Journal of Universal Computer Science, vol. 11, no. 2 (2005), 303-326
submitted: 31/7/04, accepted: 15/12/04, appeared: 28/2/05 © J.UCS

An Application of the DEDS Control Synthesis Method

František Čapkovič
(Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia
Frantisek.Capkovic@savba.sk)

Abstract: An application of the method suitable for modelling and control of general discrete event dynamic systems (DEDS) to special kinds of communication systems is presented in this paper. The approach is based on Petri nets (PN) defined in [Peterson 1981] and directed graphs (DG) described e.g. in [Diestel 1997]. It is supported by the previous author's works, especially [Čapkovič 2003]-[Čapkovič 1998], [Tzafestas and Čapkovič 1997].
Key Words: control synthesis, discrete event dynamic systems (DEDS)
Category: H.4.3, I.6.3, I.6.4, G.2.3

1 Introduction

DEDS are the systems driven by discrete events. Thus, the DEDS dynamics development strongly depends on the occurency and ordering of the discrete events. The appropriate sequence of the discrete events can considerably modify the DEDS behaviour. There are two kinds of discrete events in DEDS - spontaneous events (they cannot be influenced from without) and controllable ones (they can be forced to the system). By means of the latter events the system behaviour can be modified from without. Typical representants of DEDS are flexible manufacturing systems (FMS), communication systems, transport systems, etc. Because different processes concerning Web and/or multiagent systems are special kinds of communication systems, the modelling and control methods suitable for DEDS in general can be applied to modelling and control of them. PN are frequently used for DEDS modelling and control. To utilize PN here let us introduce the definition of the PN structure and dynamics. It is a little different from that presented in [Peterson 1981] in order to obtain the PN-based model of DEDS in the form of the classical linear discrete system (containing vectors and matrices). As to the structure PN are bipartite directed graphs with two kinds of nodes (places and transitions) and two kinds of edges (oriented arcs from the places to the transitions and oriented arcs from the transitions to the places). Formally, the PN structure is the quadruplet $\langle P, T, F, G \rangle$, $P \cap T = \emptyset$, $F \cap G = \emptyset$. Here, $P = \{p_1, p_2, ..., p_n\}$ is a finite set of the PN places p_i , $i = 1, ..., n$; $T = \{t_1, t_2, ..., t_m\}$ is a finite set of the PN transitions t_j , $j = 1, ..., m$; $F \subseteq P \times T$ is a set of the oriented arcs emerging from the places and entering the transitions. $G \subseteq T \times P$ is a set of the oriented arcs emerging from the transitions and entering the places. $\times$ symbolizes the Cartesian product. $\emptyset$ is an empty set. F can be expressed by means of the arcs incidence matrix $\mathbf{F} = \{f_{ij}\}_{n \times m}$, $f_{ij} \in \{0, M_{f_{ij}}\}$, $i = 1, ..., n$, $j = 1, ..., m$. f_{ij} represents the absence (when 0) or presence and multiplicity (when $M_{f_{ij}} > 0$) of the arc oriented from the place p_i to its output transition t_j. G can be expressed by means of the arcs incidence matrix $\mathbf{G} = \{g_{ij}\}_{m \times n}$, $g_{ij} \in \{0, M_{g_{ij}}\}$, $i = 1, ..., m$, $j = 1, ..., n$. g_{ij} represents the absence (when 0) or presence and multiplicity (when $M_{g_{ij}} > 0$) of the arc oriented from the transition t_i to its output place p_j. Furthermore, consider formally the PN dynamics to be the quadruplet $\langle X, U, \delta, \mathbf{x}_0 \rangle$, $X \cap U = \emptyset$ where $X = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of the PN state vectors, $U = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_N\}$ is the set of the PN control vectors, $\delta : P \times T \longrightarrow P$ is the transition function of PN and $\mathbf{x}_0$ is the PN initial state vector. Here $\mathbf{x}_k = (\sigma^k_{p_1}, ..., \sigma^k_{p_n})^T$, $k = 0, ..., N$ with $\sigma^k_{p_i} \in \{0, c_{p_i}\}$, $i = 1, ..., n$ being the states of the PN elementary places in the step k - i.e. 0 (passivity) or $0 < \sigma^k_{p_i} \le c_{p_i}$ (activity - in [Peterson 1981] denoted as marking) where c_{p_i} is an integer representing the capacity of the PN place p_i as to its activities (i.e. maximal number of marks) and $(.)^T$ symbolizes the matrix or vector transposition. $\mathbf{u}_k = (\gamma^k_{t_1}, ..., \gamma^k_{t_m})^T$, $k = 0, ..., N$ with $\gamma^k_{t_j} \in \{0, 1\}$, $j = 1, ..., m$ being the states of the PN elementary transitions in the step k - i.e. 1 (enabled) or 0 (disabled). Fuzzy interpretation is not cogitaded here. δ expresses formally the origin of the new states of PN places.

To create the analytical PN-based model of DEDS in the form of the linear discrete system the following analogies are done: 1) the elementary subprocesses of the system are represented by the set P of the PN places; 2) the discrete events are represented by the set T of the PN transitions; 3) the incidence matrices **F**, **G**, respectively, describe the causal relations between the subprocess and discrete events, and vice-versa; 4) the state vector of the system with integer components describes the states of the DEDS subprocesses and replaces the classical PN marking (defined e.g. in [Peterson 1981]); 5) the states of the elementary PN transitions are understood here to be the components of the control vector and express the occurrency of discrete events; 6) the formal transiton function δ is replaced by the linear discrete system having the form as follows

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}.\mathbf{u}_k \quad , \quad k = 0,\, 1,\, ...,\, N \tag{1}$$

$$\mathbf{B} = \mathbf{G}^T - \mathbf{F} \tag{2}$$

$$\mathbf{F}.\mathbf{u}_k \le \mathbf{x}_k \quad , \quad k = 0,\, 1,\, ...,\, N \tag{3}$$

in which the condition (3) has to be simultaneously met in order to avoid negative integers among the state vector components.

Simultaneously, we will utilize the DG-based model too. Namely, DG is here, as a matter of fact, the reachability tree (RT) and/or the reachability graph (RG) of the original PN. The DG structure can be expressed in mathematical terms as $\langle \mathit{\Pi}, \mathit{\Gamma}_k \rangle$ where $\mathit{\Pi} = \{\pi_1,\, \pi_2,\, ...,\, \pi_{n_{RT}}\}$ is the set of the DG nodes and $\mathit{\Gamma}_k \subseteq \mathit{\Pi} \times \mathit{\Pi}$ is the set of DG edges. In addition to this, the edge oriented from the node π_i to the node π_j includes also the PN transition with the transition function $\gamma^{(k)}_{t_{\pi_i|\pi_j}} \in \{0, 1\}$. When the transition is fired $\gamma^{(k)}_{t_{\pi_i|\pi_j}} = 1$ otherwise $\gamma^{(k)}_{t_{\pi_i|\pi_j}} = 0$. Owing to firing the transition in a step k the system passes from the state $\sigma^{(k)}_{\pi_i}(\gamma)$ (γ symbolizes the dependency on the transition function) to the state $\sigma^{(k+1)}_{\pi_j}(\gamma)$. The DG adjacency matrix $\mathbf{A}_k = \{\gamma^{(k)}_{t_{\pi_i|\pi_j}}\}_{n_{RT} \times n_{RT}}$ is the functional one because its nonzero elements are functions. Namely, the element $\gamma^{(k)}_{t_{\pi_i|\pi_j}}$ expresses the state of the transition assigned to the arc oriented from the DG node π_i to the DG node π_j in the step k. Its values change in separate steps. Therefore, the DG dynamics is described by the following equation

$$\mathbf{X}(k+1) = \Delta_k.\mathbf{X}(k) \ , \quad k = 0, 1, ..., N \tag{4}$$

where $\mathbf{X}(k) = (\sigma^{(k)}_{\pi_1}(\gamma), ..., \sigma^{(k)}_{\pi_{n_{RT}}}(\gamma))^T$ $k = 0, 1, ..., N$ is the n_{RT}-dimensional state vector of the DG in the step k; $\sigma^{(k)}_{\pi_i}(\gamma), \in \{0, 1\}$, $i = 1, ..., n_{RT}$ is the state of the elementary DG node π_i in the step k. Its value depends on actual enabling its input transitions; $\Delta_k = \{\delta^{(k)}_{ij}\}_{n_{RT} \times n_{RT}}$, $\delta^{(k)}_{ij} = \gamma^{(k)}_{t_{\pi_j|\pi_i}}$, $j = 1, n_{RT}$, $i = 1, n_{RT}$, i.e. $\Delta_k = \mathbf{A}^T_k$. It is the functional matrix where $\delta^{(k)}_{ij} \in \{0, 1\}$ is the transition function of the PN transition fixed on the edge oriented from the node π_j of the DG to the node π_i of the DG. However, it is necessary to say that the places p_i, $i = 1, ..., n$ of PN are completely different form the nodes π_i, $i = 1, ..., n_{RT}$ of the DG. While the PN places represent the states of elementary activities inside PN, the DG nodes represent the complete state vectors of the PN. It corresponds with the fact that the DG is the RG of the PN. The DG represents the alternative artificial fictive state machine (SM) with nodes π_i , $i = 1, ... n_{RT}$ representing the reachable state vectors of the PN. Namely, in the PN theory SM is a class of PN where any transition has only single input place and only single output one.

In [Čapkovič 2003], [Čapkovič and Čapkovič 2003] the procedure enumerating the *quasi-functional* adjacency matrix $\mathbf{A} \neq \mathbf{A}_k$ of the RG and the space of the PN reachable states in the form of the matrix $\mathbf{X}_{reach}$ was presented in different depth. In $\mathbf{A}$ the indices of the PN transitions are stored while the columns of $\mathbf{X}_{reach}$ are the PN state vectors $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_{n_{RT}-1}$. The inputs of the procedure are the PN structural matrices $\mathbf{F}$, $\mathbf{G}^T$ and the PN initial state vector $\mathbf{x}_0$. While the PN-based model in general (where any transition can have more than one input places as well as more than one output places) cannot be understood to be the classical SM (because of synchronization problems), the DG-based model (where DG is RG of the PN in question) is the classical SM.

1.1 The example 1

To illustrate the operation of the above mentioned procedure let us model the client-server cooperation as simply as possible. There can be distinguished the following principal partial activities expressed by PN places: p_1 = the client requests for the connection with the server; p_2 = the server is listening; p_3 = the connection of the client with the server; p_4 = sending data by the client to the server is realized; p_5 = the disconnection of the client by the client himself. The PN representing the problem is given in Fig. 1.a. The PN transitions t_1 - t_3 represent discrete events that realize the system dynamics. The order of their

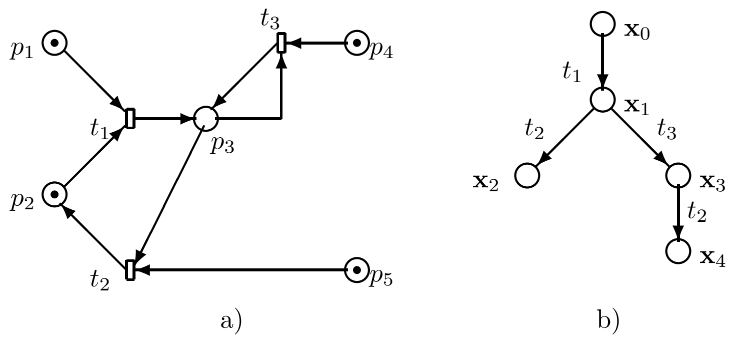


Figure 1: a) The PN-based model of the client-server cooperation; b) The corresponding RG

occurrency influences the actual system dynamics development. The inputs of the procedure - i.e. the structural matrices **F**, **G** of the mathematical model and the initial state vector $\mathbf{x}_0$ - are the following as well as the outputs - i.e. the *quasi-functional* adjacency matrix **A** of the RG (in Fig. 1.b), the corresponding transpose of the functional matrix Δ_k, and the state space of its reachable states $\mathbf{X}_{reach}$

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} ; \ \mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} ; \ \mathbf{x}_0 = (1,\ 1,\ 0,\ 1,\ 1)^T$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} ; \ \Delta_k^T = \begin{pmatrix} 0 & \gamma_{t_1} & 0 & 0 & 0 \\ 0 & 0 & \gamma_{t_2} & \gamma_{t_3} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma_{t_2} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} ; \ \mathbf{X}_{reach} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

As we can see, the nonzero elements of **A** represent the indices of the PN transitions. The elements of Δ_k^T are represented by the transition functions $\gamma_{t_i}^k$ of the transitions t_i, $i = 1, 2, 3$. In this simple example the control synthesis is simple too. After occurrency of the discrete event represented by the transition t_1 the client is connected with the server (the state $\mathbf{x}_1$). Now the client has two possibilities - to disconnect once more (by the event t_2 to the state $\mathbf{x}_2$) or to send data to the server (by t_3 to $\mathbf{x}_3$). After sending data the client can work on the server and after finishing the work he can disconnect (by t_2 to $\mathbf{x}_4$). In more complicated cases (with more places and transitions and/or with more complicated structure of PN) it is necessary to perform the automatic control synthesis.

2 The DG-based control synthesis

The problem of control in general is the following: to transform the system to be controlled from a given initial state $\mathbf{x}_0$ to a prescribed terminal state $\mathbf{x}_t$ at simultaneous fulfilling the prescribed control task specifications (like criteria, constraints, etc.). For DEDS control synthesis the very simple idea can be utilized. Consider a system being in the initial state $\mathbf{x}_0$. Consider the desirable terminal state $\mathbf{x}_t$. Develop the straight-lined RT (SLRT) from the state $\mathbf{x}_0$ towards $\mathbf{x}_t$ directed to $\mathbf{x}_t$. Develop the backtracking RT (BTRT) from the state

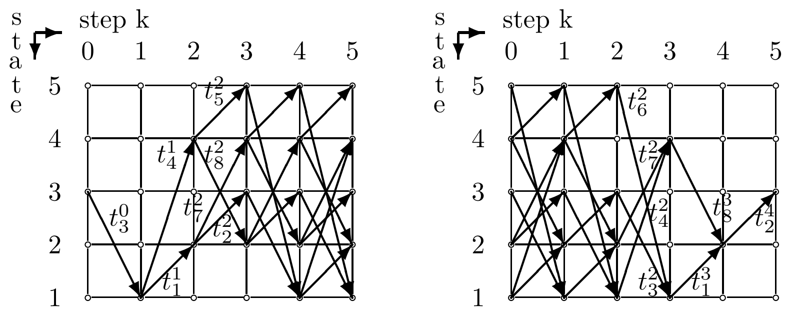


Figure 2: The straight-lined system development from the initial state (on the left) and the backtracking system development from the terminal state (on the right)

$\mathbf{x}_t$ towards $\mathbf{x}_0$ however, directed to $\mathbf{x}_t$. Intersect both the SLRT and BTRT. The trajectory (or several ones) starting from $\mathbf{x}_0$ and finishing in $\mathbf{x}_t$ is (are) obtained. To illustrate such an approach see Fig. 2 (where SLRT is given on the left and BTRT on the right) and Fig. 3 where the resulting 5 steps trajectory is given on the left. When BTRT is shifted for one step to the left before the intersection with SLRT the 4 steps trajectory given in the center in Fig. 3 is obtained. When BTRT is shifted for two steps to the left before the intersection with SLBT the

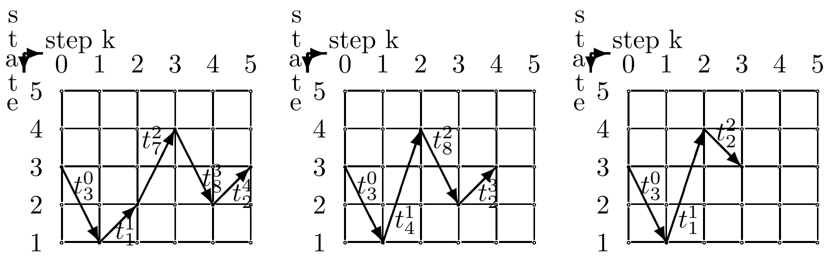


Figure 3: The intersection of the length of 5 steps (on the left), 4 steps (in the centre), and 3 steps (on the right)

3 steps trajectory given on the right in Fig. 3 is obtained. Although in general there can be more than one trajectory after intersection in any case we obtain the feasible possibilities of the system behaviour between the $\mathbf{x}_0$ and $\mathbf{x}_t$. When a criterion for optimality is given we can find even the optimal trajectory. The system behaviour can also be adapted to external conditions or demands by choosing the suitable trajectory from the feasible ones.

2.1 The procedure of the control synthesis

To avoid problems with symbolic operations at computer handling Δ_k in (4) we will understand all of the transitions to be enabled (i.e. their transition functions having the values 1). In such a way we can replace the functional matrix Δ_k by the constant matrix Δ. Thus, the DG-based approach operates with the matrix Δ being the transpose of the classical constant adjacency matrix of the RG (representing the DG and/or SM corresponding to the original PN). The constant RG adjacency matrix can be obtained from the *quasi-functional* adjacency matrix $\mathbf{A}$ by means of replacement all of the nonzero elements by the integer 1. Hence, the SLBT can be constructed in analytical terms as follows

$$^1\{\mathbf{X}_1\} = \Delta.\mathbf{X}_0 \tag{5}$$

$$^1\{\mathbf{X}_2\} = \Delta.^1\{\mathbf{X}_1\} = \Delta.(\Delta.\mathbf{X}_0) = \Delta^2.\mathbf{X}_0 \tag{6}$$

$$\dots \qquad \dots \qquad \dots$$

$$^1\{\mathbf{X}_N\} = \Delta.^1\{\mathbf{X}_{N-1}\} = \Delta^N.\mathbf{X}_0 \tag{7}$$

In general, $^1\{\mathbf{X}_j\}$ is an aggregate all of the states that are reachable from the previous states. According to graph theory $N \leq (n_{RT}-1)$. Really, the procedure can be finished when the prescribed terminal state $\mathbf{X}_t \leq {}^1\{\mathbf{X}_N\}$ (namely, when $\mathbf{X}_t > {}^1\{\mathbf{X}_N\}$, the vector $\mathbf{X}_t$ is not achieved yet). However, theoretically an arbitrary number can be chosen to be N while $\mathbf{X}_t \leq {}^1\{\mathbf{X}_N\}$, i.e. the SLRT can be developed without any stint. Now (i.e. having N), $\mathbf{X}_t$ can be denoted as $\mathbf{X}_N$. Afterwords, the BTRT is developed from $\mathbf{X}_N$ towards $\mathbf{X}_0$, however, it contains the paths oriented towards the terminal state. It is the following

$$^2\{\mathbf{X}_{N-1}\} = \Delta^T.\mathbf{X}_N \tag{8}$$

$$^2\{\mathbf{X}_{N-2}\} = \Delta^T.^2\{\mathbf{X}_{N-1}\} = (\Delta^T)^2.\mathbf{X}_N \tag{9}$$

$$\dots \quad \dots \quad \dots$$

$$^2\{\mathbf{X}_0\} = \Delta^T.^2\{\mathbf{X}_1\} = (\Delta^T)^N.\mathbf{X}_N \tag{10}$$

Here, $^2\{\mathbf{X}_j\}$ is an aggregate all of the states from which the next states are reachable. It is clear that $\mathbf{X}_0 \neq {}^2\{\mathbf{X}_0\}$ and $\mathbf{X}_N \neq {}^1\{\mathbf{X}_N\}$. It is the consequence of the fact that in general, $\Delta.\Delta^T \neq \mathbf{I_n}$ as well as $\Delta^T.\Delta \neq \mathbf{I}_n$ (where $\mathbf{I}_n$ is $(n \times n)$ identity matrix). The intersection of the trees is made as follows

$$\mathbf{M}_1 = (\mathbf{X}_0, {}^1\{\mathbf{X}_1\}, \dots, {}^1\{\mathbf{X}_{N-1}\}, {}^1\{\mathbf{X}_N\}) \tag{11}$$

$$\mathbf{M}_2 = ({}^2\{\mathbf{X}_0\}, {}^2\{\mathbf{X}_1\}, \dots, {}^2\{\mathbf{X}_{N-1}\}, \mathbf{X}_N) \tag{12}$$

$$\mathbf{M} = \mathbf{M}_1 \cap \mathbf{M}_2 \tag{13}$$

$$\mathbf{M} = (\mathbf{X}_0, \{\mathbf{X}_1\}, \dots, \{\mathbf{X}_{N-1}\}, \mathbf{X}_N) \tag{14}$$

where the matrices $\mathbf{M}_1$, $\mathbf{M}_2$ represent, respectively, the SLRT and the BTRT. The special intersection both of the trees is performed by means of the column-to-column intersection both of the matrices. Thus, $\{\mathbf{X}_i\} = \min({}^1\{\mathbf{X}_i\}, {}^2\{\mathbf{X}_i\})$, $i = 0, ..., N$ with ${}^1\{\mathbf{X}_0\} = \mathbf{X}_0$, ${}^2\{\mathbf{X}_N\} = \mathbf{X}_N$. Due to the principle of causality any *shorter* feasible solution is involved in the *longer* feasible one. It can be found when $\mathbf{M}_2$ is shifted to the left for one or more columns before the intersection with $\mathbf{M}_1$. When the intersection (13) is the zero matrix the solution of the corresponding length does not exist. It means that choosing $N = n_{RT} - 1$ ensures finding all of the solutions by means of the shifting process.

2.2 The example 2

Let us model a simple cooperation of two agents A and B by PN. The agent A needs to do an activity P, however, it is not able to do it. Therefore, A requests B to do P for him. On the base of the conversation between the agents, P is either done (if B is willing to do P and it is able to do P) or not (when B refuses to do P or it is willing to do P however, it is not able to do P). To create the PN-based model of such a system, let us consider that the places express the activities of the agents and the messages beeing routed between them as follows: p_1 = the agent A wants to do P however, it is not able to do P; p_2 = A waits for an answer from B; p_3 = A waits for a help from B; p_4 = the failure of the cooperation; p_5 = the satisfaction of the cooperation; p_6 = A requests B to do P; p_7 = B refuses to do P; p_8 = B accepts the request of A to do P; p_9 = B is not able to do P; p_{10} = doing P by B; $p_{11} = B$ receives the request of A; $p_{12} = B$ is willing to do P; p_{13} = the end of the work of B. The PN transitions correspond either to synchronizations due to the receipt of a message or to conditions of the application of actions. The PN-based model is given in Fig. 4.a. The PN structural matrices and the initial state vector are as follows

$$\mathbf{F}^T = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 0&1&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&0&1 \end{pmatrix}$$

$$\mathbf{x}_0 = (1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0)^T$$

The corresponding RG is given in Fig. 4.b. The RG *quasi-functional* adjacency

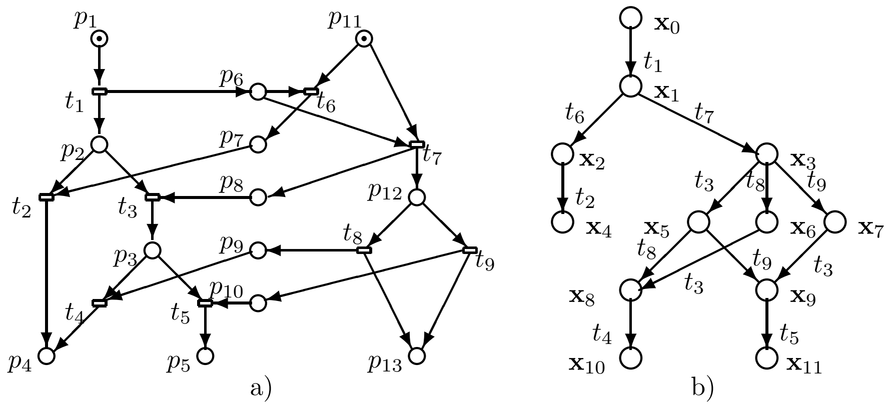


Figure 4: a) The PN-based model of the cooperation of two agents A and B solving the problem P; b) The RG corresponding to the model and to the initial state vector $\mathbf{x}_0$

matrix $\mathbf{A}$ and the space of the PN reachable states represented by the matrix $\mathbf{X}_{reach}$ are introduced below as well as the matrix $\mathbf{M}$ containing the state trajectories resulting from the control synthesis procedure. The RG clearly expresses the mutual relations among the state vectors of the modelled DEDS reachable from the initial state $\mathbf{x}_0$. However, on the other hand, the mutual relations among the elementary activities or operations of DEDS remain hidden. The situation inside the PN-based model is opposite. The interactions among the elementary activities and operations are clearly expressed as well as the synchronization possibilities, but the relations among the states vectors are hidden. In such a way the PN-based model of DEDS and its RG complement each other. It is very important at testing the DEDS properties, at the control synthesis, etc.

$$\mathbf{A} = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 6 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 8 & 9 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 9 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$\mathbf{X}_{reach} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix} \quad \mathbf{M} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Creating the simple program in MATLAB realizing the procedure descibed by the equations (5) - (14) the proposed approach to the control synthesis can be tested very quickly. The trajectory given in matrix $\mathbf{M}$ represents the situation when the terminal state expressing the successful cooperation was desired $\mathbf{x}_N = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1)^T$. It is graphically demonstrated on the left in Fig. 5. When the terminal state is different, e.g. the failure of the cooperation when the B is not able to do P, $\mathbf{x}_N = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1)^T$ and the result is different too, of course - see the trajectory on the right in Fig. 5.

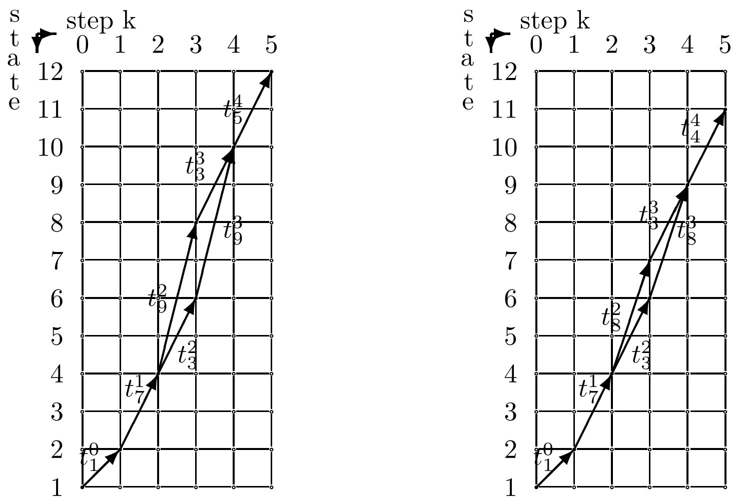


Figure 5: The resulting trajectories - in case of the successful cooperation (on the left) and in case of the failure of the cooperation when B is not able to do P (on the right)

3 The adaptivity

As to the applicability of the above described approach we can distinguish two kinds of the adaptivity.

On the one hand it is the adaptivity concerning the modifying of the system dynamics development by means of choosing a suitable trajectory from the set of feasible state trajectories obtained in the control synthesis process. Such a kind of the adaptivity can be clear e.g. from the left picture in Fig. 5 where two different feasible trajectories (expressing possibilities of the system behaviour) are presented. Because no other trajectory exists, either the trajectory $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_7, \mathbf{x}_9, \mathbf{x}_{11}\}$ corresponding to the enabling sequence of transitions $\{t_1^0, t_7^1, t_9^2, t_3^3, t_5^4\}$ or the trajectory $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_9, \mathbf{x}_{11}\}$ corresponding to the enabling sequence $\{t_1^0, t_7^1, t_3^2, t_9^3, t_5^4\}$ can be chosen in order to adapt the system behaviour to the actual demand.

On the other hand it is the adaptivity of the system behaviour by means of a structural fragment added to the original system structure. Such a fragment is able to accept demands (from without) on the system behaviour and realize them. The adaptivity is illustrated in Fig. 6. On the left the PN-based system model consisting of two processes and the structural fragment (consisting of the place p_4) is given. The fragment is able to influence the performance of the processes (the mutual exclusion, sequencing, re-run). The model parameters and the initial state are as follows

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} ; \mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} ; \mathbf{x}_0 = (1, 0, 0, 1, 1, 0, 0)^T$$

The *quasi-functional* adjacency matrix and the state space of the reachable states are the following

$$\mathbf{A}_k = \begin{pmatrix} 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} ; \mathbf{X}_{reach} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

In the centre of Fig. 6 the RG is presented while the result of the control synthesis process transforming the system from $\mathbf{x}_0 = (1, 0, 0, 1, 1, 0, 0)^T$ to $\mathbf{x}_6 = (0, 0, 1, 0, 0, 1, 0)^T$ is put on the right.

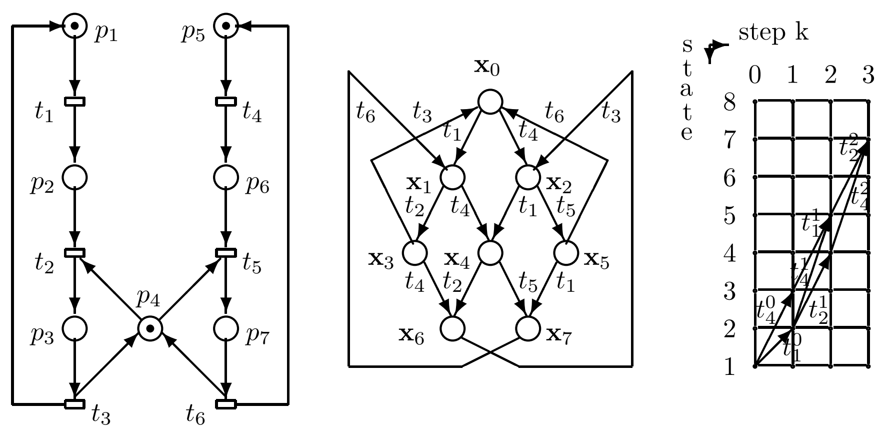


Figure 6: The PN-based model of the system behaviour (on the left), its RG (in the centre), and the feasible trajectories from $\mathbf{x}_0$ to $\mathbf{x}_6$ (on the right)

4 The BDG-based approach to SM control synthesis

The PN in general (SM as well) can be understood to be bipartite directed graphs (BDG) as follows

$$\langle S, \Delta \rangle \tag{15}$$

where $S = \{P, T\}$ is the set of the BDG nodes and $\Delta \subseteq S \times S$ is the set of BDG edges. Here $P = \{p_1, p_2, ..., p_n\}$, $T = \{t_1, t_2, ..., t_m\}$ are, respectively, the set of PN places and the set of PN transitions. The occurrence of the edges can be expressed by the $((n+m) \times (n+m))$ matrix

$$\Delta = \begin{pmatrix} \mathbf{\emptyset}_{n\times n} & \mathbf{G}^T \\ \mathbf{F}^T & \mathbf{\emptyset}_{m\times m} \end{pmatrix} \tag{16}$$

where $\mathbf{\emptyset}_{i\times j}$ in general is the $(i \times j)$ zero matrix; $\mathbf{G}$ is the $(m \times n)$ PN incidence matrix expressing $G \subseteq T \times P$; $\mathbf{F}$ is the $(n \times m)$ PN incidence matrix representing $F \subseteq P \times T$. Hence, BDG can be understood to be DG and (4) turns to

$$\{\mathbf{s}_{k+1}\} = \Delta.\{\mathbf{s}_k\}\ ,\ k = 0, 1, \dots, 2N-1 \tag{17}$$

with $\mathbf{s}_k$ being the augmented $(n + m)$-dimensional vector defined as follows

$$\{\mathbf{s}_k\} = \begin{cases} (\{\mathbf{x}_{k/2}\}^T, \mathbf{\emptyset}_m^T)^T & \text{if } k = 0,\ 2,\ 4,\ ...,\ 2N-2 \\ (\mathbf{\emptyset}_n^T, \{\mathbf{u}_{(k-1)/2)}\}^T)^T & \text{if } k = 1,\ 3,\ 5,\ ...,\ 2N-1 \end{cases} \tag{18}$$

where $\mathbf{\emptyset}_j$ in general is the j-dimensional zero vector; $\mathbf{x}_{k/2} = \mathbf{G}^T.\mathbf{u}_{(k-2)/2}$, $k = 2, 4, ..., 2N-2$; $\mathbf{u}_{(k-1)/2} = \mathbf{F}^T.\mathbf{x}_{(k-1)/2}$, $k = 1, 3, 5, ..., 2N-1$.

Now, the above-mentioned DG-based approach could be applied. However, because of the special block form of both the matrix Δ and the vector $\mathbf{s}_k$ we can alternate step-by-step two procedures with dimensionalities n, m, respectively. In such a way two matrices $\mathbf{X}$ and $\mathbf{U}$ with dimensionalities $(n\times(N+1))$, $(m\times N)$, respectively, are obtained instead of the matrix $\mathbf{M}$. They are the following

$$\mathbf{X} = (\mathbf{x}_0, \{\mathbf{x}_1\}, \dots, \{\mathbf{x}_{N-1}\}, \mathbf{x}_N) \tag{19}$$

$$\mathbf{U} = (\{\mathbf{u}_0\}, \{\mathbf{u}_1\}, \dots, \{\mathbf{u}_{N-1}\}) \tag{20}$$

4.1 The procedure in details

To explain the procedure better let us go to describe it mathematically in details. Understanding BDG to be DG and using the above described control synthesis procedure the straight-lined development is the following

$$\{\mathbf{s}_{k+1}\} = \Delta.\{\mathbf{s}_k\} \quad , \quad k = 0, 1, \dots, 2N-1 \tag{21}$$

$$\mathbf{s}_0 = (\mathbf{x}_0^T, \mathbf{\emptyset}_m^T)^T;\ \{\mathbf{s}_1\} = (\mathbf{\emptyset}_n^T, \{\mathbf{u}_0\}^T)^T;\ \{\mathbf{s}_2\} = (\{\mathbf{x}_1\}^T, \mathbf{\emptyset}_m^T)^T;$$

$$\{\mathbf{s}_3\} = (\mathbf{\emptyset}_n^T, \{\mathbf{u}_1\}^T)^T;\ \dots\ \{\mathbf{s}_{2N-1}\} = (\mathbf{\emptyset}_n^T, \{\mathbf{u}_{N-1}\}^T)^T;\ \{\mathbf{s}_{2N}\} = (\{\mathbf{x}_N\}^T, \mathbf{\emptyset}_m^T)^T$$

where $\emptyset_j$ in general is the j-dimensional zero vector. The SLRT arising from the initial vector $\mathbf{s}_0$ and directed towards the terminal state $\mathbf{s}_{2N}$ can be recorded by the matrix

$$\mathbf{M}_1 = (\mathbf{s}_0, {}^1\{\mathbf{s}_1\}, \dots, {}^1\{\mathbf{s}_{2N-1}\}, {}^1\{\mathbf{s}_{2N}\}) \tag{22}$$

The backtracking development is as follows

$$\{\mathbf{s}_{2N-k-1}\} = \Delta^T.\{\mathbf{s}_{2N-k}\}\ ,\ k = 0, 1, \dots, 2N-1 \tag{23}$$
$$\mathbf{s}_{2N} = (\mathbf{x}_N^T, \emptyset_m^T)^T;\ \{\mathbf{s}_{2N-1}\} = (\emptyset_n^T, \{\mathbf{u}_{N-1}\}^T)^T; \{\mathbf{s}_{2N-2}\} = (\{\mathbf{x}_{N-1}\}^T, \emptyset_m^T)^T;$$
$$\{\mathbf{s}_{2N-3}\} = (\emptyset_n^T, \{\mathbf{u}_{N-2}\}^T)^T;\ \dots \{\mathbf{s}_1\} = (\emptyset_n^T, \{\mathbf{u}_0\}^T)^T; \{\mathbf{s}_0\} = (\{\mathbf{x}_0\}^T, \emptyset_m^T)^T$$

The BTRT is recorded by

$$\mathbf{M}_2 = ({}^2\{\mathbf{s}_0\}, {}^2\{\mathbf{s}_1\}, \dots, {}^2\{\mathbf{s}_{2N-1}\}, \mathbf{s}_{2N}) \tag{24}$$

After the intersection (13) we have

$$\mathbf{M} = (\mathbf{s}_0, \{\mathbf{s}_1\}, \dots, \{\mathbf{s}_{2N-1}\}, \mathbf{s}_{2N}) \tag{25}$$

where $\{\mathbf{s}_i\}\ = \min({}^1\{\mathbf{s}_i\},\ {}^2\{\mathbf{s}_i\})$, $i = 0,\ 2N$ with ${}^1\{\mathbf{s}_0\} = \mathbf{s}_0$, ${}^2\{\mathbf{s}_{2N}\} = \mathbf{s}_{2N}$.

It is favourable that information about the state vectors is stored in the even columns of $\mathbf{M}$ while information about the control vectors is stored in its odd columns. In addition to this, the matrices $\mathbf{F}$ and $\mathbf{G}$ are directly the numerical constant PN incidence matrices expressing the relations $P \times T$ and $T \times P$, respectively. On the other hand the dimensionality of the matrix Δ is $((n+m) \times (n+m))$, while in the original method the dimensionality is only $(n \times n)$. Also the dimensionalities of $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}$ are greater than those in the original method. However, because of the special block form of both the matrix Δ and the vectors $\mathbf{s}_k$, $k = 0, 2N$ we can simultaneously use (alternate step-by-step) two procedures with dimensionalities n, m, respectively.

4.2 The decomposition of the procedure

What is peculiar in the above procedure is the step-by-step alternation of enumerating both the state variables and the control ones. In the sense of the previous section the straight-lined procedure is the following

$${}^1\mathbf{X} = (\mathbf{x}_0, \emptyset_{n\times N});\ {}^1\{\mathbf{u}_0\} = \mathbf{F}^T.\mathbf{x}_0 \tag{26}$$
$${}^1\mathbf{U} = ({}^1\{\mathbf{u}_0\}, \emptyset_{m\times(N-1)});\ {}^1\{\mathbf{x}_1\} = \mathbf{G}^T.{}^1\{\mathbf{u}_0\} \tag{27}$$
$${}^1\mathbf{X} = (\mathbf{x}_0, {}^1\{\mathbf{x}_1\}, \emptyset_{n\times(N-1)});\ {}^1\{\mathbf{u}_1\} = \mathbf{F}^T.{}^1\mathbf{x}_1 \tag{28}$$
$${}^1\mathbf{U} = ({}^1\{\mathbf{u}_0\}, {}^1\{\mathbf{u}_1\}, \emptyset_{m\times(N-2)});\ {}^1\{\mathbf{x}_2\} = \mathbf{G}^T.{}^1\{\mathbf{u}_1\} \tag{29}$$
$${}^1\mathbf{X} = (\mathbf{x}_0, {}^1\{\mathbf{x}_1\}, {}^1\{\mathbf{x}_2\}, \emptyset_{n\times(N-2)});\ \dots \tag{30}$$

$$\ldots \quad \ldots \quad \ldots \quad {}^1\{\mathbf{u}_{N-1}\} = \mathbf{F}^T.{}^1\mathbf{x}_{N-1} \tag{31}$$

$${}^1\mathbf{U} = ({}^1\{\mathbf{u}_0\}, {}^1\{\mathbf{u}_1\}, \ldots, {}^1\{\mathbf{u}_{N-1}\});\ {}^1\{\mathbf{x}_N\} = \mathbf{G}^T.{}^1\{\mathbf{u}_{N-1}\} \tag{32}$$

$${}^1\mathbf{X} = (\mathbf{x}_0, {}^1\{\mathbf{x}_1\}, {}^1\{\mathbf{x}_2\}, \ldots, {}^1\{\mathbf{x}_N\}) \tag{33}$$

where ${}^1\mathbf{U}$ is $(m \times N)$ matrix and ${}^1\mathbf{X}$ is $(n \times (N+1))$ matrix. The left upper index ${}^1(.)$ points out performing the straight-lined procedure. The backtracking procedure run as follows

$${}^2\mathbf{X} = (\emptyset_{n\times N}, \mathbf{x}_N);\ {}^2\{\mathbf{u}_{N-1}\} = \mathbf{G}.\mathbf{x}_N \tag{34}$$

$${}^2\mathbf{U} = (\emptyset_{m\times(N-1)}, {}^2\{\mathbf{u}_{N-1}\});\ {}^2\{\mathbf{x}_{N-1}\} = \mathbf{F}.{}^2\{\mathbf{u}_{N-1}\} \tag{35}$$

$${}^2\mathbf{X} = (\emptyset_{n\times(N-1)}, {}^2\{\mathbf{x}_{N-1}\}, \mathbf{x}_N);\ {}^2\{\mathbf{u}_{N-2}\} = \mathbf{G}.{}^2\{\mathbf{x}_{N-1}\} \tag{36}$$

$${}^2\mathbf{U} = (\emptyset_{m\times(N-1)}, {}^2\{\mathbf{u}_{N-2}\}, {}^2\{\mathbf{u}_{N-1}\});\ {}^2\{\mathbf{x}_{N-2}\} = \mathbf{F}.{}^2\{\mathbf{u}_{N-1}\} \tag{37}$$

$${}^2\mathbf{X} = (\emptyset_{n\times(N-2)}, {}^2\{\mathbf{x}_{N-2}\}, {}^2\{\mathbf{x}_{N-1}\}, \mathbf{x}_N);\ \ldots$$

$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad {}^2\{\mathbf{u}_1\} = \mathbf{G}.{}^2\{\mathbf{x}_2\} \tag{38}$$

$${}^2\mathbf{U} = (\emptyset_{n\times 1}, {}^2\{\mathbf{u}_1\}, \ldots, {}^2\{\mathbf{u}_{N-1}\});\ {}^2\{\mathbf{x}_1\} = \mathbf{F}.{}^2\{\mathbf{u}_1\} \tag{39}$$

$${}^2\mathbf{X} = (\emptyset_{n\times 1}, {}^2\{\mathbf{x}_1\}, {}^2\{\mathbf{x}_2\}, \ldots, \mathbf{x}_N);\ {}^2\{\mathbf{u}_0\} = \mathbf{G}.{}^2\{\mathbf{x}_1\} \tag{40}$$

$${}^2\mathbf{U} = ({}^2\{\mathbf{u}_0\}, {}^2\{\mathbf{u}_1\}, \ldots, {}^2\{\mathbf{u}_{N-1}\});\ {}^2\{\mathbf{x}_0\} = \mathbf{F}.{}^2\{\mathbf{x}_0\} \tag{41}$$

$${}^2\mathbf{X} = ({}^2\{\mathbf{x}_0\}, {}^2\{\mathbf{x}_1\}, {}^2\{\mathbf{x}_2\}, \ldots, \mathbf{x}_N) \tag{42}$$

where ${}^2\mathbf{U}$ is $(m \times N)$ matrix and ${}^2\mathbf{X}$ is $(n \times (N+1))$ matrix. The left upper index ${}^2(.)$ points out performing the backtracking procedure.

The final phase of the control problem solving consists in the special intersection described above. In such a way we have both the system state trajectories and corresponding control strategies

$$\mathbf{X} = {}^1\mathbf{X} \cap {}^2\mathbf{X} \tag{43}$$

$$\mathbf{X} = (\mathbf{x}_0, \{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_{N-1}\}, \mathbf{x}_N) \tag{44}$$

$$\mathbf{U} = {}^1\mathbf{U} \cap {}^2\mathbf{U} \tag{45}$$

$$\mathbf{U} = (\{\mathbf{u}_0\}, \{\mathbf{u}_1\}, \ldots, \{\mathbf{u}_{N-1}\}) \tag{46}$$

5 The implementation of the proposed approach

The procedure of the implementation of the above method to the PN is the following: 1) setting a given initial state vector $\mathbf{x}_0$ of the PN places to be the root node of the RT; 2) computing the *quasi-functional* adjacency matrix $\mathbf{A}$ of the RT developed from the root node as well as the PN state space. The state space consists of $\mathbf{x}_0$ and all of the state vectors reachable from $\mathbf{x}_0$. These feasible state vectors are stored as the columns of $\mathbf{X}_{reach}$. 3) understanding the RT to be the special DG described above; 4) creating RG in two steps: (i) mutual connecting the RT leaves (terminal nodes) occurring repeatedly (i.e. having the same name) into one node; (ii) connecting the obtained node with the nonterminal RT node with the same name; 5) understanding the RG to be SM; 6) utilizing one of the methods mentioned above, i.e. the DG-based method or the BDG-based one. However, the latter one was defined above only for SM. Therefore, it has to be modified in order to be applied to PN with general structure.

5.1 The algorithm generating RT

There exists the relatively simple algorithm for generating RT of PN with general structure. It is verbally described in [Peterson 1981] and formally expressed in details by many other authors. We need the parameters of RT - i.e. the functional adjacency matrix $\mathbf{A}_k$ and the feasible reachable states. A simple computer realization of the algorithm in the form of the MATLAB procedure was developed respecting [CzechResearchGroup]. Although it does not give $\mathbf{A}_k$ directly, it fully satisfies our needs. The inputs of the procedure are $\mathbf{F}$, $\mathbf{G}$, $\mathbf{x}_0$. The procedure yields on its output the $(n_{RT} \times n_{RT})$-dimensional matrix $\mathbf{A}$ and the $(n \times n_{RT})$-dimensional matrix $\mathbf{X}_{reach}$. $\mathbf{A} = \{a_{i,j}\}_{n_{RT} \times n_{RT}}$ is the *quasi-functional* adjacency matrix. Its elements $a_{i,j}$, $i = 1, ..., n_{RT}$, $j = 1, ..., n_{RT}$ are either equal to 0 (when the RT nodes i, j are not connected) or to a positive integer denoting the ordinal number of the transition through which the oriented arc passes from the RT node i to the RT node j. The integer n_{RT} denotes the number of feasible states (i.e. $\mathbf{x}_0$ and all of the states reachable from $\mathbf{x}_0$). The feasible states are given as the columns of the matrix $\mathbf{X}_{reach}$. It is necessary to say that the RT and its corresponding RG have the same matrix $\mathbf{A}$. The procedure is the following:

```
Xreach=x0;
A=[0];
[n,m]=size(F);
B=Gt-F;
i=0;
while i<size(Xreach,2)
        i=i+1;
        for k=1:m
            x(k)=all(Xreach(:,i) >= F(:,k));
        end
        findx=find(x);
        for k=1:size(findx,2)
            bb = Xreach(:,i)+B(:,findx(k));
            matrix=[];
            for j=1:size(Xreach,2)
                matrix=[matrix,bb];
            end
            v=all(matrix == Xreach);
            j=find(v);
            if any(v)
                A(i,j)= findx(k);
            else
                Xreach=[Xreach,bb];
                A(size(A,1)+1,size(A,2)+1)=0;
                A(i,size(A,2))=findx(k);
            end
        end
Xreach;
A;
end
```

The following functions are used in the MATLAB procedure: *any(v)* returning 1 if any of the elements of the vector $\mathbf{v}$ are non-zero, *all(v)* returning 1 if all of the elements of the vector $\mathbf{v}$ are non-zero, and j = *find(v)* returning the indices of the vector $\mathbf{v}$ that are non-zero. In addition to this the function *size* is utilized. When their operand is a matrix $\mathbf{M}$ with the dimensinality $(n \times m)$ the command d = *size(M)* returns the two-element row vector $\mathbf{d} = (n, m)$ containing the number of rows and columns of the matrix $\mathbf{M}$. When the second operand - the scalar *dim* - is added, then the command x = *size(M,dim)* returns the length x of the dimension specified by the scalar *dim*. When *dim = 1* it returns the number of rows $(x = n)$, when *dim = 2* it returns the number of columns $(x = m)$ of the matrix $\mathbf{M}$.

5.2 The modification of the BDG-based approach

In contrast to the DG-based approach which is suitable for PN with general structure the above described BDG-based approach is suitable only for SM. It handles matrices $\mathbf{F}$, $\mathbf{G}$ as well as the PN transitions. However, after the trasformation PN with general structure to the DG-based model we have not such matrices. Therefore, after enumerating the *quasi-functional* adjacency matrix $\mathbf{A}$ we have to utilize this matrix for finding the matrices $\mathbf{F}_{RG}$ and $\mathbf{G}_{RG}$. Because of the fact that the PN transitions indices can occur more then once among the elements of the *quasi-functional* matrix $\mathbf{A}$ some confusions could occur. To avoid these it is necessary to rename the original PN transitions in order to obtain fictive transitions that occure only once. The number of them is T_r representing the global number of the nonzero elements of $\mathbf{A}$. The renaming is performed raw-by-raw so that the nonzero elements are replaced by integers - ordinal numbers starting from 1 and finishing at T_r. Thus, the auxiliary matrix $\mathbf{A}_{T_r} = \{a_{T_r}(i,j)\}_{n_{RT} \times n_{RT}}$ is obtained from $\mathbf{A}$. The disassambling of the matrix $\mathbf{A}_{T_r}$ into the incidence matrices is given as follows. For $i = 1, ..., n_{RT}$, $j = 1, ..., n_{RT}$ if $a(i,j) \neq 0$ and $a_{T_r}(i,j) \neq 0$ we set $T_{rTt}(a(i,j), a_{T_r}(i,j)) = 1$, $F_{RG}(i, a_{T_r}(i,j)) = 1$, $G_{RG}(a_{T_r}(i,j), j) = 1$ else we set the elements of the matrices to be equal to 0. Here, $\mathbf{T}_{rTt} = \{T_{rTt}(i,j)\}_{m \times T_r}$ is the transformation matrix between the original set of transitions and the fictive one. Hence, $\mathbf{U} = \mathbf{T}_{rT_t}.\mathbf{U}^*$ where the matrix $\mathbf{U}^*$

yields the control strategies (20) or (46) computed by means of the set of the fictive transitions and $\mathbf{F}_{RG}$ and $\mathbf{G}_{RG}$.

5.3 The example 3

Consider the DEDS of the software kind given in [Pastor et al.]. It is concerning the coding and it has the PN-based model with general structure (even with multiplicity of the oriented arcs). The PN-based model is presented in Fig. 7.a. We can see that $n = 4$, $m = 3$, $\mathbf{x}_0 = (2, 0, 1, 0)^T$ and the structural matrices are the following

$$\mathbf{F} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix} ; \mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Utilizing the above MATLAB procedure we obtain parameters of the corresponding RT placed in Fig. 7.b. The RT structure is described by the *quasi-functional* adjacency matrix $\mathbf{A}$ while the space of the feasible states are expressed by the columns of the matrix $\mathbf{X}_{reach}$. They represent the feasible state vectors $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_6$.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix} ; \mathbf{X}_{reach} = \begin{pmatrix} 2 & 0 & 3 & 1 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 2 & 2 \\ 1 & 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 2 & 2 & 4 & 4 & 6 \end{pmatrix}$$

Afterwards, the auxiliary matrix $\mathbf{A}_{T_r}$ is constructed by means of $\mathbf{A}$. It is the *quasi-functional* adjacency matrix concerning the transitions of the fictive PN. Such a PN is a fictive SM corresponding to the original PN. Namely, the elements $\mathbf{A}_{T_r}$ represent the indices of the fictive transitions. Thus, the transformation matrix $\mathbf{T}_{rTt}$ between both the transitions of the original PN and the transitions of the fictive SM can be enumerated by means of the matrices $\mathbf{A}$ and $\mathbf{A}_{T_r}$ (compare both of the matrices). Then, the matrices $\mathbf{F}_{RG}$, $\mathbf{G}_{RG}$ are computed by means of disassembling $\mathbf{A}_{T_r}$. All of the matrices in question are as follows

$$\mathbf{A}_{T_r} = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 11 & 0 & 0 \end{pmatrix} ; \mathbf{T}_{rTt} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

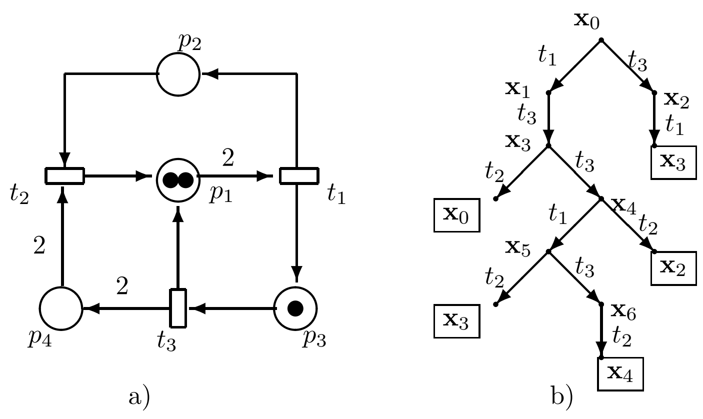


Figure 7: a) The PN-based representation of DEDS; b) The corresponding RT

$$\mathbf{F}_{RG} = \begin{pmatrix} 1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix} ; \; \mathbf{G}^T_{RG} = \begin{pmatrix} 0&0&0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0 \\ 0&0&1&1&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

In case when $\mathbf{x}_t = \mathbf{x}_6 = (1, 2, 0, 6)^T$ is chosen to be the terminal state the DG-based method of the DEDS control synthesis yields the solution (47) of the control synthesis problem - i.e. finding the state trajectories. Its graphical expression is on the left in Fig. 8. When the BDG-based method is used the solution of the control synthesis problem (as to finding the state trajectories) is the same like that in the DG-based method. The solution of the control synthesis problem as to the fictive control trajectories is given by (48). It was found on base of the renamed (fictive) transitions. The actual control trajectories are given after the backward process (i.e. after the return to the original set of transition) by (49) and they are graphically expressed on the right in Fig. 8.

$$\mathbf{X}_1 = \begin{pmatrix} 1&0&0&2&0&0 \\ 0&1&0&0&2&0 \\ 0&1&0&0&4&0 \\ 0&0&2&0&0&8 \\ 0&0&0&2&0&0 \\ 0&0&0&0&2&0 \\ 0&0&0&0&0&2 \end{pmatrix} ; \; \mathbf{X}_2 = \begin{pmatrix} 2&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 3&0&0&1&0&0 \\ 0&2&0&0&1&0 \\ 0&0&1&0&0&1 \end{pmatrix} ; \; \mathbf{X} = \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{pmatrix} \qquad (47)$$

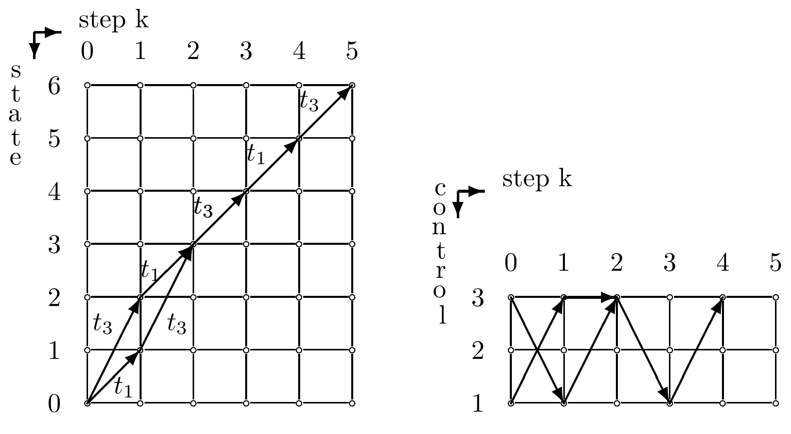


Figure 8: The solution of the state trajectories (on the left) and the control trajectories (on the right)

$$\mathbf{U}_1^* = \begin{pmatrix} 1\,0\,0\,2\,0 \\ 1\,0\,0\,2\,0 \\ 0\,1\,0\,0\,2 \\ 0\,1\,0\,0\,4 \\ 0\,0\,2\,0\,0 \\ 0\,0\,2\,0\,0 \\ 0\,0\,0\,2\,0 \\ 0\,0\,0\,2\,0 \\ 0\,0\,0\,0\,2 \\ 0\,0\,0\,0\,2 \\ 0\,0\,0\,0\,0 \end{pmatrix} ; \mathbf{U}_2^* = \begin{pmatrix} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0 \\ 2\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0 \end{pmatrix} ; \mathbf{U}^* = \begin{pmatrix} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0 \end{pmatrix} \quad (48)$$

$$\mathbf{U} = \mathbf{T}_{rTt}.\mathbf{U}^* = \begin{pmatrix} 1\,1\,0\,1\,0 \\ 0\,0\,0\,0\,0 \\ 1\,1\,1\,0\,1 \end{pmatrix} \quad (49)$$

6 The control synthesis of DEDS with the infinite state space

The PN-based models with general structure can conduce to the RT and RG with cycles. The number of their nodes has the tendency to be infinite. In such a case the above described procedure is not able to compute the matrices $\mathbf{A}$ and $\mathbf{X}_{reach}$. Therefore, the RT is replaced [Peterson 1981] by the so called coverability graph (CG). Here, the symbol ω is introduced in order to express infinity elements of the state vectors creating the CG nodes. Namely, the RT with cycles is degenerated and it obstructs utilizing the above mentioned approach to the control synthesis, the corresponding RG has a specific shape, and the adjacency matrix cannot be expressed in the desirable unambiguous form. The situation can be illustrated by means of the following simple example. Very useful MATLAB program by Iordache for computing RG in general (i.e. including CG) is presented in [Iordache and Antsaklis].

6.1 The example 4

Consider the very simple example of the FMS, namely the robotic cell consisting of two conveyors $C1$, $C2$, NC-machine tool M with buffer B (having the input section $B1$ and the output section $B2$), and the robot R. R picks up a part from $C1$ and places it into B. After machining the part by M (i.e. after the completion of the final part) R takes the final part from the buffer and places it on $C2$. The interpretation of the places is as follows: p_1 = waiting input parts, p_2 = waiting output parts, $p_3 = R$ is available, $p_4 = M$ is available, p_5 = contents of B. Taking parts by R from $C1$ is modelled by the transition t_1, the placement of the finished parts by R on $C2$ is modelled by t_3, and the machining the part by M is modelled by t_2. The PN-based model is given in Fig. 9.a. It is clear that in this example the FMS can be replaced by a software system too. In case when the infinite capacities of the PN places are used the corresponding RT and RG are given in Fig. 9.b and Fig. 9.c. The framed states in RT represent the repeating

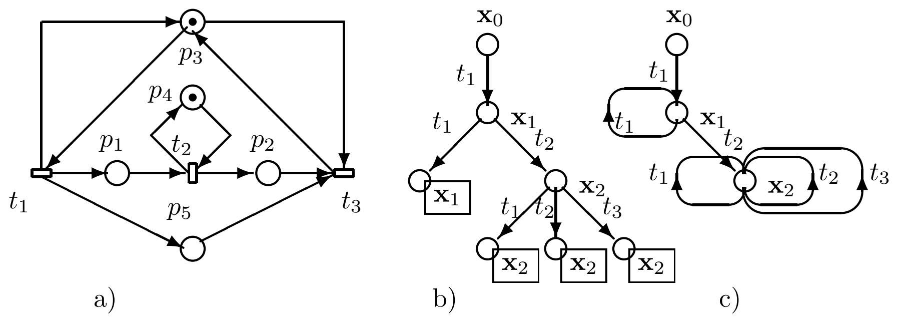


Figure 9: a) The PN-based model of the flexible manufacturing system with the infinite state space; b) The corresponding RT; c) The corresponding RG

states. However, the cycles engender in the RG in this case. These cycles generate the states containing places with so called ω elements, $\omega = 0, 1, ..., \infty$. In such a way the state space is going to be infinite. The PN model parameters, the initial state $\mathbf{x}_0$ and the space of the reachable states $\mathbf{X}_{reach}$ are the following

$$\mathbf{F} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} ; \ \mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} ; \ \mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} ; \ \mathbf{X}_{reach} = \begin{pmatrix} 0 & \omega & \omega \\ 0 & 0 & \omega \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & \omega & \omega \end{pmatrix}$$

It is clear from Fig. 9 that the *quasi-functional* adjacency matrix $\mathbf{A}$ and the functional matrix Δ_k of the DG-based model are as follows

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & (1,2,3) \end{pmatrix} ; \ \Delta_k = \begin{pmatrix} 0 & 0 & 0 \\ \gamma_{t_1} & \gamma_{t_1} & 0 \\ 0 & \gamma_{t_2} & (\gamma_{t_1}, \gamma_{t_2}, \gamma_{t_3}) \end{pmatrix}$$

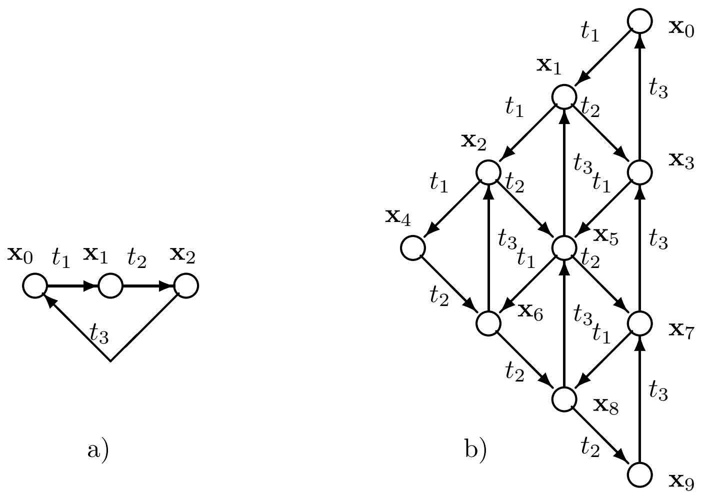
As we can see the elements $a_{3,3}$ and $\delta_{3,3}^{(k)}$ are ambiguous. In addition to this the numerical computing with the element ω occuring among the elements of the reachable state vectors is also ambiguous. Consequently, we cannot utilize the above approach to the control synthesis in such a case. Therefore, it is necessary to use a modified approach.

6.2 The modified approach considering the capacities of PN places

When the capacities of the PN places are infinite the corresponding RT and RG can be *degenerated* as it was presented in the previous example in Fig. 9. When the capacities are strongly prescribed as finite the situation is changed. Therefore, it is necessary to define the corresponding capacities of the PN places. The different capacities yield different RT and RG. Fortunately, the real technical devices usually have natural terminations. In general, there exist different kinds of restrictions. They depend on the kind of technological equipment and many times they are flexible. Consider e.g. the capacity of the buffer of the NC-machine tool in the Example 2 to be only 1, i.e. $c_{p_5} = 1$. It means that either the input part or the finished one can be placed into the buffer only. In such a case the RG given in Fig. 10.a is obtained. The *quasi-functional* adjacency matrix and the space of the reachable states are the following

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 3 & 0 & 0 \end{pmatrix} ; \ \mathbf{X}_{reach} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

In this case there exists only one possible trajectory $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2\}$ which is simultaneously feasible. It can be realized by means of the sequence of enabled transitions $\{t_1, t_2, t_3\}$. However, when the different capacities $c_{p_i} = 3, \ i = 1, \ 2, \ 5$



$\mathbf{x}_0$ t_1 $\mathbf{x}_1$ t_2 $\mathbf{x}_2$
t_3
a)
b)
$\mathbf{x}_0$ $\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{x}_5$ $\mathbf{x}_6$ $\mathbf{x}_7$ $\mathbf{x}_8$ $\mathbf{x}_9$
t_1 t_2 t_3

Figure 10: The corresponding RG of the inquired FMS. a) when the capacity $c_{p_5} = 1$; b) when the capacities $c_{p_i} = 3$, $i = 1, ..., 5$

are used, the different RG is obtained. It is presented in Fig. 10.b. Here, the results $\mathbf{A}$, $\mathbf{X}_{reach}$, and the $\mathbf{M}$ (where the trajectories transforming the system from $\mathbf{x}_0 = (0, 0, 1, 1, 0)^T$ to the terminal state $\mathbf{x}_9 = (0, 3, 1, 1, 3)^T$ are comprehended) are as follows while the graphical expression of the solution is given in Fig. 11.

$$\mathbf{X}_{reach} = \begin{pmatrix} 0 & 1 & 2 & 0 & 3 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 2 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 & 3 & 2 & 3 & 2 & 3 & 3 \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \end{pmatrix} ; \ \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

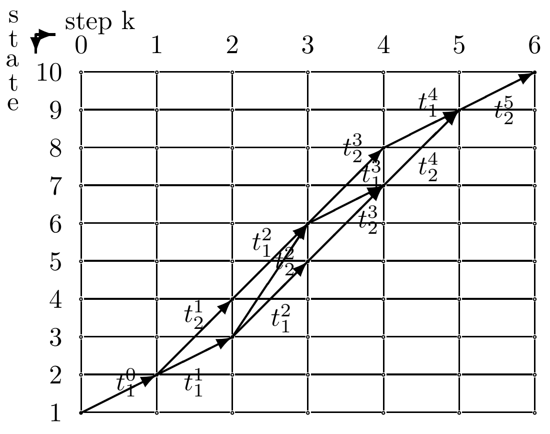


Figure 11: The resulting feasible trajectories from the initial state $\mathbf{x}_0$ to the terminal state $\mathbf{x}_9$ when the capacities $c_{p_i} = 3$, $i = 1$, ..., 5

7 The graphical tool GraSim

The graphical tool GraSim was developed in order to automate the control synthesis. By means of the mouse and icons the RG is drawn. After this the control synthesis can be done. Both the straight-lined system development and the backtracking one can be find autonomously for any finite number of steps (i.e. for any length of trajectories). However, the feasible trajectories (i.e. the intersection of the SLRT and BTRT) can also be found directly (without separate finding the autonomous developments). The default length is the shortest one. The tool yields (in the graphical form) all of the system feasible trajectories from a given initial state to a prescribed terminal one. It allows to analyze the trajectories one by one. It is very important. Namely, e.g. DEDS handled in the Example 4 has five feasible trajectories shown in Fig. 11. They are given by the following sequences of states $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9\}$, $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9\}$, $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$, $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9\}$, $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}$.

8 Conclusions

The approach to the modelling and control synthesis of DEDS was presented in this paper. Its applicability to the special communication systems was demonstrated on several examples. The approach is suitable especially for DEDS described by PN with the finite state space. However, the possibility of partial dealing with the DEDS described by PN with infinite state space was pointed out too. In order to automate the control synthesis process the graphical tool GraSim was developed. Here it was characterized and concisely described.

Acknowledgements

The author thanks VEGA (Slovak Grant Agency for Science) for the partial support under grant # 2/3130/24. Thanks go also to author's colleagues for their thoroughful reading the paper and for their valuable remarks and comments.

References

[Čapkovič 2003] Čapkovič, F.: "The Generalised Method for Solving Problems of DEDS Control Synthesis"; Proc. Developments in Applied Artificial Intelligence. Lect. Notes Comp. Sci. 2718, Springer, Berlin (Jul 2003), 702-711.

[Čapkovič and Čapkovič 2003] Čapkovič, F., Čapkovič, P.: "Petri Net-based Automated Control Synthesis for a Class of DEDS"; Proc. 2003 IEEE Conf. on Emerging Technologies and Factory Automation. IEEE Press, Piscataway, NJ, USA (Sep 2003), 297-304.

[Čapkovič 2002] Čapkovič, F.: "Control Synthesis of a Class of DEDS"; Kybernetes. The Int. Journ. of Sys. and Cybern., 31, 9/10 (Dec 2002) 1274-1281.

[Čapkovič 2002a] Čapkovič, F.: "A Solution of DEDS Control Synthesis Problems"; Systems Analysis Modelling Simulation, 42, 3 (March 2002) 405-414.

[Čapkovič and Čapkovič 2001] Čapkovič, F., Čapkovič, P.: "Intelligent Control Synthesis of Manufacturing Systems"; Proc. Engineering of Intelligent Systems. Lect. Notes Comp. Sci. 2070, Springer, Berlin (Jun 2001), 767-776.

[Čapkovič 2000] Čapkovič, F.: "Intelligent Control of Discrete Event Dynamic Systems"; Proc. IEEE Int. Symp. on Intelligent Control. IEEE Press, Patras, Greece (Jul 2000) 109-114.

[Čapkovič 2000a] Čapkovič, F.: "Modelling and Control of Discrete Event Dynamic Systems"; BRICS Report Series RS-00-26. Dept. Comp. Sci., Aarhus Univ., Denmark (Oct 2000) `http://www.brics.dk/RS/00/Ref/BRICS-RS-00-Ref/BRICS-RS-00-Ref.html`

[Čapkovič 2000b] Čapkovič, F.: "Solution of DEDS Control Synthesis Problems"; Proc. IFAC Conf. on Control System Design. Pergamon, Elsevier Science, Oxford, UK (Jun 2000) 343-348.

[Čapkovič 1999] Čapkovič, F.: "Automated Solving of DEDS Control Problems"; Proc. Multiple Approaches to Intelligent Systems. Lect. Notes Comp. Sci. 1611, Springer, Berlin (May/Jun 1999), 735-746.

[Čapkovič 1998] Čapkovič, F.: "Knowledge-Based Control Synthesis of Discrete Event Dynamic Systems"; Advances in Manugacturing. Decision, Control and Information Technology, Chapter 19, Springer, London (1998) 195-206.

[CzechResearchGroup] "http://dce.felk.cvut.cz/cak/Research/index_Research.htm"

[Diestel 1997] Diestel, R.: "Graph Theory"; Springer, New York (1997).

[Iordache and Antsaklis] Iordache, M.V., Antsaklis, P.J.: "Software tools for the supervisory control of Petri nets based on place invariants"; Univ. of Notre Dame, Notre Dame, USA, technical report ISIS-2002-003, 2002.

[Pastor et al.] Pastor, E., Cortadella, J., Pena, M.A.: "Structural Methods to Improve the Symbolic Analysis of Petri Nets"; Application and Theory of Petri Nets 1999. Lect. Notes Comp. Sci. 1639, Springer, Berlin (Jun 1999) 26-45.

[Peterson 1981] Peterson, J.L.: "Petri Net Theory and Modeling the Systems"; Prentice Hall, New York (1981).

[Tzafestas and Čapkovič 1997] Tzafestas, S.G., Čapkovič, F.: "Petri Net-Based Approach to Synthesis of Intelligent Control for DEDS"; Computer Assisted Management and Control of Manufacturing Systems, Chapter 12, Springer, Berlin (1997) 325-351.