Appl. Math. and Comp. Sci., 1992, vol.2, No.1, 65-86

PETRI NET APPROACH TO THE SYNTHESIS OF SELF-REPROGRAMMING CONTROL SOFTWARE

ZBIGNIEW BANASZAK*, ROBERT WÓJCIK*, KELWYN A. D'SOUZA**

A new approach to automated design of adaptive concurrent control software of discrete event systems is presented. An algorithm transforming a given specification of the pipeline-like flowing concurrent processes into a Petri net model of admissible control flows of competing processes is introduced. Models generated guarantee the time-independent and deadlock-free execution of processes. Based on the assumption that some of the system shared resources may be substituted by others, e.g. in case of their damage, a fault-tolerant recovery procedure aimed at the reprogramming of Petri net-based control software is provided. This procedure guarantees the fault-tolerant operation of real-time controlled systems. The formal description of modelling procedures is described in details.

1. Introduction

For many years, one of the research topics of system safety and reliability has been the development of sufficiently efficient representation suitable for programming automation of adaptive, logical industrial controllers. In order to reduce the costs born for generation, debugging, and maintenance of industrial controllers, many Petri net-based methodologies have been developed, such as Petri net tools for the modelling and performance evaluation (Mamath and Vishwanadham, 1986; D'Souza, 1991; Chu Zhou *et. al.*, 1990), and for the specification and analysis of discrete state systems (Gentina *et. al.*, 1988; Shibata *et. al.*, 1988; Willson and Krogh, 1990) as well as for the rapid prototyping (Feldbrugge, 1988; Duggan and Browne, 1988) and automated development of the control software (Banaszak, 1988; Pathak, 1988). In all of the above mentioned approaches to the programming automation, the stages of the controlled processes specification, modelling, analysis and implementation (control code generation) may be easily distinguished.

This paper presents a methodology aimed at self-programming of discrete controllers, employed in the course of adaptive control of concurrently processed material or data flows. The approach proposed is based on the concept of automatic generation of Petri net models of admissible, i.e. deadlock-free, control flows (Banaszak and Krogh, 1990; Roszkowska and Banaszak, 1988; Banaszak and

* Institute of Engineering Cybernetics, Wrocław Technical University, 50-372 Wrocław, Poland
** Dept. of Industrial and Management Systems Engineering, University of South Florida, Tempa, Florida 33620, USA

Krogh, 1991). In turn the transformation of the processes specification (providing the description of the desired behaviour of the controlled system) into an equivalent Petri net model plays the main role in the concept applied.
Its characteristic feature is that it allows one to omit the time–consuming stage of the analysis of net models.

An algorithm performing the transformation of the concurrent processes specification is different from that in (Banaszak and Mowafak, 1988; Banaszak, 1991). It applies a Petri net model representation of distinguished events and conditions, and employs a new, based on the critical–loop concept procedure for deadlock handling.

The main contribution of this paper is an algorithm transforming a given specification of the pipeline–like flowing concurrent processes into a standard form Petri net model of admissible control flows. A class of considered processes specification takes into account only the sequential processes (i.e. processes without branching, merging, or alternative operation sequences). Each operation requires a subset of resources available in the system. We call it multiple resource–per–operation processes. It is assumed that some of the system shared resources may be substituted by other ones, e.g. in case of their damage. Based on the above mentioned assumptions, a fault–recovery procedure aimed at the reprogramming of the Petri net–based control algorithms is provided. The technique of automatic design of Petri net models encompassing admissible material flows, and a relevant fault–recovery procedure are combined in a method aimed at automatic design of adaptive real–time logical controllers.

The paper is organized as follows. In Section 2, the tasks of selection of processes specification and formalism employed as well as problem statement are presented. Then, in Section 3, we show how the Petri net models of admissible control programs can be automatically constructed for a preassumed specification of processes controlled. In Section 4, we show how the results of Section 3 can be utilized in a method aimed at automatic design of adaptive and real–time logical controllers. The advantages and limitations of the obtained results are disscussed in Section 5 along with a scope for future research.

2. Modelling Automation

During the last decade, since the computer integrated manufacturing systems have occurred, many demands of the flexibility and reliability have increased. The effective use of such systems, able of treating a wide variety of products or jobs, depends on their hardware and software components responsibility for discrete control.

In order to increase the reliability of manufacturing systems operation the applied discrete controllers, ranging from simple programmable logic controllers to general purpose computers, should exhibit adaptive control features such as, for instance, the fault tolerantness. Taking this into account, we present a methodology that addresses the specification of the required behaviour of the controlled system, the automated generation of controller programs, and the fault recovery procedure employed in the course of an adaptive control of concurrently flowing processes.

The modelling and control problems of Flexible Manufacturing Systems (FMSs) require the designer to consider the concurrent and asynchronous interactions of events occuring in the course of material flows processed on a set of programmed machine tools as well as in the course of data and information processing observed on the computer and LAN levels (Banaszak, 1991). Such systems, by permitting materials or data processing tasks to be performed concurrently, make possible more efficient use of systems resources. This way leads to increasing FMS efficiency but implies, however, a problem being a common one for concurrent system programmer. The problem expresses the following question: Do the programs generating the required system behavior satisfy certain requirements?

Among the requirements considered, the most important ones is a guarantee of the control software correctness (including the protection of the shared resources, absence of deadlocks and processes starvation) as well as the real–time execution of control algorithms enabling the system to reach an optimum of the preassumed system performance.

Concurrency has been a fundamental issue in distributed computing systems and operating systems of large computers for many years. Only recently, however, has the concept of concurrent control been introduced into a domain of FMSs design and operation. The most notable difference between material or control flows in an FMS and data flows in a computer concerns the requirements assumed for resources allocation. For instance, the production routes in an FMS indicate an order in which preassumed entities of required resources must be allocated and deallocated to the pipeline–like performed jobs. In turn, for computer systems it is usually assumed that only bounds on the total number of entities of resources required by each process are known.

2.1. Processes Specification

The requirement concerning the correctness of the FMS control software will play a primary role in our approach applied to automation of adaptive concurrent control programming. We suppose that it is possible to generate the dynamic, automaton–like models of admissible (i.e. correct in the sense mentioned ealier) control flows automatically. So, our problem can be stated as follows: How to utilize the possibility of an automatic synthesis of the net models of admissible control flows in order to develop a method aimed at the solution of the system adaptation tasks. As an illustration the tasks associated with adaptation to the breakdowns of system's resources and the production objective changes can be considered.

In order to present the problem more precisely let us introduce the following assumptions imposed on the systems considered:

- an FMS consists of a finite number of resources, such as programmable machine tools, industrial robots, and storage buffers, which may function concurrently,

- an FMS is functionally and structurally redundant, i.e. some of its resources (or their functions) are replacable for others,
- each resource may have multiple units,
- in an FMS a finite set of concurrent asynchronously executed processes is performed,
- each process realizes a pipeline–like flow of workpieces (jobs) along a given production route,
- some of the system resources may be shared among different processes,
- pipeline–like flowing jobs, which arrive into the system randomly, cannot be divided and no more than one job may utilize single instance of a given resource type at a time.

For further considerations we assume also that the specification of system behaviour will be limited to the set of production routes and to the function determining a number of units of each resource type as well as the resource requirement function which determines a number of units of each resource type required by an operation at a given stage of process performance, required in the course of processes execution. So, as the process specification the following form is applied

$$PS = (\{PR_i | i = \overline{1,v}\}, \varphi, \psi) \tag{1}$$

where $PR_i = (O_{i,1}, \{R_j | J_1\}), (O_{i,2}, \{R_j | j \in J_2\}), ..., (O_{i,u}, \{R_j | j \in J_u\}), ..., (O_{i,r}, \{R_j | j \in J_r\})$ is the i–th production route encompassing the technological order of technological (i.e. machining, transportation, storage etc.) operations performed on the system resources through which the i–th type of product is completed,

$(O_{i,u}, \{R_j | j \in J_u\})$ means that in order to perform the u–th stage of the i–th process, i.e. the process executed along the i–th production route, a subset $\{R_j | j \in J_u\}$ of the all system resource types $R = \{R_j | j \in J\}$ is required, and a number of each resource type requested on the u–th stage is determined by the following function, where J is the number of the all resource types,

$\psi : U \times V \times J \to N$ is the resource requirements function determining the number of the j–th resource type units required on the u–th stage of the i–th process performance and defined for $u \in U$, $i \in V$ and $j \in J$ such that $U \subset N$, $U = \max_{i=\overline{1,v}}\{|PR_i|\}$, $V \subset N$, $V = \{i | i = \overline{1,v}\}$, where $|PR_i|$ is a length of PR_i, and $\mathrm{crd}^i S \overset{\text{def}}{\Longleftrightarrow} s_i$ for $S = (s_1, s_2, ..., s_i, ..., s_r)$,

$\varphi : R \to N$ is the capacity function which determines a maximal number of units of each resource type declared in the considered process specification, $N = \{1, 2, 3, ...\}$ is a set of natural numbers.

Moreover, it is assumed that the same number of units of a given resource type is released that was previously requested in order to execute an operation.

2.2. Modelling Formalism

Treating a given process specification PS as an input data in a task of automated generation of controller programs we employ a class of the Petri nets called Place/Transition system (Peterson, 1985; Murata, 1989) as a formalism suitable for representation of material flow models.

A Place/Transition system (P/T–net for short) is a sixtuple

$$PN = (P, T, E, W, K, M_0) \tag{2}$$

where P and T are finite sets of places and transitions, respectively, such that

$$P \cup T \neq \emptyset \text{ and } P \cap T = \emptyset,$$

$E \subset (P \times T) \cup (T \times P)$ is a flow relation such that, $\text{dom}(E) \cup \text{cod}(E) = P \cup T$,
$W : E \longrightarrow N$ is an arc weight function,
$K : P \longrightarrow N$ is a place capacity function,
$M_0 : P \longrightarrow N_0$ is an initial marking such that the following condition holds,
$(\forall p \in P)(M_0(p) \leq K(p))$, where $N_0 = N \cup \{0\}$.

The marking of a PN indicates the number of tokens in each place, which is the current state of the system.

The enableness conditions are determined as follows

$$\begin{array}{ll} \text{(i)} & (\forall p \in {}^{\bullet}t)\ (M(p) \geq W(p,t)), \\ \text{(ii)} & (\forall p \in t^{\bullet} \cup {}^{\bullet}t \cap t^{\bullet})\ (M(p) \leq K(p) - W(t,p) + W(p,t)), \end{array} \tag{3}$$

where ${}^{\bullet}t = \{p | (p,t) \in E\}$, $t^{\bullet} = \{p | (t,p) \in E\}$.

An enabled transition $t \in T$ of PN can fire, thus removing $W(p,t)$ tokens from each input place $p \in {}^{\bullet}t$ and placing $W(t,p) - W(p,t)$ tokens in each output place $p \in t^{\bullet}$.

Because of space limitations, the formal definitions of the next–state function and the reachability set as well as structural and functional properties of Petri nets are omitted.

P/T–net formalism serves for representation of models encompassing all possible material flows in the system controlled. Its extention to the predicate P/T–nets we assume to be suitable for representation of the models encompassing only the admissible (deadlock–free) realizations of the modelled material flows, and obtained from the appriopriate P/T–net specifications.

A predicate P/T–net, PN_{Pr} is determined as follows

$$PN_{Pr} = (P, T, E, W, K, Pr, M_0), \tag{4}$$

where $Pr = \{F(M, \mathcal{L}) | M \in R(PN, M_0) \ \& \ \mathcal{L} \in \mathbb{L}\}$ is a set of predicates which extend the enableness conditions (3) for the following one

$$F(M, \mathcal{L}) \quad \text{is true}, \tag{5}$$

and where $R(PN, M_0)$ is the reachability set of markings obtained from M_0 in $PN = (P, T, E, W, K, M_0)$, and $\mathbb{L}$ is a set of subgraphs of resource precedence graph $G = (R, <)$ corresponding to a given process specification (1).

2.3. Problem Statement

Thus, we assume that there exists the following transformation

$$A : \{PS\} \rightarrow \{PN_{Pr}\}, \tag{6}$$

converting given process specification PS into the relevant predicate P/T–net model PN_{Pr} which encompasses the admissible material flows performed in the controlled system. The controller implementation of PN_{Pr} model serves then for a control program supervising the time–independent flows of concurrent processes. In other words, the control computer maintains a PN_{Pr} model of the controlled system, using the marking of the net to determine required control actions.

Taking this into account a problem we are facing now is: How to apply the transformation algorithm (6) into a procedure aimed at designing of adaptive industrial controllers employed for the real–time supervision of concurrently flowing processes? The flow chart of such procedure which emphasizes the fault recovery aspect of the adaptive control is shown in Figure 1.

Besides the already discussed stage of the predicate P/T–net models synthesis, the considered procedure employs the stages corresponding to the following tasks:

- selection of efficient components,
- selection of the system resources which should be released from a system,
- replacement of a current control program by a newly generated one.

The first one from the above mentioned problems is stated as follows. For a given system's component identified as a broken down one, from a set of resources capable to replace that component, select a resource which extremizes a given function of system performance costs. Such a goal function can be determined for instance either on the basis of a set of *a priori* given parameters determining the cost of particular components replacement or on the basis of a set of standard criterions evaluating system performance such as throughput, index of machines utilization, makespan, lateness and so on.

The second problem corresponds to the evaluation of so called inter–state distance measure (e.g. determined by a number of resources units required for releasing from the system) applied for the marking encompassing the current system's state (i.e. allocation of system resources) and the marking belonging to a reachability set of a PN_{Pr} model of a new control (fault recovery) algorithm. The performance index constructed on the basis of the concept of inter–state distance measure may assume that costs corresponding to the given distances are *a priori* given (e.g. resulting directly from the costs of particular resources releasing or from the costs of the particular products (jobs) manufacturing).

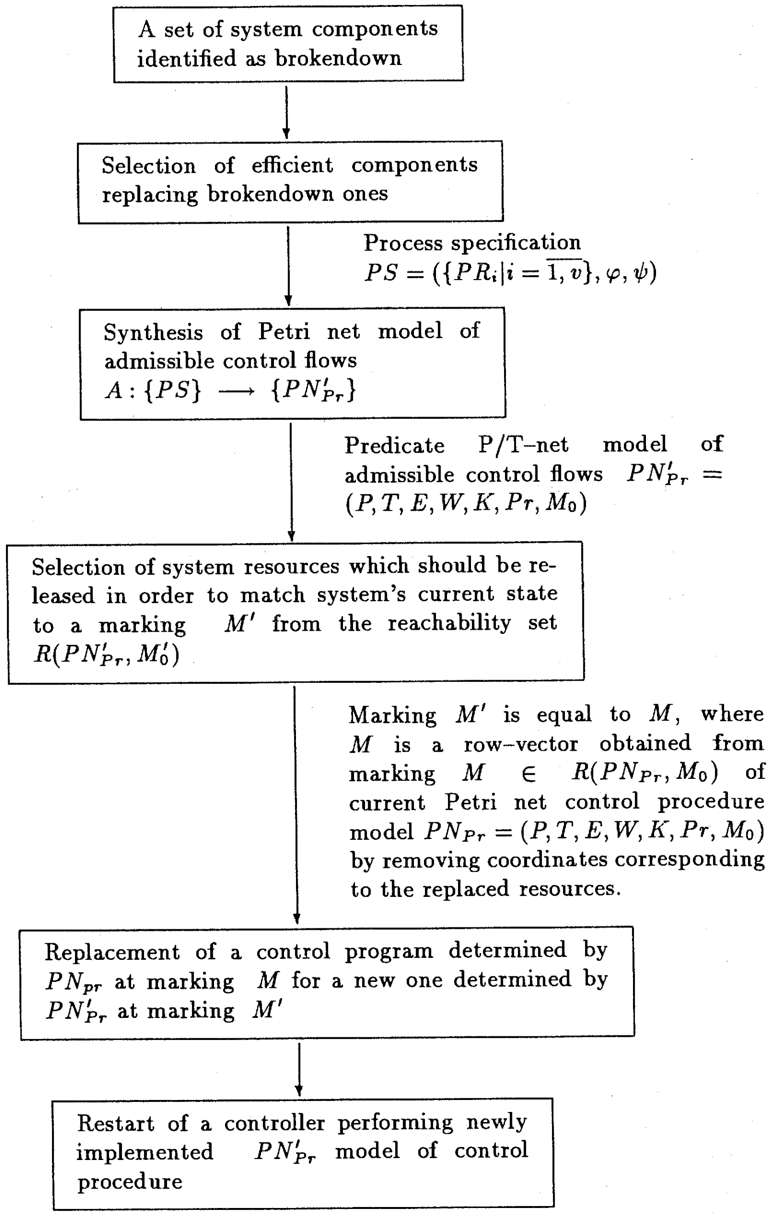
A set of system components identified as brokendown
Selection of efficient components replacing brokendown ones
Process specification $PS = (\{PR_i \mid i = \overline{1,v}\}, \varphi, \psi)$
Synthesis of Petri net model of admissible control flows $A : \{PS\} \longrightarrow \{PN'_{Pr}\}$
Predicate P/T–net model of admissible control flows $PN'_{Pr} = (P, T, E, W, K, Pr, M_0)$
Selection of system resources which should be released in order to match system's current state to a marking M' from the reachability set $R(PN'_{Pr}, M'_0)$
Marking M' is equal to M, where M is a row–vector obtained from marking $M \in R(PN_{Pr}, M_0)$ of current Petri net control procedure model $PN_{Pr} = (P, T, E, W, K, Pr, M_0)$ by removing coordinates corresponding to the replaced resources.
Replacement of a control program determined by PN_{pr} at marking M for a new one determined by PN'_{Pr} at marking M'
Restart of a controller performing newly implemented PN'_{Pr} model of control procedure

Fig. 1. Flow chart of fault recovery procedure.

The last of the above listed problems belongs rather to the software engineering ones. It corresponds to the problem of task oriented software design, especially aimed for the computer implementation of the predicate P/T–net models of feasible control programs (i.e. transformation of the given PN_{Pr} model into executable control programs) as well as for the specific management functions responsible, for instance, for the control programs stopping and restarting, replacement and monitoring, and so on.

Because the first and last tasks are implementation dependent ones we will focus our attention on the remaining, more general problems regarding automation of the PN_{Pr} models synthesis and selection of intendent system's resources releasing.

Methods aimed at solution of the selected subproblems will be discussed in the next sections.

3. Programming Automation

In this section a problem of development of an algorithm allowing to transform a given process specification (1) into a corresponding predicate P/T–net (4) is considered. According to the problem statement introduced in Section 2 the task considered can be decomposed into the following two tasks

- automatic design of PN_{Pr} model encompassing all possible material flows according to the given production processes specification, i.e. development of an algorithm able to perform the following transformation
$$A1 : \{PS\} \rightarrow \{PN\} \tag{7}$$
- modification of the obtained model in order to transform it to a new form encompassing only the admissible (deadlock–free) realizations of modelled material flows, i.e. development of an algorithm able to perform the following transformation
$$A2 : \{PN\} \rightarrow \{PN\} \tag{8}$$

In order to present an algorithm aimed at solution of the first problem, let us introduce the following algebraic representation of PN

$$PN = (C, K, M_0) \tag{9}$$

where C is the incidence matrix size of $m \times n$, $m = ||T||$, $n = ||P||$, $||T||$ and $||P||$ denotes the cardinality of sets T and P, respectively, K and M_0 denotes the row vectors, size of $1 \times n$, corresponding to the capacity function and initial marking, respectively.

Of course, such assumption restricts our future considerations to P/T–net without loops, however does not limit the domain of considered processes. The algebraic representation of PN is a well suited form for the analysis of the net model behaviour using the following equation

$$M' = M + e[i]C \tag{10}$$

where $e[i]$ is a unit row–vector of size $1 \times m$, which is zero everywhere, except the i–th component corresponding to the transition t_i enabled at marking M.

In consequence, equation (10) can be employed as a basic model in the course of simulation based evaluation of various variants of modelled material flows.

3.1. Algorithm of PN Models Synthesis

According to the introduced algebraic form, an algorithm considered, i.e. determined by transformation (7), should consists of the following three steps.

Step 1. According to a given process specification $PS = (\{PR_i | i = \overline{1,v}\}, \ \varphi, \psi)$, set up the following incidence matrix C of size $m \times n$,

$$C = [C' | C''] \tag{11}$$

whose submatrix C' of size $m \times n'$, where $m = \sum_{k=1}^{v} |PR_k| + v$, $n' = m - v$,

$$C' = \begin{bmatrix} \boxed{C^1} & & & & \\ & \boxed{C^2} & & & \\ & & \ddots & & \\ & & & \boxed{C^k} & \\ & & & & \ddots \\ & & & & & \boxed{C^v} \end{bmatrix} \tag{12}$$

is such that

$$c'_{ij} = \begin{cases} c^k_{rq} & \text{if } i = \sum_{w=1}^{k-1} m_w + r \text{ and } j = \sum_{w=1}^{k-1} n_w + q \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

and elements of its each submatrix C^k, of size $m_k \times n_k$, are determined as follows

$$c^k_{rq} = \begin{cases} 1 & \text{if } r = q \quad \text{and } r = \overline{1, m_k - 1} \\ -1 & \text{if } r - 1 = q \quad \text{and } r = \overline{2, m_k} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

and where $m_k = |PR_k| + 1$, $n_k = m_k - 1$.

In turn, submatrix C'' of size $m \times n''$, where $n'' = ||R||$

$$C'' = \begin{array}{|c|}\hline \overset{\star 1}{C} \\ \hline \vdots \\ \hline \overset{\star k}{C} \\ \hline \vdots \\ \hline \overset{\star v}{C} \\ \hline \end{array} \tag{15}$$

such that

$$c''_{ij} = \overset{\star v}{c}_{rq} \qquad \text{for} \qquad i = \sum_{w=1}^{k-1} m_w + r \tag{16}$$

and elements of its each submatrix $\overset{\star k}{C}$, of size $m_k \times n''$, are determined as follows

$$\overset{\star k}{c}_{rq} = \begin{cases} \psi(u,k,j) & \text{if } r = u, \quad \text{and } q = n' + j \quad \text{for } R_j \in \text{crd}^2\text{crd}^u PR_k \\ -\psi(u,k,j) & \text{if } r = u+1, \quad \text{and } q = n' + j \quad \text{for } R_j \in \text{crd}^2\text{crd}^u PR_k \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

Step 2. According to a given function φ, the capacity function K is determined as follows

$$\begin{aligned} K(n'+k) &= \varphi(R_k), & \forall k &= \overline{1, \|R\|} \\ K(j) &= 1, & \forall j &= \overline{1, n'} \end{aligned} \tag{18}$$

where $\|R\|$ denotes the cardinality of set R.

Step 3. The initial marking M_0 is determined as a zero–vector

$$M_0(i) = 0, \quad \forall i = \overline{1, n'} \tag{19}$$

According to the above presented steps the transformation (7) is performed.

3.2. Algorithm for PN_{Pr} Models Synthesis

In order to determine an algorithm aimed at the solution of the second task (described by the transformation (8)) for constraints $W(e) = 1$, $e \in E$, the following procedure is introduced.

Step 1. According to a given process specification PS (or PN determined by algorithm $A1$) a resource precedence graph G is designed

$$G = (R, <), \tag{20}$$

where $<\subset R \times R$;

$$(R_i, R_j) \in < \Longleftrightarrow (\exists k = \overline{1,v})(\exists l = \overline{1,|PR_k|})(R_i \in \mathrm{crd}^2\mathrm{crd}^l PR_k \ \&$$
$$\& \ R_j \in \mathrm{crd}^2\mathrm{crd}^{l+1} PR_k), \quad R_i, R_j \in R$$

Then, for each loop L in G its critical capacity $c(L) = n(L) - 1$, where $n(L) = \sum_{R_j \in LG} \psi(u,k,j)$ is the number of units of all resources included in L, is calculated. Of course, each loop $L = (LG, <_{LG})$ is a subgraph of G, i.e. $LG \subset R$ and $<_{LG} \subset <$. For each state S (encompassing the current resources allocation) and loop L, the number of resources belonging to L and allocated to the processes at state S, denoted by $n(S,L)$, is calculated.

Step 2. According to the satisfactory condition for S to be safe (Wójcik and Banaszak, 1991)

IF: for each loop L of G, $n(S,L) \le c(L)$ and for two loops L, L' of G such that L and L' are directly (but in such a way that $LG \cap L'G = \emptyset$ or $\|LG \cap L'G\| = 1$) or indirectly (i.e. via other loops) connected,

$$n(S,L) + n(S,L') \le c(L) + c(L') - 1 \quad (21)$$

THEN: S is safe,

determine then the sets of resources included by loops

$$R(L) = \{R_i \in LG | LG \subset R\} \quad (22)$$
$$R(LL') = \{R_i \in LL'G | LL'G \subset R\}$$

where L and L' are connected loops.

The new enableness conditions are extended for the following predicates

$$F(M,L) = (\sum_{j \in H} M(j) < c(L) \ \& \ H = \{k, l, ..., n\} \&$$
$$\& \ LG = \{R_k, R_l, ..., R_n\}), \ \forall LG \subset R \quad (23)$$

$$F(M,LL') = (\sum_{j \in H} M(j) < c(L) + c(L') - 1 \ \& \ H = \{k, l, ..., n\} \&$$
$$\& \ LL'G = \{R_k, R_l, ..., R_n\}), \ \forall LL'G \subset R$$

In case when one of the above mentioned predicate forms is not true, i.e. when there exists marking M such that for some loop L or connected loops L and L', either $F(M,L)$ or $F(M,LL')$ is not true, then transition $t_i \in T$ is not enabled if there exists $j = \overline{1,n}$ such that $c_{ij} > 0$ and $M(j) < \varphi(R_j)$ and $R_j \in LR$, while such that $c_{ik} < 0$ and $M(k) > 0$ and $R_k \notin LR$, or $c_{ik} \geq 0$ holds, for each $k = \overline{1,n}$.

The predicates provided set up a set Pr of a predicate P/T–net model of admissible, i.e. deadlock–free, material flows of modelled processes. Also, it means that they may be directly applied in the course of calculation of formula (10). Presented steps determine the transformation (8).

3.3. Example of Algorithms Operation

In order to illustrate how the algorithm A1 and A2 can be applied for solution of the task of automatic control flow model synthesis (6) let us consider the following example.

The considered Flexible Machining Cell (FMC), shown in Figure 2, consists of two machine tools M_1 and M_2, and four conveyors $CV_1 - CV_4$, and two industrial robots R_1 and R_2, and a gripper changing station equipped with two grippers G_1 and G_2, respectively.

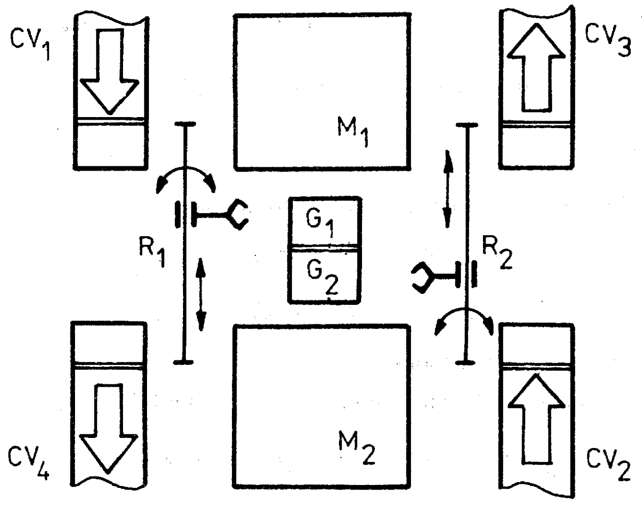

Fig. 2. Flexible machining cell.

Let us assume the following processes specification

$$PS = (\{PR_1,\ PR_2\}, \varphi, \psi) \tag{24}$$

where

$$PR_1 = (O_{1,1}, \{R_1, G_1\}), (O_{1,2}, \{M_1\}), (O_{1,3}, \{R_2, G_2\})$$

$$PR_2 = (O_{2,1}, \{R_2, G_2\}), (O_{2,2}, \{M_2\}), (O_{2,3}, \{R_1, G_1\})$$

$$\varphi(R_1) = \varphi(R_2) = \varphi(M_1) = \varphi(M_2) = \varphi(G_1) = \varphi(G_2) = 1$$

$$\psi(1,1,1) = \psi(1,1,2) = \psi(2,1,3) = \psi(3,1,5) = \psi(3,1,6) =$$

$$\psi(1,2,5) = \psi(1,2,6) = \psi(2,2,4) = \psi(3,2,1) = \psi(3,2,2) = 1$$

and such that the elements of the following set of system components $\{R_1, G_1, M_1, M_2, R_2, G_2\}$ are ordered in the same way as the elements of ordered set $\{1,2,3,4,5,6\}$. It means that in the FMC considered two concurrent, pipeline-like flowing processes are performed. According to $PR_1(PR_2)$ each workpiece is picked up from CV_1 (CV_2) by R_1 (R_2) equipped with G_1 (G_2) and then transported to M_1 (M_2). After the machining operation is completed the workpiece is removed from M_1 (M_2) and placed on CV_3 (CV_4) by R_2 (R_1) equipped with G_2 (G_1).

According to the algorithm which realizes transformation (7) the PN of the following algebraic representation is obtained

$$PN = (C, K, M_0) \tag{25}$$

where

$$\begin{array}{c|rrr|rrr|rrrrrr}
t_j \setminus p_i & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 \\
3 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 \\
4 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\
\hline
5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\
7 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & -1 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0
\end{array}$$

$C =$ (matrix above),

$$(\forall i = \overline{1,12})(K(i) = 1 \ \& \ M_0(i) = 0)$$

The graphical representation of the P/T–net model obtained is shown in Figure 3.

It is easy to note that places p_1, p_2 and p_3 (p_4, p_5 and p_6) correspond to conditions encompassing the executions of a workpiece transportation from CV_1 (CV_2) to M_1 (M_2) (and performed by R_1 (R_2) equipped with G_1 (G_2)), its machining on M_1 (M_2), and then its transportation from M_1 (M_2) to CV_3 (CV_4) along pR_1 (PR_2). In turn, places $p_7 - p_{12}$ correspond to R_1, G_1, M_1, M_2, R_2 and G_2, respectively. Transitions $t_1 - t_4$ ($t_5 - t_8$) encompass events occuring along pR_1 (PR_2) and associated with taking a workpiece from CV_1 (CV_2), its fixing on M_1 (M_2) and begining of the machining, then finishing and taking the workpiece from M_1 (M_2), and at the end its placing on CV_3 (CV_4), respectively.

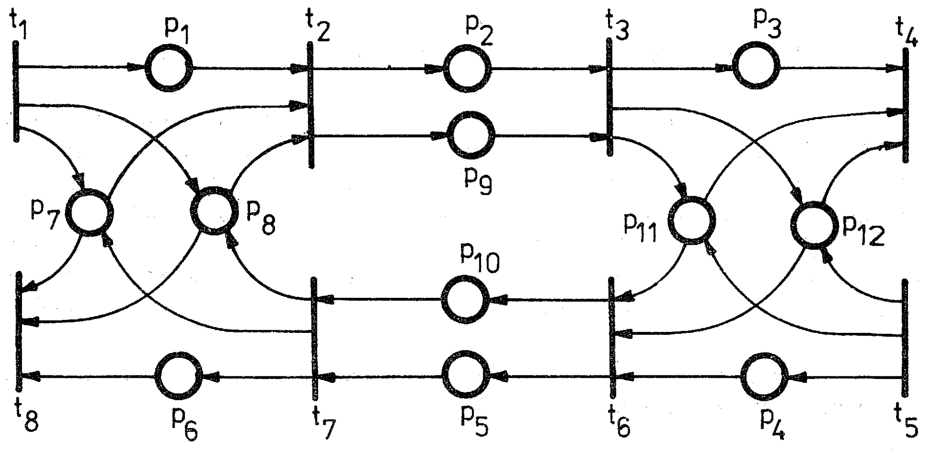

Fig. 3. Graphical representation of P/T–net determined by (25).

Due to the first step of the algorithm which realizes transformation (8) and according to process specification (24) the resource precedence graph, shown in Figure 4, can be considered.

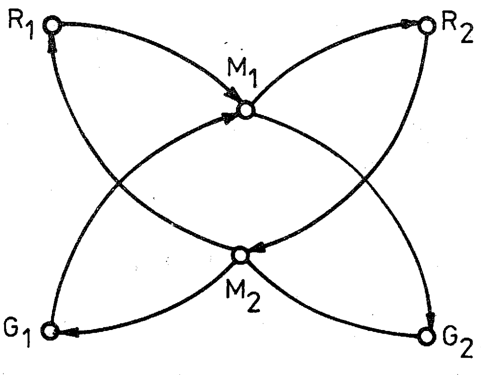

Fig. 4. Resource precedence graph obtained from (24).

It is easy to note that

IF $M = (1,1,0,0,1,0,1,1,1,1,0,0)$ AND ($LG = \{R_1, M_1, R_2, M_2\}$ OR $LG' = \{R_1, M_1, G_2, M_2\}$ OR $LG'' = \{G_1, M_1, R_2, M_2\}$ OR $LG''' = \{G_1, M_1, G_2, M_2\}$)

THEN transition t_1 of P/T–net from Figure 3 is not enabled to fire because predicate $F(M,L)$, defined by (27) is not true,

and analogously

IF $M = (0,1,0,1,1,0,0,0,1,1,1,1)$ AND ($LG = \{R_1, M_1, R_2, M_2\}$ OR $LG' = \{R_1, M_1, G_2, M_2\}$ OR $LG'' = \{G_1, M_1, R_2, M_2\}$ OR $LG''' = \{G_1, M_1, G_2, M_2\}$)

THEN transition t_5 of P/T–net from Figure 3 is not enabled to fire because predicate $F(M,L)$, defined by (23) is not true.

It follows that PN_{Pr} determined by (25) and predicates (23) can be treated as a net model of admissible realizations of the processes considered. Its reachability tree is shown in Figure 5, where for simplicity of further considerations we introduce the following notation, according to which the states are determined by indices of first six coordinates of markings M such that $M(i) > 0$, $i = \overline{1,6}$. For instance, to marking $M = (1,1,0,0,1,0,1,1,1,1,0,0)$ the following notation (1,2,5) is corresponding.

The algorithms introduced aimed at automatic synthesis of PN_{Pr} models of admissible control flows are the core of an approach to automation aimed at adaptive control programming.

4. Adaptive Control Procedure

The presented technique is computationally efficient and can be easy and successfully applied in many cases related to synchronization of competing processes. Besides an application to designing of self–programming logical controllers and computer aided modelling and performance evaluation tools (Banaszak and Roszkowska, 1988; Banaszak, 1990) the approach seems to be very useful for automatic synthesis of adaptive controllers capable to supervise the processes execution as well as to maintain the system activities in the case when its components breakdown occur.

Our concept employed in the course of an adaptive controller design is based on an idea of repetitive synthesis of control procedures caused by the current changes of system resources availability, and then updating (replacement) of the former control procedures being not adequate to the changes occured.

4.1. Illustration of Self–Reprogramming Concept

It is assumed that the functional redundancy of machine tools, allowing their mutual replacement, is provided. Let us assume also that during the execution of processes specified by (24) the malfunction of machine M_2 is recognized.

According to the preassumed functional redundancy the identified breakdown leads to the following modified processes specification

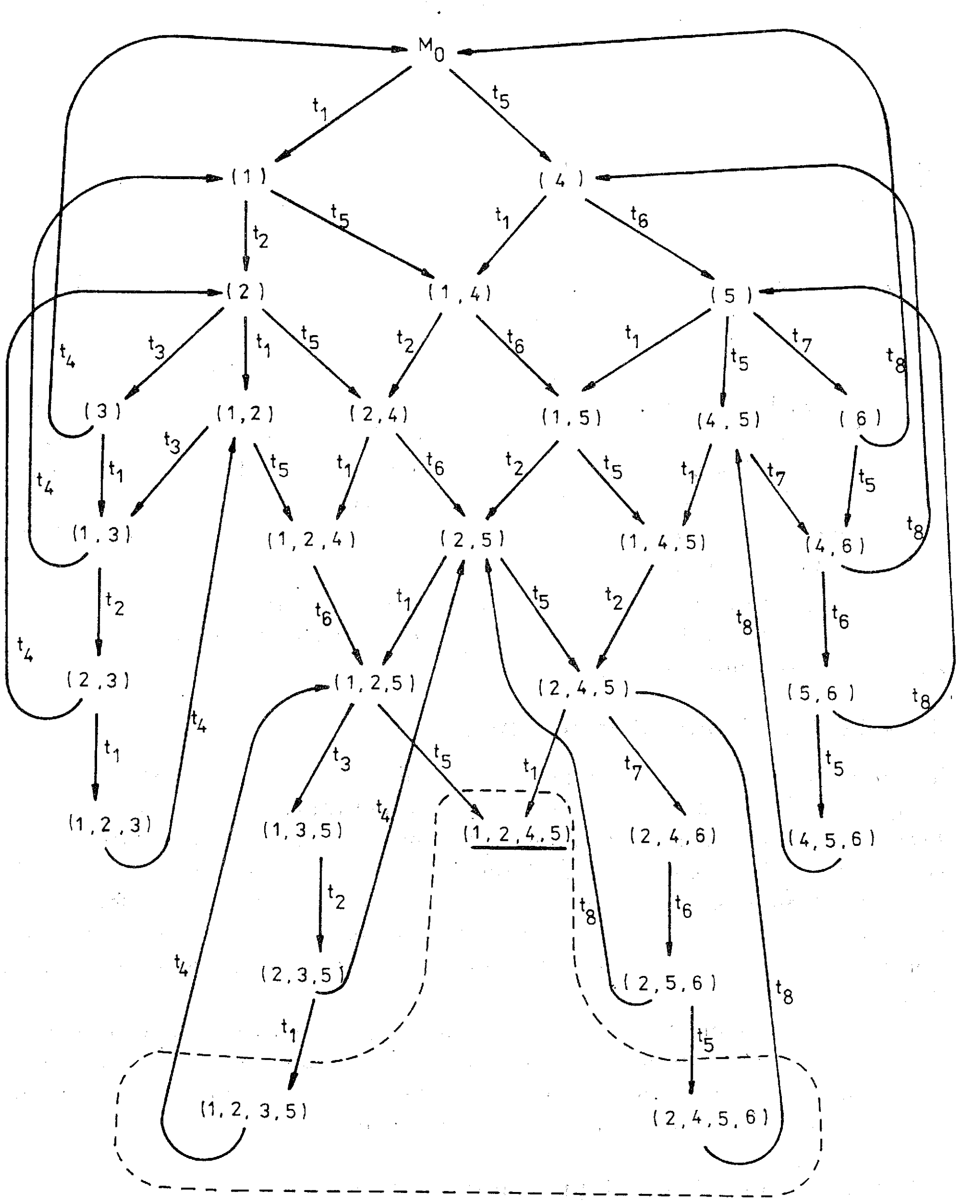

Fig. 5. Reachability tree of the P/T–net from Figure 3. The dashed line includes the markings which are forbidden by predicates (23) of PN_{Pr}. Underlined markings depict deadlocks.

$$PS' = (\{PR'_1, PR'_2\}, \varphi, \psi) \tag{26}$$

where

$$PR'_1 = (O_{1,1}, \{R_1, G_1\}), (O_{1,2}, \{M_1\}), (O_{1,3}, \{R_2, G_2\})$$

$$PR'_2 = (O_{2,1}, \{R_2, G_2\}), (O_{2,2}, \{M_1\}), (O_{2,3}, \{R_1, G_1\})$$

$$\varphi(R_1) = \varphi(R_2) = \varphi(M_1) = \varphi(G_1) = \varphi(G_2) = 1$$

$$\psi(1,1,1) = \psi(1,1,2) = \psi(2,1,3) = \psi(3,1,5) = \psi(3,1,6) =$$

$$\psi(1,2,1) = \psi(1,2,6) = \psi(2,2,4) = \psi(3,2,1) = \psi(3,2,2) = 1$$

and such that the elements of the folowing set of system components $\{R_1, G_1, M_1, R_2, G_2\}$ are ordered in the same way as the elements of ordered set $\{1,2,3,5,6\}$.

The resource precedence graph corresponding to process specification (26) is shown in Figure 6, while the corresponding reachability graph of the final control flow algorithm is shown in Figure 7.

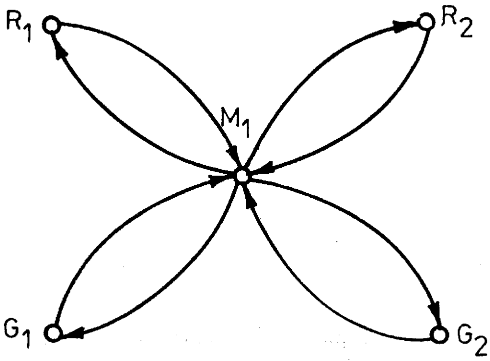

Fig. 6. The resource precedence graph corresponding to (26).

Let us assume that machine M_2 was broken down at the moment when along the production route PR_1 the only one workpiece (being machined on M_2) has been processed, while along the route determined by P_2R the two workpieces (machined on M_2 and transported from CV_3 to M_2 by R_2 equipped with G_2) have been processed simultaneously. Such situation corresponds to the following marking $M = (0,1,0,1,1,0,0,0,1,1,1,1)$ included by reachability graph from

Figure 5, where it is encoded as a marking (2,4,5). Of course, this marking is not included in reachability graph from Figure 7.

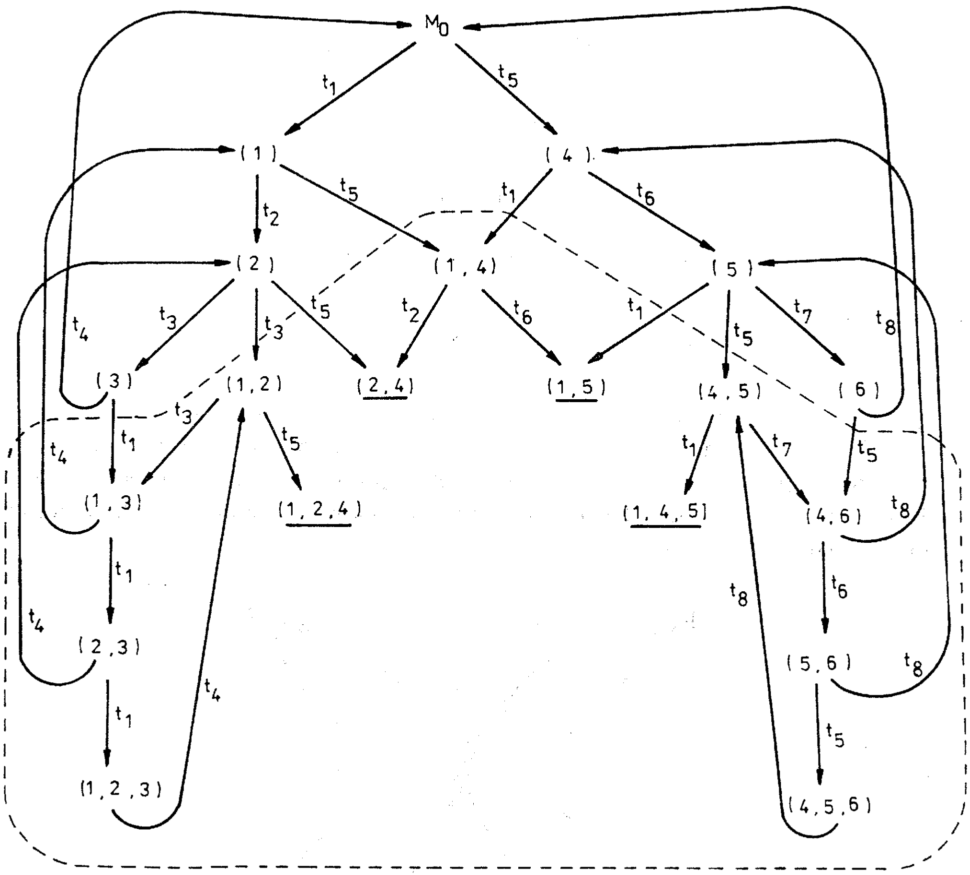

Fig. 7. The reachability graph of PN_{Pr} model of modified control procedure. The dashed line includes the markings which are forbidden by predicates (23) of PN_{Pr}. Underlined markings depict deadlocks.

Let us note that since the markings from reachability graph shown in Figure 7 do not include the coordinates corresponding to places p_5 and p_{10}, i.e. to the places encompassing the state of M_2 operation, the markings that only can be considered in order to continue processes execution are the following ones
$M' = (0,1,0,0,0,0,0,0,1,0,0,0)$ and $M'' = (0,0,0,1,0,0,0,0,0,0,1,1)$.
It means that one of these markings can serve as an initial marking of PN'_{Pr} model replacing the former one corresponding to the situation before machine M_2

breakdown occurrence, and only one of them should be selected. This decision making problem can be treated as a problem of optimization. Depending on a given criterion, for instance being a measure of cost required in order to remove a workpiece from a given technological device, and/or costs required by the workpieces processing, an appriopriate, best, decision should be undertaken. The example provided above can be seen as an illustration showing the way of possible fault–recovery procedure operation.

4.2. Control Procedure

The problem we consider now concerns finding out of an algorithm useful for the automatic replacement of the procedure being executed until a breakdown occurs, by the newly developed one. To solve this problem, the following three stage procedure is proposed.

Stage of Identification

Determine which system component has broken down, and which other components can take over its functions. If there is no component capable to replace the one broken down then some of the processes performed in the system cannot be continued, else go to Decision stage.

Stage of Decision

From the set of system components capable to replace the broken down component, choose the best one (in appriopriate sense) and then set up a new control procedure and go to Modification stage.

Stage of Modification

Suspend the realization of to–be–replaced control procedure, and then after the apdating of the workpiece displacement, start the newly obtained control procedure.

Note that in the procedure considered the PN_{Pr} net model of possible control flows and the synchronization mechanism are combined together and can constitute the new (adapted to the changes occured) control procedure. Moreover, such a procedure reflecting some of the feasible material flows, after a dispatching rule implementation, can be applied directly as a real–time (time–independent) control program for an industrial controller.

5. Concluding Remarks

The approach presented above is our contribution to the automatic synthesis of adaptive control programs procedures. Its objective is to provide the system recovery algorithms aimed at the control of discrete systems which consist of functionally redundant components.

In other words, the approach we are offering here allows to realize the automatic conversion of an input problem (stated by the controlled processes specification) into a computer program suitable for the intended application. It allows us to reduce the time and effort involved in iterative (statement by statement) software completion as well as to release programmers from debugging and program correctness examination. The solution proposed is based on the concept of functional redundancy of the controlled system's components. This means that some functions of the production system can be performed alternatively by different components, though with another efficiency. It implies that the main system's functions will be maintained, i.e. fault–tolerant operation will be maintained, as far as internal or external disturbances will not exhaust the preassumed, still available system's functional redundancy.

An alternative approach which could be considered is based on assumption that input processes specification captures the information defining which objects and which of their functions are equivalent. Such an extended specification allows one to create control procedures which include in their structure the preassumed functional and/or structural redundancy of the system components. It permits in case of occurrence of partial or total damage of device, to execute alternative control (allowed by model structure built–in redundancy) in order to maintain the basic systems functions.

The application perspective of the results obtained concerns the automated programming of the reliable real–time industrial controllers as well as the designing of computer–aided tools aimed at modelling and performance evaluation of FMSs. We belive that the approach presented may be extended to other areas where deadlock avoidance is critical, such as message routing in data communication networks and multiprogramming operating systems of multiprogrammed computing systems.

It remains also to solve a designing problem of adaptive control procedures capable to work in the on–line mode, when new processes are introduced to the system which actually performs other operations.

It should be born in mind, however, that our considerations only concern the asynchronous control strategy of sequential processes cooperation. Thus, as a subject of further work, it remains to find out methods aimed at the design of models which are capable to cope with the processes specified by partially ordered sets of operations.

References

Banaszak Z. (1987): *Automatic design of adaptive control algorithms.*– in: Automatic Control World Congress 1987, 10th Triennal World Congress of the IFAC, Munich FRG, v.7, Oxford: Pergamon Press 1988, IFAC Proc. series, No.12, pp.275–280.

Banaszak Z. (1990): *CAMPES: a computer–aided modelling and performance evaluation system for FMS.*– Proc. 4th Int. Conf. on Integrated Problems of Industrial Control, Kiev, Ukraine, v.3, pp.177–181.

Banaszak Z. (Ed.) (1991): *Modelling and Control of FMS (Petri net approach).*– Wrocław: Wrocław Technical University Press.

Banaszak Z. and Mowafak H. Abdul–Hussin (1988): *Petri net approach to automatic real–time program synthesis.*– Control and Cybernetics, v.17, No.4, pp.361–375.

Banaszak Z. and Krogh B.H. (1990): *Petri net approach to deadlock handling problem.*– Proc. 5th Int. Symp. on Computer and Information Sciences, Cappadocia, Nevsehir, Turkey, v.2, pp.785–793.

Banaszak Z. and Krogh B.H. (1991): *Real–time synchronization mechanism for complex concurrently competing processes coordination.*– App. Math. and Comp. Sci., v.1, No.1, pp.37–52.

Banaszak Z. and Roszkowska E. (1988a): *Deadlock avoidance in pipeline concurrent processes.*– Podstawy Sterowania, No.18, pp.3–17.

Banaszak Z. and Roszkowska E. (1988b): *System of computer production planning.*– Proc. 3rd Int. Symp. on Computer and Information Sciences, Cesme, Izmir, Turkey, pp.819–827.

Chu Zhou M., DiCesare F. and Rudolph D. (1990): *Control of a flexible manufacturing systems using Petri nets.*– Prep. 11th IFAC World Congress on Automatic Control in the Service of Mankind, Tallinn, Estonia. Eds V.Utkin and U.Jaaksoo, v.9, pp.43–48.

Duggan J. and Browne J. (1988): *ESPNET – expert system based simulator of Petri nets.*– IEEE Proc., v.135, No.4, pp.239–247.

D'Souza K.A. (1991): *Modeling Controls for Computer Integrated Assembly Cells Using Petri Nets.*– Ph.D. Dissertation Proposal, Department of Industrial Engineering, University of South Florida, Tampa, Florida, U.S.A.

Feldbrugge F. (1988): *Petri net tool overview 1989.*– in: Lecture Notes in Computer Science, Eds G.Goos and J.Hartmanis, No.424, Advances in Petri Nets, 1989 (Ed.) Grzegorz Rozenberg, Berlin: Springer–Verlag, pp.151–178.

Gentina J.C., Bourey J.P. and Kapusta M. (1988): *Coloured adaptive structured Petri–net. A tool for the automatic synthesis of hierarchical control of flexible manufacturing systems* .– Computer–Integrated Manufacturing Systems, v.1, No.1, pp.39–47.

Mamath M. and Vishwanadham N. (1986): *Applications of Petri net based models in the modeling and analysis of flexible manufacturing systems.*– IEEE Int. Conf. Robotics and Automation, pp.312–317.

Murata T. (1989): *Petri nets: properties, analysis and applications.*– Proc. of IEEE, v.77, No.4, pp.541–580.

Pathak D.K. (1988): *Automated development of control software for discrete manufacturing systems.*– Master's thesis, Carnegie–Mellon Univ, USA.

Peterson J.L. (1985): *Petri Nets (An introduction)* . – Berlin: Springer–Verlag, New York: Heidelberg.

Roszkowska E. and Banaszak Z. (1988): *Computer–aided control and management system for FMSs: A Petri net approach.*– in: Systems Prospects. The Next Ten Years of System Research. Proc. of the United Kingdom Systems Society Conference. UKSS, Hull England, New York, Plenum Press, 1989, pp.297–300.

Shibata K., Ueda Y., Yuyama S. and Hasegawa H. (1988): *A study on verification of service specification in communication software development.*– Trans. of IEICE, v.E–71, No.12, pp.1203–1211.

Willson R.G. and Krogh B.H. (1990): *Petri net tools for the specification and analysis of discrete controllers.*– IEEE Trans. on Software Engineering, v.16, No.1, pp.39–50.

Wójcik R. and Banaszak Z. (1991): *Deadlock handling methods for pipeline concurrently flowing processes.*– Scientific Papers of Institute of Engineering Cybernetics of the Technical University of Wroclaw, Poland, No.89, Wroclaw, pp.53–58.