Ph. D. Robert Wójcik
Institute of Engineering Cybernetics (I-6)
Wrocław University of Technology

CP APPROACH TO DESIGN OF REPETITIVE MANUFACTURING PROCESSES

A class of repetitive concurrent manufacturing processes using common resources in mutual exclusion is considered. A system with any resource requests structure can be seen as a composition of subsystems with n cyclic processes sharing one resource. A problem of finding a schedule such that no process waits for access to the resources has been formulated as a task of constraint programming. State space of the problem's solutions has been reduced by use of necessary and sufficient conditions for existence of a waiting-free n-process realisation. An example illustrating a method of problem solving using interval constraint logic programming has been presented.

ZASTOSOWANIE METODY PROGRAMOWANIA Z OGRANICZENIAMI DO PROJEKTOWANIA POWTARZALNYCH PROCESÓW PRODUKCYJNYCH

Rozpatrywana jest klasa powtarzalnych, współbieżnych procesów produkcyjnych wykorzystujących wspólne zasoby w trybie wzajemnego wykluczania. System o dowolnej strukturze żądań zasobowych procesów może być rozpatrywany jako połączenie podsystemów złożonych z n procesów cyklicznych współdzielących jeden zasób. Problem wyznaczania harmonogramu, który gwarantuje nie występowanie oczekiwania procesów na dostęp do zasobów sformułowano jako zadanie programowania z ograniczeniami. Przestrzeń rozwiązań problemu zmniejszono dzięki wykorzystaniu warunków koniecznych i wystarczających dla istnienia realizacji n-procesu, w której żaden proces nie czeka na dostęp do zasobów. Przedstawiono przykład ilustrujący metodę rozwiązywania problemu w oparciu o programowanie logiczne z ograniczeniami zadanymi w postaci przedziałów.

1. INTRODUCTION

Planning production flow in discrete production systems with concurrently executed processes using common resources in mutual exclusion requires solving a problem of resource conflicts resolution [3], [6], [9]. A solution of this problem is the best schedule taking into account certain evaluation criterion, defining an order of using the resources, e.g. machines, stores, tools, by the processes and guaranteeing deadlock-free and starvation-free execution of the processes.

Processing a batch of work-pieces according to a given production route, defining a sequence of operations provided for the final product, creates a set of repetitive production processes. Each operation in the route is using one production resource for a certain amount of time. Assuming that the following work-piece is introduced to the system after finishing the previous one the cyclic process is created with a cycle time equal to the sum of the operation times specified in the executed production route. In case when several products is processed at the same time the production system can be seen as a system of concurrent cyclic processes sharing resources in mutual exclusion [2], [3].

In this work systems of cyclic processes with no deadlock possibility are considered [3], [9]. This type of systems in many cases can be analysed as composed of subsystems consisting of several processes sharing one resource. The problem to be solved in this work consists in finding a schedule with no process waiting for access to the resources and therefore does not require resource conflict resolution. In particular for a given system of n cyclic processes sharing a resource and fixed operation times the starting times of the processes such that a waiting-free schedule exists are looked for [11]. Assuming that the operation times can be chosen from the bounded set of discrete values solving the problem is equivalent to design of a production system with the following properties:

- The processes will never wait for access to the resources;
- The initial state of the system, defined by starting times of production tasks, belongs to the system's cyclic steady-state;
- The system's cycle time is equal to the least common multiple of cycle times of the processes.

In order to find a waiting-free schedule for the n-process system a model based on modulus equations presented in [10], [11] will be used. The model allows define a set of constraints on starting times of the processes, which guarantee existence of a waiting-free schedule. These constraints can be implemented by a program defined in a constraint logic programming language Oz [4], [11] solving the problem using its predefined searching procedures based on interval analysis and reduction of domain of decision variables [7]. For the operations times given the constraint programming method proposed allows finding the all possible solutions, i.e. a set of all starting times of the processes leading to waiting-free execution of the system (if exist), or find the specific solution, e.g. the first solution fulfilling a certain criterion (if exists).

2. SYSTEM OF PROCESSES

A system of repetitive manufacturing processes consists of a set of processes sharing common resources in mutual exclusion; see Fig.1. Each process P_i, (i=1,2,...,n), representing one product processing, executes periodically a sequence of the operations using resources defined by $Z_i = (R_{i1}, R_{i2}, ..., R_{il(i)})$, where $l(i)$ denotes a length of production route. The operations times are given by a sequence $ZT_i = (r_{i1}, r_{i2}, ..., r_{il(i)})$, where $r_{i1}, r_{i2}, ..., r_{il(i)} \in N$ are defined in the uniform time units (N – set of natural numbers). For instance the system shown in Fig.1 consists of ten resources and seven processes. The resources R_1, R_2, R_3, R_4 are shared ones, since each one is used by at least two processes, and the resources R_5, R_6, R_7, R_8, R_9, R_{10} are non-shared because each one is exclusively used by only one process. The processes P_1, P_2, P_3, P_4, P_5, P_6, P_7 are executing operations using resources given by the sequences: $Z_1 = (R_1, R_2, R_3, R_4)$, $Z_2 = (R_1, R_5)$, $Z_3 = (R_1, R_6)$, $Z_4 = (R_1, R_7)$, $Z_5 = (R_2, R_8)$, $Z_6 = (R_3, R_9)$ and $Z_7 = (R_4, R_{10})$. The system considered can be seen as composed of four subsystems each one with n cyclic processes sharing a single resource. The n-process subsystems are defined as follows: subsystem $S_1 = (P_1, P_2, P_3, P_4)$ – the processes are using resource R_1; subsystem $S_2 = (P_1, P_5)$ – the processes are using resource R_2; subsystem $S_3 = (P_1, P_6)$ – the processes are using resource R_3; subsystem $S_4 = (P_1, P_7)$ - the processes are using resource R_4. Because the n-process subsystems have no common resources it is possible to analyse their behaviour separately to find the initial resource allocation times of the processes for which waiting-free schedules exist. The schedules designed for each subsystem can be joined together to obtain a waiting-free schedule for the whole system.

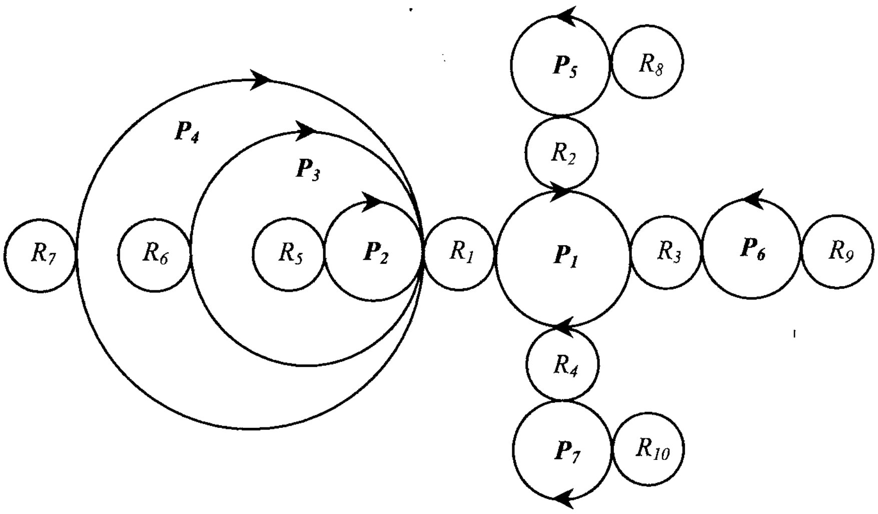

Fig.1. System of repetitive manufacturing processes with concurrency: P_1, P_2, P_3, P_4, P_5, P_6, P_7 – processes; R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10} – resources.

The n-process system $(P_1,\dots,P_i,\dots,P_j,\dots, P_n)$ consists of n cyclic processes sharing a single resource, e.g. subsystem of processes (P_1, P_2, P_3, P_4) sharing resource R_1 shown in Fig.1. Each process P_i $(i=1,2,\dots,n)$ executes periodically a sequence of the operations using resources defined by $Z_i = (R, O_i)$, where R denotes a shared resource used by the processes and O_i denotes a non-shared resource used by process P_i. The operations times are given by a sequence $ZT_i = (r_i, o_i)$, where $r_i, o_i \in N$. A cycle time of P_i is defined by relation $c_i = r_i + o_i$. For instance in the subsystem (P_1, P_2, P_3, P_4) (Fig.1) the shared resource $R=R_1$, the resources R_2, R_3, R_4 are represented by the non-shared resource O_1, the resources $O_2=R_5$, $O_3=R_6$, $O_4=R_7$.
The first operation executed by each processes (the sequence Z_i always begins with shared resource R) can be initiated at different times in relation to time $t_p=0$. In the following it will be assumed that one of the processes, e.g. the process with the greatest cycle time, always starts at time $t_p=0$ and the other processes start at times $t \geq t_p$. It was

shown [10], [11] that behaviour of the n-process system depends on the operation times and starting times (phases) of the component processes. The system's dynamics can be analysed taking into account dynamics of each 2-process subsystem [1]. In the following some results used to solve the problem of waiting-free schedule design will be recalled.

3. MODELLING DYNAMICS

A natural model for behaviour analysis of discrete processes with periodicity is modulus algebra [5], [8]. Using its properties it is possible to design recurrent module equations defining times of any process resource request in relation to a chosen process request and to find conditions guaranteeing waiting-free execution of the n-process system.

3.1. Basic properties of modulus algebra

It is known that for any integer $a \in Z$ and any $p \in N$ ($p>0$) the following relation holds:

$$a = wp + r \tag{1}$$

A number $w \in Z$ is a quotient and number $r \in Z$ & $0 \leq r < p$ is a remainder [8]. The representation (1) is unique since for a given p it is $w = a \text{ div } p$ and $r = a - wp = a \bmod p$.
Two integers $a,b \in Z$ are considered modulus $p \in N$ equal if $a = b + kp$ and $k \in Z$. This special equality is called "congruence". It would be said that a and b are congruent (module equal) with respect to p. Congruence is defined with respect to a given p, and any and all values of $k \in Z$. A modular equality of a and b is written as $a = b \pmod p$ or $a \equiv b \pmod p$. Two integers $a,b \in Z$ congruent with respect to p, divided by p, have the same remainder $0 \leq r < p$. This follows from relations $a = wp + r$ & $a = b + kp$, hence $b = wp - kp + r = (w-k)p + r$. The last property can be written using mod operator: $r = a \bmod p = b \bmod p$. It can also be noticed that:

$$\forall a \in Z \ \& \ \forall p \in N \ \text{ if } 0 \leq a < p, \text{ then } a \bmod p = a \tag{2}$$

For the n-process system the modulus algebra can be used to derive shifts between times of resource requests of any process and the nearest resource allocation times of a chosen process. The shifts will be used as local starting times of the processes.

3.2. Local starting times of the processes

Consider a system of processes $(P_1,\ldots,P_i,\ldots,P_j,\ldots, P_n)$ sharing a single resource (Fig.1). Let $x_i(k) \in N \cup \{0\}$, where $k=0,1,2,3,\ldots$, denote times at which a process P_i, where $i \in \{1,2,\ldots,n\}$, requests access to the shared resource and $a_i(k) \in N \cup \{0\}$ times at which it receives access to the resource. There is $0 \leq x_i(k) \leq a_i(k)$ and it is assumed that a starting time of a chosen process P_i is equal to $x_i(0) = 0$ and a starting time of any process P_j, where $j \neq i$ is such that $x_j(0) \geq 0$.
Assuming that the processes are executed independent each other (no resource sharing) subsequent resource requesting times $x_i(k)$ are equal to the allocation times $a_i(k)$ and can be calculated according to the equation $x_i(k+1) = a_i(k+1) = a_i(k) + c_i$ (Fig.2). Therefore, $x_i(k) = a_i(k) = a_i(0) + k * c_i$. In case of concurrent execution of processes the relevant

formula has to take into account a waiting time $w_i(k)$. Hence, $a_i(k+1) = x_i(k+1) + w_i(k) = a_i(k) + c_i + w_i(k)$. A parameter $t_{ij}(l) \in N \cup \{0\}$ (Fig.2), where $l=0,1,2,...$, defines distance between a resource request time $x_j(l)$ of the process P_j and the nearest resource allocation time $a_i(k) \le x_j(l)$ of the process P_i. The shift $t_{ij}(l)$ can be used as a local starting time of P_j in relation to resource allocation time of P_i.

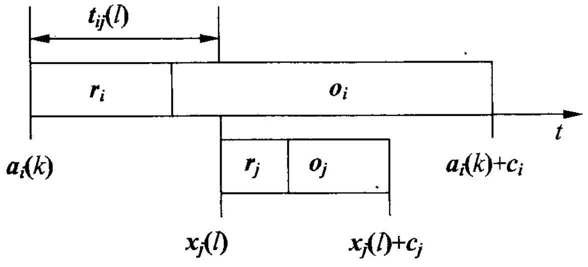

Fig.2. Resource request/allocation times.

Previous research showed that for any 2-process subsystem (P_i, P_j) of the n-process system, where $i \neq j$ & $i,j \in \{1,2,...,n\}$, it is possible to derive values $t_{ij}(l)$, $l=0,1,2,...$, using recurrent modulus equations [10], [11]. In case when each process execution is independent to the others then resource request times are equal to resource allocation times, i.e. $x_j(l)=a_j(l)$. Therefore, times $t_{ij}(l)$ can be calculated from the equation:

$$t_{ij}(l) = x_j(l) \bmod c_i = a_j(l) \bmod c_i = [a_j(0) + lc_j] \bmod c_i \ \& \\ \& \ t_{ij}(l) \in [0,c_i) \ \& \ a_i(0)=0 \ \& \ a_j(0) \ge 0 \qquad (3)$$

It can be proven [10], [11] that for independent execution of the processes (no resource sharing) resource request times of process P_j, calculated in relation to resource allocation times of process P_i, can occur only at local times $t_{ij}(l) \in [0,c_i)$ (3) given by formula:

$$t_{ij}(l) = a_j(l) \bmod c_i = f_{ij}(l)D_{ij} + y_{ij}(l) \ \& \ l=0,1,2,... \qquad (4)$$

where $D_{ij}=D_{ji}=\text{g.c.d.}(c_i,c_j)$ & $c_i=D_{ij}m_{ij}$ & $c_j=D_{ji}m_{ji}$ & g.c.d.$(m_{ij},m_{ji})=1$ & $m_{ij},m_{ji} \in N$ &
& $f_{ij}(l)=[f_{ij}(0) + lm_{ji}] \bmod m_{ij}$ & $y_{ij}(l)=y_{ij}(0)$ &
& $t_{ij}(0)=a_j(0) \bmod c_i$ & $f_{ij}(0)= t_{ij}(0) \text{ div } D_{ij}$ & $y_{ij}(0)=t_{ij}(0) \bmod D_{ij}$ &
& $0 \le f_{ij}(0) < m_{ij}$ & $0 \le y_{ij}(0) < D_{ij}$ & g.c.d. – the greatest common divisor.

Let $W_{ij}=\{0,1,...,m_{ij}-1\}$. From $f_{ij}(l)= [f_{ij}(0) + lm_{ji}] \bmod m_{ij}$ (4) it follows that a range of $f_{ij}(l)$ is such that $f_{ij}(l) \in W_{ij}$. It can be shown [10] that $f_{ij}(l)$ achieves periodically, with period m_{ij}, all values from the set W_{ij}, i.e. $f_{ij}(l)=f_{ij}(l+k*m_{ij})$ for $k=0,1,2,...$ and $l=0,1,2,...,m_{ij}-1$. Therefore, function $f_{ij}(l)$ defines permutation of the set W_{ij}.
By symmetry it is also possible to define local starting times $t_{ji}(l) \in [0,c_j)$ as times of resource requests $x_i(l)$ of process P_i in relation to the nearest resource allocation times $a_j(k) \le x_i(l)$ of process P_j. The corresponding formula is given according to (4) by exchanging the indexes i, j.

$$t_{ji}(l) = a_i(l) \bmod c_j = f_{ji}(l)D_{ji} + y_{ji}(l) \ \& \ l=0,1,2,... \quad (5)$$

It is possible to show [10] that the following reverse transformations hold:

$$f_{ij}(l)=[m_{ij} - f_{ij}(l) - (D_{ij} - y_{ij}(l)) \text{ div } D_{ij}] \bmod m_{ji} \ \& \ y_{ji}(l)=[D_{ij} - y_{ij}(l)] \bmod D_{ij} \quad (6)$$

The presented formulas can be used to define conditions, which fulfilled, guarantee waiting-free execution of the processes.

4. PROGRAMMING CONSTRAINTS FOR WAITING-FREE EXECUTION OF PROCESSES

Dynamics of the n-process system depends on a dynamics of its 2-process subsystems (P_i, P_j), where $i \neq j$ & $i,j \in \{1,2,...,n\}$. It was shown [10] that behaviour of each subsystem (P_i, P_j) can be analysed taking into account the operation times and local starting times $t_{ij}(l) \in [0,c_i)$ (4) of process P_j, calculated in relation to resource allocation times of P_i, or local starting times $t_{ji}(l) \in [0,c_j)$ (5) of the process P_i, calculated in relation to resource allocation times of P_j. In particular the following theorems define constraints for existence of waiting-free schedules of the 2-process system (P_i, P_j).

Theorem 1. A waiting-free schedule will be executed by the 2-process system (P_i, P_j) if and only if exists starting time $t_{ij}(0) \in [0,c_i)$, where $t_{ij}(0) = f_{ij}(0)D_{ij} + y_{ij}(0)$ (4), such that

$$r_i \leq y_{ij}(0) \leq D_{ij} - r_j \quad (7)$$

When the condition (7) holds then the 2-process system's cycle time is equal to T_{ij} = l.c.m.$(c_i,c_j) = (c_i * c_j)/D_{ij}$, where l.c.m.$(c_i,c_j)$ denotes the least common multiple.
If the Theorem 1 holds, then also theorem taking into account $t_{ji}(0)$ (5) holds. This is because from (6) $y_{ji}(l)=[D_{ij} - y_{ij}(l)] \bmod D_{ij}$ and for $l=0$ from (7) $y_{ij}(0) \leq D_{ij} - r_j < D_{ij}$ there is $y_{ji}(0) = D_{ij} - y_{ij}(0)$ (2). Hence, from (7) $r_i \leq D_{ij} - y_{ji}(0) \leq D_{ij} - r_j$. Since, $D_{ij} = D_{ji}$ there is

$$r_j \leq y_{ji}(0) \leq D_{ji} - r_i \quad (8)$$

Theorem 2. The conditions (7) and (8) are equivalent, i.e. $r_i \leq y_{ij}(0) \leq D_{ij} - r_j$ if and only if $r_j \leq y_{ji}(0) \leq D_{ji} - r_i$.

The n-process system consists of $n(n-1)/2$ different 2-process subsystems (P_i, P_j) defined by $i<j$ & $i,j \in \{1,2,...,n\}$. If for each (P_i, P_j) constraint (7) holds, then the processes will never wait for access to the resources. This configuration is stable since processes do not disturb each other and therefore positions of any two processes will be not changed. Taking into account Theorem 2 the necessary and sufficient condition for existence of waiting-free schedule for the n-process system is given by the following theorem [11].

Theorem 3. A waiting-free schedule will be executed by the n-process system if and only if for each subsystem (P_i, P_j), where $i<j$ & $i,j\in\{1,2,...,n\}$, exists a local starting time $t_{ij}(0)\in[0,c_i)$, where $t_{ij}(0)=f_{ij}(0)D_{ij}+y_{ij}(0)$ (4), or a local time $t_{ji}(0)\in[0,c_j)$, where $t_{ji}(0)=f_{ji}(0)D_{ji}+y_{ji}(0)$ (5), such that

$$(r_i \le y_{ij}(0) \le D_{ij} - r_j) \vee (r_j \le y_{ji}(0) \le D_{ji} - r_i) \qquad (9)$$

When the condition (9) holds for each $i<j$ & $i,j\in\{1,2,...,n\}$ then a cycle time of the waiting-free n-process system is equal to T = l.c.m.$(c_1,...,c_i,...c_j,...,c_n)$.
Since the n-process waiting-free system has a cyclic steady-state with a period T it is enough to consider starting resource allocation times $a_i(0)$, i=1,2,...,n, such that

$$0 \le a_i(0) < T \qquad (10)$$

According to Theorem 3 in order to find all starting positions $a_i(0)$ (4) of the processes, for which waiting-free schedules exists, a constraint based finite domain combinatorial problem defined by a set of constraints (4), (5), (9) and (10) over finite sets of nonnegative integers has to be solved.

5. FINDING STARTING TIMES USING CP METHOD

Solution to the constraint based problem of finding starting times of the processes defined by relation (10) will be given using constraint logic programming (CLP) method [4], [7] implemented in a constraint programming (CP) language Oz, which is a tool of the Mozart CLP system [4].
Two basic techniques of constraint programming are constraint propagation and constraint distribution. Constraint propagation is an efficient inference mechanism designed to narrow the variable domains. It is based on a logical analysis of the constraints to derive the new constraints, which define a smaller space of the admissible solutions. For instance, in case of the following domain constraints $X<Y$ & $X\in[5,14]$ & $Y\in[1,10]$ constraint propagation can narrow the domains of X and Y to $X\in[5,9]$ and $Y\in[6,10]$. It can analyse a domain of a chosen decision variable, e.g. X, starting from the smallest values (strategy value:min; e.g. X=5) or from the biggest values (strategy value:max; e.g. X=14). The constraint propagation reduces a size of a solution search space.
Constraint distribution splits a problem into complementary cases once constraint propagation cannot advance further. Usually, a distribution strategy is defined on a sequence of variables x_1, x_2, ..., x_k used in a model of a problem. When a distribution step is necessary, the strategy selects (according to the standard strategy or user defined heuristics) a not yet determined variable in the sequence and distributes on this variable. For instance, the search space can be distributed into disjoint spaces by constraints $x_1=u$ and $x_1\neq u$, where an integer u is consistent with the set of constraints. In particular, if X=9, then the unique solution is X=9 & Y=10, and the space defined by $X\neq 9$ yet has to be analysed. By iterating propagation and distribution, propagation will eventually determine the solutions of a problem.

To develop the Oz language script solving a given problem a model and a distribution strategy have to be designed. A model of a problem is a representation of the problem as

a finite domain one. A model specifies the variables and the constraints representing the problem. The art of constraint programming consists in finding for a problem a model and a distribution strategy that yield the smallest and computationally feasible search tree [4], [7].

5.1. CP model design

The problem considered consists in finding (if exist) all starting resource allocation times $a_i(0)\in[0,T)$ (10) of the processes, where $i=1,2,\ldots,n$, for which a waiting-free schedules exist for a given n-process system.

Behaviour of the n-process system can be analysed in any time interval $B_k=[a_k(0), a_k(0)+c)$ such that $0 \le a_k(0) < a_k(0)+c < T$, where $k\in\{1,2,\ldots,n\}$ & $c\in N$. It can be noticed that for $c=cmax=\max(c_1,\ldots,c_n)$, i.e. $cmax$ is a cycle time of the slowest process, the interval B_k is the smallest one for which each process P_i receives access to the shared resource at least once. Therefore it is enough to consider starting resource allocation times $a_i(0)\in[a_k(0), a_k(0)+cmax)$. In particular, it is possible to choose $a_k(0)$ equal to a starting time of the slowest process P_k and to assume that the observation zone starts at $a_k(0)=0$. Hence, for $cmax=c_k$, domains of variables $a_i(0)$ are defined by the following constraints:

$$a_k(0)=0 \ \ \& \ \ 0 \le a_i(0) < c_k \ \ \& \ \ c_k=\max(c_1,\ldots,c_n) \tag{11}$$

In order to solve the problem considered all values of $a_i(0)$ (11) for which local starting times $t_{ij}(0)\in[0,c_i)$ (4) and $t_{ji}(0)\in[0,c_j)$ (5) fulfilling constraints (9) exist, have to be found, where $i<j$ & $i,j\in\{1,2,\ldots,n\}$. By introducing variables

$$s_{ij}=a_j(0)-a_i(0) \ \ \& \ \ a_j(0)\ge a_i(0) \tag{12}$$

$$s_{ji}=a_i(0)-a_j(0) \ \ \& \ \ a_i(0)\ge a_j(0), \tag{13}$$

which denote a distance between any starting resource allocation times of the processes, it is possible to derive new constraints integrating constraints given by (9) and (11).. From (11) $a_i(0),a_j(0)\in[0,c_k)$, hence also $s_{ij},s_{ji}\in[0,c_k)$, where $c_k=\max(c_1,\ldots,c_n)$ & $i<j$ & $i,j\in\{1,2,\ldots,n\}/\{k\}$. According to (12) $s_{kj}=a_j(0)-a_k(0)=a_j(0)$ and from (13) $s_{ki}=a_i(0)-a_k(0)=a_i(0)$. Hence, taking into account (11) $s_{ki},s_{kj}\in[0,c_k)$.The following conditions hold:

$$s_{ij}=a_j(0)-a_i(0)=[a_j(0)-a_k(0)]-[a_i(0)-a_k(0)]=s_{kj}-s_{ki} \ \ \& \ \ s_{kj}\ge s_{ki} \tag{14}$$

$$s_{ji}=a_i(0)-a_j(0)=[a_i(0)-a_k(0)]-[a_j(0)-a_k(0)]=s_{ki}-s_{kj} \ \ \& \ \ s_{ki}\ge s_{kj} \tag{15}$$

$$s_{ij},\ s_{ji},\ s_{ki},\ s_{kj}\in[0,c_k) \ \ \& \ \ c_k=\max(c_1,\ldots,c_n) \tag{16}$$

The local starting times $t_{ij}(0)\in[0,c_i)$ (4) and $t_{ji}(0)\in[0,c_j)$ (5) can be derived using the following formulas:

$$t_{ij}(0) = s_{ij} \bmod c_i \quad \& \quad t_{ji}(0) = s_{ji} \bmod c_j \tag{17}$$

Let $u_{ij}=(s_{ij} \text{ div } c_i)$, where $u_{ij}\in N\cup\{0\}$. From (1) $s_{ij}=(s_{ij} \text{ div } c_i)c_i + (s_{ij} \bmod c_i) = (u_{ij})c_i + t_{ij}(0)$. Hence, using (4) $s_{ij}=(u_{ij})D_{ij}m_{ij} + f_{ij}(0)D_{ij} + y_{ij}(0) = [u_{ij}m_{ij} + f_{ij}(0)]D_{ij} + y_{ij}(0) = v_{ij}D_{ij} + y_{ij}(0)$, where $v_{ij} = [u_{ij}m_{ij} + f_{ij}(0)]$. Finally, taking into account constraint for u_{ij} and m_{ij}, $f_{ij}(0)$ (4) there is $s_{ij} = v_{ij}D_{ij} + y_{ij}(0)$ and $v_{ij}\in N\cup\{0\}$. A domain of variable v_{ij} can be reduced. For $s_{ij}\in[0,c_k)$ (16) there is $0\leq v_{ij}D_{ij}+y_{ij}(0)<c_k$ & $y_{ij}(0)\in[0,D_{ij})$ (4). Hence, $0\leq v_{ij}<c_k/D_{ij}$. Assuming condition (9) and denoting $y_{ij}=y_{ij}(0)$ the following formulas, defining a distance between starting resource allocation time of P_j and starting resource allocation time of P_i, hold:

$$s_{ij} = v_{ij}D_{ij} + y_{ij} \;\;\&\;\; v_{ij}\in[0,\, c_k/D_{ij}) \;\;\&\;\; y_{ij}\in[r_i,\, D_{ij} - r_j] \tag{18}$$

Corresponding formulas defining distance $s_{ji}\in[0,c_k)$ (16) are given below:

$$s_{ji} = v_{ji}D_{ji} + y_{ji} \;\;\&\;\; v_{ji}\in[0,\, c_k/D_{ji}) \;\;\&\;\; y_{ji}\in[r_j,\, D_{ji} - r_i] \tag{19}$$

Using condition (18) and (19) it is possible to transform the problem considered to the following constraint programming problem.
Given is the n-process system with the operation times of the processes specified by $ZT_i=(r_i,o_i)$, where $r_i,o_i\in N$ are defined in the uniform time units and $i=1,\ldots,n$. A cycle time of a process P_i is defined as $c_i = r_i + o_i$. Let a starting time of the slowest process P_k, where $c_k=\max(c_1,\ldots,c_n)$ & $k\in\{1,\ldots,n\}$, is such that $a_k(0)=0$. Starting times $a_i(0)\in[0,c_k)$ of the processes, where $i\in\{1,\ldots,n\}/\{k\}$, are defined in relation to the time $a_k(0)=0$. Let a time shift $s_{ki}=a_i(0)-a_k(0)=a_i(0)$, where $s_{ki}\in[0,c_k)$, and a time shift for any two P_i, P_j, where $i<j$ & $i,j\in\{1,2,\ldots,n\}/\{k\}$, is defined according to (14) as $s_{ij} = s_{kj} - s_{ki}$, for $s_{kj} \geq s_{ki}$, or is defined according to (15) as $s_{ji} = s_{ki} - s_{kj}$, for $s_{ki} \geq s_{kj}$. The problem is to find, if exist, all $s_{ki}\in[0,c_k)$ where $i\in\{1,2,\ldots,n\}/\{k\}$, and all s_{ij}, $s_{ji}\in[0,c_k)$ where $i<j$ & $i,j\in\{1,2,\ldots,n\}/\{k\}$, such that constraints (18) and (19) hold.
A CP-based problem defined by a CP model of a given n-process system can be implemented using CLP language Oz and a programming system Mozart [4], [7].

5.2. Computational experiment

A solution of a problem of finding starting times of the processes for which waiting-free schedules exist will be illustrated on the example of a system with four cyclic processes sharing single resource. A standard *first fail* (*ff*) distribution strategy available in the Oz language is selected to distribute the constraints on the variables. According to this strategy variables are analysed starting from the undetermined variable for which the number of different possible values is minimal. The intervals defining constraints for the variables are searched using a strategy value:min.
Let us consider a *4*-process system (P_1, P_2, P_3, P_4) defined by the following relations: $ZT_1=(r_1,o_1)$ & $r_1=1$ & $o_1=17$ & $c_1=18$; $ZT_2=(r_2,o_2)$ & $r_2=2$ & $o_2=10$ & $c_2=12$; $ZT_3=(r_3,o_3)$

& r_3=1 & o_3=5 & c_3=6; ZT_4=(r_4,o_4) & r_4=1 & o_4=3 & c_4=4. There is: D_{12}=D_{21}= g.c.d.(c_1,c_2)=6, D_{13}=D_{31}=g.c.d.(c_1,c_3)=6, D_{14}=D_{41}=g.c.d.(c_1,c_4)=2, D_{23}=D_{32}=g.c.d.(c_2,c_3) =6, D_{24}=D_{42}=g.c.d.(c_2,c_4)=4, D_{34}=D_{43}=g.c.d.(c_3,c_4)=2. The system's cycle time T=l.c.m. (c_1,c_2,c_3,c_4)=36. Since, c_1 = max(c_1,c_2,c_3,c_4) = 18, hence process P_k, where k=1, is the slowest one. Time shifts s_{12}, s_{13}, $s_{14}\in[0,c_1)$. Time shifts s_{ij}, $s_{ji}\in[0,c_1)$, where $i<j$ & $i,j\in\{1,2,3,4\}/\{1\}$, can be calculated using s_{12}, s_{13}, s_{14} according to relations (14) and (15). Taking into account (18) and (19) domains of the variables s_{12}, s_{13}, $s_{14}\in[0,c_1)$ and s_{ij}, $s_{ji}\in[0,c_1)$ where $i<j$ & $i,j\in\{2,3,4\}$, are defined by the following constraints:

- $s_{12} = v_{12}D_{12} + y_{12}$ & $v_{12}\in[0, c_1/D_{12})$ & $y_{12}\in[r_1, D_{12} - r_2]$;
- $s_{13} = v_{13}D_{13} + y_{13}$ & $v_{13}\in[0, c_1/D_{13})$ & $y_{13}\in[r_1, D_{13} - r_3]$;
- $s_{14} = v_{14}D_{14} + y_{14}$ & $v_{14}\in[0, c_1/D_{14})$ & $y_{14}\in[r_1, D_{14} - r_4]$;
- $s_{23} = s_{13} - s_{12}$ & $s_{13} \ge s_{12}$ & $s_{23} = v_{23}D_{23} + y_{23}$ & $v_{23}\in[0, c_1/D_{23})$ & $y_{23}\in[r_2, D_{23} - r_3]$;
- $s_{32} = s_{12} - s_{13}$ & $s_{12} \ge s_{13}$ & $s_{32} = v_{32}D_{32} + y_{32}$ & $v_{32}\in[0, c_1/D_{32})$ & $y_{32}\in[r_3, D_{32} - r_2]$;
- $s_{24} = s_{14} - s_{12}$ & $s_{14} \ge s_{12}$ & $s_{24} = v_{24}D_{24} + y_{24}$ & $v_{24}\in[0, c_1/D_{24})$ & $y_{24}\in[r_2, D_{24} - r_4]$;
- $s_{42} = s_{12} - s_{14}$ & $s_{12} \ge s_{14}$ & $s_{42} = v_{42}D_{42} + y_{42}$ & $v_{42}\in[0, c_1/D_{42})$ & $y_{42}\in[r_4, D_{42} - r_2]$;
- $s_{34} = s_{14} - s_{13}$ & $s_{14} \ge s_{13}$ & $s_{34} = v_{34}D_{34} + y_{34}$ & $v_{34}\in[0, c_1/D_{34})$ & $y_{34}\in[r_3, D_{34} - r_4]$;
- $s_{43} = s_{13} - s_{14}$ & $s_{13} \ge s_{14}$ & $s_{43} = v_{43}D_{43} + y_{43}$ & $v_{43}\in[0, c_1/D_{43})$ & $y_{43}\in[r_4, D_{43} - r_3]$.

Taking into account the parameters of the *4*-process system the following relations hold:

$s_{12}, s_{13}, s_{14}, s_{23}, s_{24}, s_{34}, s_{32}, s_{42}, s_{43}\in[0, 18)$;
$v_{12}\in[0, 3), y_{12}\in[1, 4]$; $v_{13}\in[0, 3), y_{13}\in[1, 5]$; $v_{14}\in[0, 9), y_{14}\in[1, 1]$;
v_{23}, $v_{32}\in[0, 3), y_{23}\in[2, 5], y_{32}\in[1, 4]$; v_{24}, $v_{42}\in[0, 4.5), y_{24}\in[2, 3], y_{42}\in[1, 2]$; since v_{24}, $v_{42}\in N\cup\{0\}$, therefore v_{24}, $v_{42}\in[0, 4]$; v_{34}, $v_{43}\in[0, 9), y_{34}\in[1, 1], y_{43}\in[1, 1]$.
A model presented can be implemented using predefined abstractions available in the Oz language. The executable script given below can find solution vectors defined by (s_{12}, s_{13}, s_{14}, s_{ij}, s_{mn}, s_{pq}, y_{12}, y_{13}, y_{14}, y_{ij}, y_{mn}, y_{pq}), where $i\neq j$ & $i,j\in\{2,3\}$, $m\neq n$ & $m,n\in\{2,4\}$, $p\neq q$ & $p,q\in\{3,4\}$. In particular, the program generates solution vectors in case of (s_{12}, s_{13}, s_{14}, s_{23}, s_{24}, s_{34}, y_{12}, y_{13}, y_{14}, y_{23}, y_{24}, y_{34}).

```
local Find in
   proc {Find Root}
      S12 S13 S14 S23 S24 S34 Y12 Y13 Y14 Y23 Y24 Y34 V12 V13 V14
      V23 V24 V34 D12 D13 D14 D23 D24 D34
   in
      Root=sol(s12:S12 s13:S13 s14:S14 s23:S23 s24:S24 s34:S34
               y12:Y12 y13:Y13 y14:Y14 y23:Y23 y24:Y24 y34:Y34)
%domains of variables
S12::0#17 S13::0#17 S14::0#17 S23::0#17 S24::0#17 S34::0#17
D12::6#6  D13::6#6  D14::2#2  D23::6#6  D24::4#4  D34::2#2
V12::0#2  V13::0#2  V14::0#8  V23::0#2  V24::0#4  V34::0#8
Y12::1#4  Y13::1#5  Y14::1#1
Y23::2#5  %for Y32 change into 1#4
Y24::2#3  %for Y42 change into 1#2
Y34::1#1  %for Y43 the same relation holds 1#1
%constraints for variables S23, S24, S34
S23=:S13-S12 S13>=:S12 S24=:S14-S12
S14>=:S12 S34=:S14-S13 S14>=:S13
S12=:V12*D12+Y12 S13=:V13*D13+Y13 S14=:V14*D14+Y14
```

```
S23=:V23*D23+Y23 S24=:V24*D24+Y24 S34=:V34*D34+Y34
%start propagation and distribution
{FD.distribute ff Root}
 end
   {Browse {SearchAll Find}}   %find all solutions
end
```

In the case considered a total number of solutions is equal to 27. Solutions with the same values of $(y_{12}, y_{13}, y_{14}, y_{23}, y_{24}, y_{34})$ define a four subsets of starting times, which belong to the same waiting-free schedules. The examples of the solution vectors for each subset are as follows:

- sol(s_{12}:1, s_{13}:4, s_{14}:7, s_{23}:3, s_{24}:6, s_{34}:3, y_{12}:1, y_{13}:4, y_{14}:1, y_{23}:3, y_{24}:2, y_{34}:1); a starting time of a waiting-free schedule of type 1; the schedule is shown in Fig.3;
- sol(s_{12}:3, s_{13}:8, s_{14}:9, s_{23}:5, s_{24}:6, s_{34}:1, y_{12}:3, y_{13}:2, y_{14}:1, y_{23}:5, y_{24}:2, y_{34}:1); a starting time of a waiting-free schedule of type 2;
- sol(s_{12}:2, s_{13}:4, s_{14}:5, s_{23}:2, s_{24}:3, s_{34}:1, y_{12}:2, y_{13}:4, y_{14}:1, y_{23}:2, y_{24}:3, y_{34}:1); a starting time of a waiting-free schedule of type 3;
- sol(s_{12}:4, s_{13}:8, s_{14}:11, s_{23}:4, s_{24}:7, s_{34}:3, y_{12}:4, y_{13}:2, y_{14}:1, y_{23}:4, y_{24}:3, y_{34}:1); a starting time of a waiting-free schedule of type 4.

In the other cases, i.e. (s_{23}, s_{24}, s_{43}), (s_{23}, s_{42}, s_{34}), (s_{23}, s_{42}, s_{43}), (s_{32}, s_{24}, s_{34}), (s_{32}, s_{24}, s_{43}), (s_{32}, s_{42}, s_{34}), (s_{32}, s_{42}, s_{43}), a number of solution vectors defined is the same as in the first case. All derived starting times belong to the four different waiting-free schedules the same as previously defined type1, type 2, type 3 and type 4.

```
      0                                     36
      |                                      |
P1    |ROOOOOOOOOOOOOOOOOROOOOOOOOOOOOOOOOO|ROOOOOOO...
      |                                      |
P2    |ORROOOOOOOOOORROOOOOOOOOORROOOOOOOOO|ORROOOOO...
      |                                      |
P3    |OOOOROOOOOROOOOOROOOOOROOOOOROOOOORO|OOOOROOO...
      |                                      |
P4    |OOOROOOROOOROOOROOOROOOROOOROOOROOOR|OOOROOOR...
      |               T = 36                 |
```

Fig.3. A waiting-free schedule of type 1. A letter R *denotes a time unit of using shared resource and a letter* O *– a time unit of using non-shared resource.*

6. CONCLUSIONS

A problem of finding waiting-free schedules for repetitive manufacturing processes using common resources in mutual exclusion has been considered. In many cases systems of processes can be seen as composed of subsystems with *n* processes sharing single resource. Using necessary and sufficient conditions for waiting-free execution of the *n*-process system and modulus arithmetic properties a CP-based method for finding starting times of the processes for which waiting-free schedules exist has been presented. For a given operation times of the processes the method allows derive, if exist, all possible initial resource allocation times for which a waiting-free execution is possible. The method has been implemented using constraint logic programming language Oz and tested in case of a given *4*-process system. An analysis of solution vectors allows answer to a question what is a number of different waiting-free schedules for a given *n*-process system. The CP-method designed can also be used to automate searching of the operations times for which exist starting times of the processes belonging to a waiting-free schedule. The extension of the method requires a procedure for calculating the greatest common divisor of two integers representing cycle times of processes P_i, P_j, where $i,j \in \{1,2,...,n\}$ & $i \neq j$. In the example presented the Oz script designed uses the greatest common divisors as a given data.

The *n*-process system is a deadlock-free system. Further research can be focused on a problem of finding waiting-free schedules for systems of cyclic processes with deadlock possibility.

REFERENCES

[1] Alpan G., Jafari M. A.: *Dynamic Analysis of Timed Petri Nets: a Case of Two Processes and a Shared Resource.* IEEE Trans. on Robotics and Automation, Vol.13, No. 3, 1997, pp. 338-346.

[2] Alpan G., Jafari M. A.: *Synthesis of Sequential Controller in the Presence of Conflicts and Free Choices.* IEEE Trans. on Robotics and Automation, Vol.14, No. 3, 1998, pp. 488-492.

[3] Banaszak Z., Krogh B.: *Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows.* IEEE Trans. on Robotics and Automation, 1990, Vol. 6, No. 6, pp. 724-734.

[4] Benhamou F.: *Interval constraint logic programming. In Andreas Podelski, editor, Constraints: Basics and Trends.* Lecture Notes in Computer Science, vol. 910, pp. 1-21, Springer-Verlag, Berlin, 1995.

[5] Cohen H.: *A Course in Computational Algebraic Number Theory.* Springer-Verlag, Berlin, Heidelberg, 1993.

[6] Polak M., Majdzik P., Banaszak Z., Wójcik R.: *The Performance Evaluation Tool for Automated Prototyping of Concurrent Cyclic Processes.* Fundamenta Informaticae, Vol. 60, No.1-4, April 2004, pp. 269-289.

[7] Saraswat V.: *Concurrent Constraint Programming.* MIT Press, 1994.

[8] Schroder M. R.: *Number Theory in Science and Communications.* 3rd edition, Springer-Verlag, Berlin, Heidelberg, 1997.

[9] Wójcik R.: *Metody dynamicznej alokacji zasobów w zadaniach sterowania ESP.* Zeszyty Naukowe Politechniki Śląskiej, seria Automatyka, z.109, Gliwice, 1992, pp. 321-332.

[10] Wójcik R.: *Performance evaluation of repetitive manufacturing processes using state vectors approach.* Computer Integrated Manufacturing. Advanced Design and Management (eds. Skołud B., Krenczyk D.). WNT, Warszawa, 2003, pp. 612-619.

[11] Wójcik R.: *Towards Strong Stability of Concurrent Repetitive Processes Sharing Resources.* Systems Science, Vol. 27, No. 2, 2001, pp. 37-47.