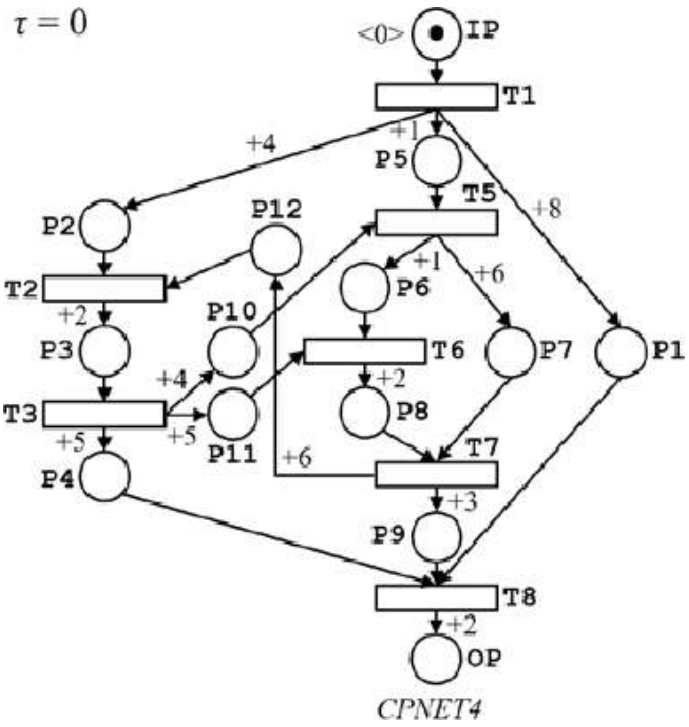

Figure 6. PNTS *CPNET4* in its entry state S_e.

Activity	Duration	Previous activities
A	2	—
B	3	—
C	4	—
D	2	C
E	6	B, D
F	5	A
G	5	C
H	3	E, F

Table 1. Table of activities and their dependencies of studied process.

- selected activity of the studied process and its duration is associated with each edge of the CPM chart,
- each node of the CPM chart is associated with its serial number (indicated in the upper half of the node), the earliest possible activation time of the given node (shown in the

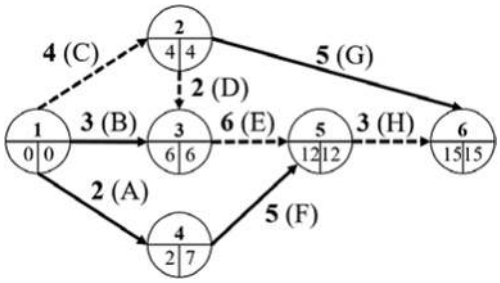

Figure 7. CPM chart of process activities listed in **Table 1**.

lower left quarter of the node) and the possible latest activation time of the given node (shown in the lower right quarter circle of the node),

- the desired links of the individual activities are expressed through the oriented edges and their associated nodes in the CPM chart,
- the critical path of the CPM chart passes through its nodes where the earliest possible activation time is equal to the latest possible activation time, that is, through nodes 1, 2, 3, 5 and 6, and it is thus formed by A, D, E and H activities (these activities are represented by dashed line graphs in the CPM chart) with a total duration of 15 time units.

The pure-formed PPNTS *PROC* that represents the process comprising the activities listed in **Table 1** can be seen in **Figure 8**, where

- *PROC* := [*PROC*1, [$BASE_{\mathbf{C}}$, $BASE_{\mathbf{G}}$].**COMP**(**T5**, 5)].**SYNC**(**IP**, **OP**, **T1**, **T7**, 1, 1, 0, 4, 0),
- *PROC*1 := [[$BASE_{\mathbf{A}}$, $BASE_{\mathbf{F}}$].**COMP**(**T3**, 5), [$BASE_{\mathbf{B}}$, $BASE_{\mathbf{E}}$].**COMP**(**T4**, 6)]

.**SYNC**(**P1**, **H**, **T2**, **T6**, 1, 1, 2, 3, 3).

Places **A**, **B**, **C**, **D**, **E**, **F**, **G** and **H** of PNTS *PROC* represent individual activities of the studied process and the appropriate values of the time interval function *TI* then express the time durations of relevant activities (i.e., for instance, the time duration of the activity **A** is represented by the value of *TI*(**T2**, **A**) = 2, etc.).

In order to find the critical path of the process represented by PPNTS *PROC*, we first perform the association of each place and transition of the PPNTS *PROC* with the value of the critical path function *CP* that is introduced in the following Definition 13.

Definition 13. Let PPNTS *PPNTS* := (P, T, A, AF, TP, TI, IP, OP, RP, S_e), *PPNTS* ∈ ***CPPNET***. The **critical path function** CP: $(P \cup T) \rightarrow \mathbf{N}_0$ is defined as follows:

i. $CP(IP) := 0$,

ii. $\forall p \in (P \setminus IP)$: $CP(p) := CP(t) + TI(t, p)$, where $t = {}^\bullet p$,

iii. $\forall t \in T$: $CP(t) := max(\{CP(p_1), CP(p_2), \ldots, CP(p_n)\})$, where ${}^\bullet t = \{p_1, p_2, \ldots, p_n\}$, $n \in \mathbf{N}$. □

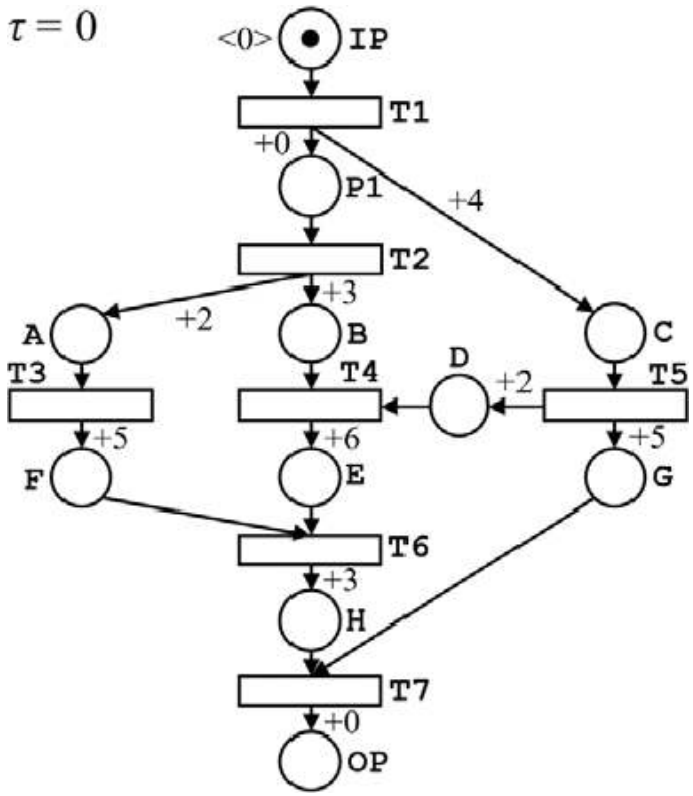

Figure 8. PPNTS *PROC* of process activities listed in **Table 1**.

The PPNTS *PROC* whose each place and transition is associated with the value of the critical path function *CP* can be seen in **Figure 9** (where, for instance, *CP*(**E**) = *CP*(**T4**) + *TI*(**T4**, **E**) = 6 + 6 = 12, where **T4** = •**E**; *CP*(**T4**) = *max*({*CP*(**B**), *CP*(**D**)}) = *max*({3, 6}) = 6, where •**T4** = {**B**, **D**}, etc.).

It follows directly from the Definition 4 that the value of the critical path function *CP* associated with any transition $t \in T$ of the arbitrary *PPNTS* := (*P*, *T*, *A*, *AF*, *TP*, *TI*, *IP*, *OP*, *RP*, S_e), where S_e := ((1, 0, …, 0), (<0>, <>, …, <>), 0), then represents the net time τ value when the given transition *t* will be fired. The value of the critical path function *CP* associated with the output place *OP* (i.e., *CP*(*OP*)) then immediately indicates the total duration of the process critical path (i.e., the net time τ value when the transition *t* = •*OP* will be fired). The algorithm for finding the set of PPNTS nodes of which the project critical path is formed is then obvious and it is expressed by the following pseudocode in PASCAL (the set of nodes forming the critical path of the project is then contained in the **CriticalPath** variable):

Node := *OP*; **CriticalPath** := {*OP*};

WHILE (**Node** <> *IP*) DO

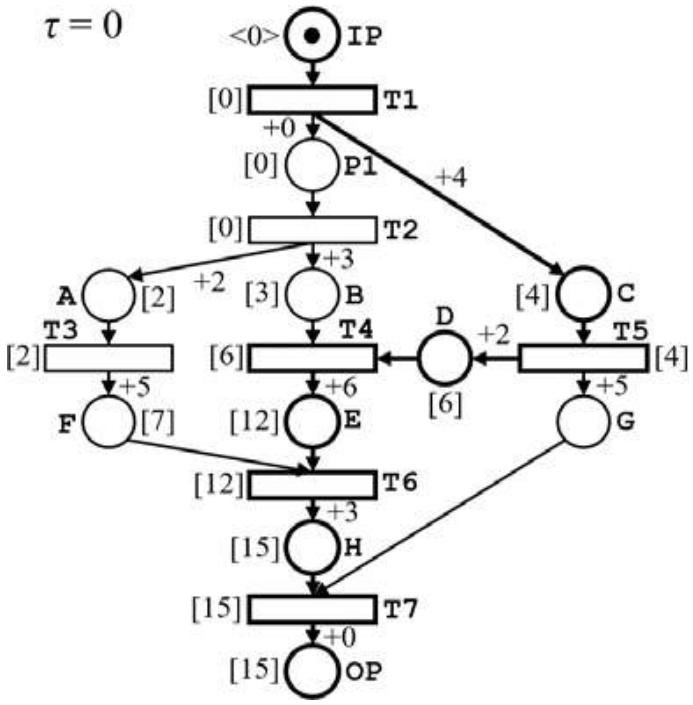

Figure 9. PPNTS *PROC* with the associated values of critical path function *CP*.

BEGIN

MaxValue := $max(\{CP(X_1), CP(X_2), \ldots, CP(X_n)\})$, where $\bullet$**Node** = $\{X_1, X_2, \ldots, X_n\}$, $n \in \boldsymbol{N}$;

Node := X_i, where $(X_i \in \{X_1, X_2, \ldots, X_n\}) \wedge (CP(X_i) =$ **MaxValue**$)$;

CriticalPath := **CriticalPath** $\cup$ {**Node**};

END;

The critical path of PPNTS *PROC* is after applying of the above algorithm represented by the set **CriticalPath** := {**IP**, **T1**, **C**, **T5**, **D**, **T4**, **E**, **T6**, **H**, **T7**, **OP**} (see **Figure 9**). It is also clear that the given PPNTS may contain more critical paths with the same total time duration.

5. Conclusions

Further research in the field of PNTSs is mainly focused on the definition of additional unary, binary and *n*-ary PPPA operators preserving their specified properties, for instance, the binary **SUBST** operator that performs the substitution of the given PNTS for the selected place of another PNTS, and so on. In the field of the project management the research is focused on

modeling complex processes, which individual activities can additionally share in parallel a selected set of the resources. These resources are then represented in the given PNTS by individual tokens located in the resource places of its selected net marking. Finding the time-optimal critical path of such a process as well as verifying the properties of the given PNTS that models such a process is generally a nontrivial problem and the use of PPPAs plays a crucial role here.

Another class currently being studied is the class of multiprocess Petri nets with time stamps that represents the generalization of the class of PNTS. The given multiprocess Petri net with time stamps then represents the finite set of processes each of which is modeled by a separate PNTS that share a common set of resources modeled by its individual resource places and their tokens. Many of the studied properties of the multiprocess Petri nets with time stamps are similar or identical to those of the PNTS class and they allow for a further generalization of the concept of a critical path formed by a sequence of activities of the process modeled by the given multiprocess Petri nets with time stamps.

Acknowledgements

This chapter was supported by the SGS at the Faculty of Economics, VŠB-TU Ostrava, under Grant Evaluation of comparative applications using cognitive analysis and method of data envelopment analysis, number SP2018/146.

Author details

Ivo Martiník

Address all correspondence to: ivo.martinik@vsb.cz

VŠB-Technical University of Ostrava, Ostrava, Czech Republic

References

[1] David R, Alla H. Discrete, Continuous and Hybrid Petri Nets. 2nd ed. Berlin: Springer-Verlag; 2010. 550 p. ISBN: 978-3642424694

[2] Reisig W. Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets. 1st ed. Berlin: Springer-Verlag; 1998. 302 p. ISBN: 3-540-62752-9

[3] Diaz M. Petri Nets: Fundamental Models, Verification and Applications. 1st ed. London: John Willey & Sons; 2009. 656 p. ISBN: 978-0-470-39430-4

[4] Popova-Zeugmann L. Time and Petri Nets. 1st ed. Heidelberg: Springer-Verlag; 2013. 209 p. ISBN: 978-3662514351

[5] Furia CA, Mandrioli D, Morzenti A, Rossi M. Modeling Time in Computing. 1st ed. Heidelberg: Springer-Verlag; 2012. 424 p. ISBN: 978-3642323317

[6] van Hee K, Sidorova N. The right timing: Reflections on the modeling and analysis of time. In: Application and Theory of Petri Nets and Concurrency 2013, 34th International Conference Proceedings; 24–28 June 2013; Milan. Berlin: Springer-Verlag; 2013. pp. 1-20

[7] Martos-Salgado M, Rosa-Velardo F. Dynamic networks of timed Petri nets. In: Application and Theory of Petri Nets and Concurrency 2014, 35th International Conference Proceedings; 23–27 June 2014; Tunis. Berlin: Springer-Verlag; 2014. pp. 294-313

[8] van der Alst W, van Hee K. Workflow Management: Models, Methods and Systems. 1st ed. Massachusetts: MIT Press; 2004. 384 p. ISBN: 978-0262720465

[9] Huang H, Jiao L, Cheung T, Mak WM. Property-Preserving Petri Net Process Algebra in Software Engineering. 1st ed. Singapore: World Scientific Publishing; 2012. 318 p. ISBN: 978-981-4324-28-1

[10] O'Brien JJ, Plotnick FL. CPM in Construction Management. 8th ed. New York: McGraw-Hill; 2016. 736 p. ISBN: 978-1259587276