

CONSTRAINT PROGRAMMING APPROACH TO DESIGNING CONFLICT-FREE SCHEDULES FOR REPETITIVE MANUFACTURING PROCESSES

Robert Wójcik

*The Institute of Computer Engineering, Control and Robotics
Wrocław University of Technology, Poland,
robert.wojcik@pwr.wroc.pl*

A problem of designing conflict-free schedules for concurrent repetitive manufacturing processes using resources in mutual exclusion is considered. Processes that share resources can wait for resources (machines) required to complete successive operations in their production routes. Systems of processes sharing only one resource are considered. A problem of finding operation times and starting times of the processes for which a schedule exists, with no process waiting for access to the shared resource has been solved using the constraint programming (CP) approach. The necessary and sufficient conditions for existence of conflict-free schedules have been implemented in the Oz/Mozart constraint programming system used to find the schedules.

1. INTRODUCTION

Planning of production flow in modern enterprises often requires solving a problem of finding a schedule defining an order of the operations executed by concurrent repetitive manufacturing processes using common resources in mutual exclusion, which in turn requires solving a problem of resource conflicts resolution (Alpan, 1997). A solution of this problem is the best schedule taking into account some constraints imposed on manufacturing operations as well as other objectives, e.g. producing parts according to just-in-time philosophy (Kanban, 1989), ensuring deadlock-free execution of processes (Banaszak, 1990), or robustness of processes (Wójcik, 2001).

In the repetitive manufacturing processes the same components are produced over and over again according to given production routes, defining a sequence of operations required for completion of the final product (Kanban, 1989), (Wójcik, 2004). Each operation in the route is using one resource (machine, buffer) for a certain amount of time defined by the operation time. Assuming that the next component may be started when the previous one has been produced a repetitive manufacturing 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 orders are

processed at the same time the production system can be seen as a system of concurrent, repetitive manufacturing processes sharing resources in mutual exclusion (Alpan, 1997), (Wójcik, 2001).

The increased requirements concerning the time necessary to design a production plan implies a need to apply methods and tools which can be used for rapid prototyping of alternative ways of manufacturing processes execution (Banaszak, 1990), (Alpan, 1998). The method presented in this paper is based on the constraint programming (CP) (Saraswat, 1994). This approach allows finding of a feasible manufacturing schedule, which satisfies constraints imposed on the processes execution, e.g. constraints ensuring that no process will ever wait for resource allocation (Wójcik, 2001). It can be seen as an attractive alternative for methods based on computer simulation or operations research.

2. PROBLEM FORMULATION

Consider a system composed of n repetitive manufacturing processes, which are sharing a single resource, i.e. the n -process. For given domains of the operation times, assuming the processes are being repeated forever (a number of parts produced according to a production route can be infinite), the problem considered consists in finding the operation times and the starting times of the processes for which a feasible schedule exists with no process waiting for access to the shared resource. The conflict-free schedule must fulfill all constraints imposed on the processes' execution by the precedence relations and by the resources availability.

3. DEFINING SYSTEM OF REPETITIVE PROCESSES

In a manufacturing system shown in Fig.1 three types of parts P_1 , P_2 , P_3 are produced using machines R , O_1 , O_2 , O_3 . Assuming the infinite number of parts for each type, the production of the parts creates three cyclic manufacturing processes, repeated forever, and sharing resource R in mutual exclusion, i.e. the 3-process.

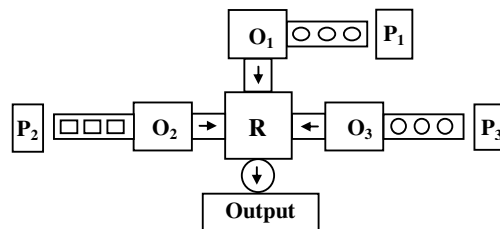


Figure 1 – A manufacturing system producing three types of parts P_1 , P_2 , P_3

The n -process system $(P_1, \dots, P_i, \dots, P_j, \dots, P_n)$ consists of n repetitive manufacturing processes sharing a single resource R , e.g. the 3-process system (P_1, P_2, P_3) . Each process P_i executes periodically a sequence of operations using resources defined by $Z_i = (R, O_i)$, where R denotes a shared resource used by the processes, O_i represents the non-shared resources used by process P_i , and $i=1, 2, \dots, n$. The operation times ZT_i

$= (r_i, o_i)$, corresponding to the route Z_i , belong to certain domains $r_i \in DR_i$, $DR_i = \{r_{i1}, r_{i2}, \dots, r_{ij}, \dots, r_{iw}\}$, and $o_i \in DO_i$, $DO_i = \{o_{i1}, o_{i2}, \dots, o_{ij}, \dots, o_{iw}\}$, where $r_{ij}, o_{ij} \in N$. A cycle time of P_i is defined by relation $c_i = r_i + o_i$.

It has been shown (Wójcik, 2001) that the behaviour of the n -process system depends on the operation times and starting times (phases) of the component 2-process subsystems. In the following conditions for existence of a waiting-free schedule for n -process system will be given (Wójcik, 2004).

4. THE ALGEBRAIC MODEL OF THE SYSTEM

A natural model describing dynamics of repetitive manufacturing processes sharing a resource is modulus algebra (Schroder, 1997), (Wójcik, 2001). To avoid resource conflicts it is enough to exclude situations when a process is requesting shared resource while other process is using it. Taking into account the properties of the modulus algebra it is possible to design recurrent modulus equations that define times of any process resource request in relation to a chosen process request and to find conditions ensuring waiting-free execution of the n -process system.

4.1 Local Starting Times

Consider a system of cyclic processes $(P_1, \dots, P_i, \dots, P_j, \dots, P_n)$ sharing a single resource (Fig.1). Let $x_i(k) \in N \cup \{0\}$ for $k=0, 1, 2, \dots$, denote times at which a process P_i , where $i \in \{1, 2, \dots, n\}$, requests access to the shared resource R and $a_i(k) \in N \cup \{0\}$ times at which it receives access to the resource (Fig.2). 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 given and equal to $x_i(0) = 0$. A starting time of any other process P_j , where $j \neq i$ is such that $x_j(0) \geq 0$. The times $x_i(k)$, $a_i(k)$, $i = 1, 2, \dots, n$, and $k=0, 1, 2, \dots$, define a schedule of the processes.

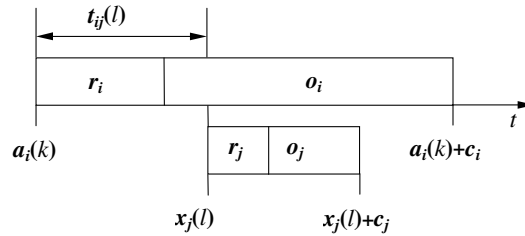


Figure 2 – Local resource requesting time $t_{ij}(l)$ for processes (P_i, P_j)

Assuming that only one process P_j is executed in the system (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 the case of concurrent execution of the processes the relevant formula has to take into account a waiting time $w_i(k)$ of the process P_i , i.e. $a_i(k+1) = x_i(k+1) + w_i(k) = a_i(k) + c_i + w_i(k)$. A parameter $t_{ij}(l) \in N$ & $t_{ij}(l) \in [0, c_i)$, $l=0, 1, 2, \dots$ (Fig.2), defines distance between a resource request time $x_j(l)$ of process P_j and the nearest resource allocation time $a_i(k) \leq x_j(l)$ of process P_i .

The shift $t_{ij}(l)$ can be used as a local starting time of P_j in relation to the nearest, previous resource allocation time of P_i .

For any 2-process subsystem (P_i, P_j) of the n -process system, where $i \neq j$ & $i, j \in \{1, 2, \dots, n\}$, it is possible to derive (Wójcik, 2004) values $t_{ij}(l)$, $l=0, 1, 2, \dots$, using recurrent modulus equations. It can be proven that in the case of no resource conflicts, resource request times of process P_j , calculated in relation to resource allocation times of process P_i , can occur only at times $t_{ij}(l) \in [0, c_i)$ given by

$$t_{ij}(l) = x_j(l) \bmod c_i = a_j(l) \bmod c_i = f_{ij}(l)D_{ij} + y_{ij}(l) \quad \& \quad l=0, 1, 2, \dots \quad (1)$$

$a_j(0)$, $j \in \{1, 2, \dots, n\}$ – starting times of the processes,
 $D_{ij} = D_{ji} = g.c.d.(c_i, c_j)$ & $c_i = D_{ij}m_{ij}$ & $c_j = D_{ji}m_{ji}$ & $g.c.d.(m_{ij}, m_{ji}) = 1$ & $m_{ij}, m_{ji} \in N$ &
 & $f_{ij}(l) = [f_{ij}(0) + lm_{ij}] \bmod m_{ij}$ & $y_{ij}(l) = y_{ij}(0)$ &
 & $t_{ij}(0) = a_j(0) \bmod c_i$ & $f_{ij}(0) = t_{ij}(0) \operatorname{div} D_{ij}$ &
 & $y_{ij}(0) = t_{ij}(0) \bmod D_{ij}$ & $0 \leq f_{ij}(0) < m_{ij}$ & $0 \leq y_{ij}(0) < D_{ij}$
 & $g.c.d.$ - the greatest common divisor.

By the symmetry of (P_i, P_j) and (P_j, P_i) a starting time $t_{ji}(l) \in [0, c_j)$ can be defined.

4.2 The Conditions for the Conflict-Free Processes Execution

To avoid resource conflicts between the processes it is enough to avoid requesting of the shared resource by P_j at time intervals $[a_i(k), a_i(k) + r_i]$, and analogously, requesting it by P_i at time intervals $[a_j(l), a_j(l) + r_j]$. The necessary and sufficient condition for the existence of a conflict-free schedule for n -process system has been given in (Wójcik, 2001), (Wójcik, 2004).

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

$$(r_i \leq y_{ij}(0) \leq D_{ij} - r_j) \vee (r_j \leq y_{ji}(0) \leq D_{ji} - r_i) \quad (2)$$

When the condition (2) holds for each $i < j$ & $i, j \in \{1, 2, \dots, n\}$ then a cycle time of a conflict-free n -process system is equal to $T = l.c.m.(c_1, \dots, c_i, \dots, c_j, \dots, c_n)$; $l.c.m.$ - the least common multiple.

Since the conflict-free (waiting-free) n -process system has a cyclic, steady-state with a period $T = l.c.m.(c_1, \dots, c_i, \dots, c_j, \dots, c_n)$ it is enough to consider starting times $a_i(0)$, $i=1, 2, \dots, n$, of the processes such that

$$0 \leq a_i(0) < T \quad \& \quad T = l.c.m.(c_1, \dots, c_i, \dots, c_j, \dots, c_n) \quad (3)$$

5. THE CONFLICT-FREE SCHEDULES DESIGN

The problem of designing a conflict-free schedule for the considered n -process system can be formulated as follows: Given are domains of the operation times

$DR_i = \{r_{i1}, r_{i2}, \dots, r_{ij}, \dots, r_{iw}\}$ – admissible values for operation time r_i , and $DO_i = \{o_{i1}, o_{i2}, \dots, o_{ij}, \dots, o_{iv}\}$ – admissible values for operation time o_i , $i=1,2,\dots,n$. Find values of the operation times $r_i \in DR_i$, $o_i \in DO_i$, and starting times $a_i(0)$, $i=1,2,\dots,n$, of the processes for which a conflict-free steady-state schedule exists. Assuming $r_i \in DR_i$, $o_i \in DO_i$, and $c_i = r_i + o_i$ the considered constraint satisfaction problem (CSP) consists of finding times $a_i(0)$ (3) for which constraints (1) and (2) hold.

The problem will be solved using CP method and CP Oz/Mozart tool (Saraswat, 1994). In this approach the model defining variables and constraints of the problem can be implemented in a declarative way (without implementing an algorithm for solving the problem), using predefined abstractions of the CP language Oz. This creates an attractive alternative for currently available, decision support systems used for manufacturing processes design, based on computer simulation or conventional operations research techniques.

5.1 The Constraint Programming Method

Two basic techniques of constraint programming are constraint propagation and constraint distribution (Saraswat, 1994).

The 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. The constraint propagation reduces size of a search space.

The constraint distribution splits a problem into complementary cases once the constraint propagation cannot advance further. Usually, a distribution strategy is defined with a sequence of variables x_1, x_2, \dots, 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, i.e. the search space can be distributed into disjoint spaces by substitution $x_i = u$ and $x_i \neq u$, where an integer u belongs to variable's x_i domain. The art of constraint programming consists in designing a model for a problem and a distribution strategy that yield the smallest search tree.

5.2 The Constraint-Based Model

Let us consider starting times $a_i(0) \in [0, T)$ (3) of the processes, $i=1,2,\dots,n$. Behaviour of the n -process system can be analysed in any time interval $B_k = [a_k(0), a_k(0) + c)$ such that $0 \leq a_k(0) < a_k(0) + c < T$, and $k \in \{1,2,\dots,n\}$, $c \in N$. It can be noticed that for $c = c_{max} = \max(c_1, \dots, c_n)$, i.e. c_{max} is a cycle time of the slowest process, the interval B_k is the smallest one for which each process P_i in a waiting-free system receives access to the shared resource at least once. Therefore, it is enough to consider starting resource allocation times $a_i(0) \in [a_k(0), a_k(0) + c_{max})$. 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 $c_{max} = c_k$, domains of variables $a_i(0)$ are defined by the following constraints:

$$a_k(0) = 0 \ \& \ 0 \leq a_i(0) < c_k \ \& \ c_k = \max(c_1, \dots, c_n) \quad (4)$$

In order to solve the problem considered, all values $a_i(0)$ (4) for which local starting times $t_{ij}(0) \in [0, c_i]$ (1) and $t_{ji}(0) \in [0, c_j]$ (1) fulfilling constraints (2) exist, have to be found, where $i < j$ & $i, j \in \{1, 2, \dots, n\}$. By introducing variables s_{ij} , s_{ji} , which denote a distance between any two starting times of the processes, it is possible to derive the new constraints integrating constraints given by (2) and (4).

$$\begin{aligned} s_{ij} &= a_j(0) - a_i(0) \ \& \ a_j(0) \geq a_i(0) \\ s_{ji} &= a_i(0) - a_j(0) \ \& \ a_i(0) \geq a_j(0) \end{aligned} \quad (5)$$

From (4) $a_i(0), a_j(0) \in [0, c_k]$, hence also $s_{ij}, s_{ji} \in [0, c_k]$, where $c_k = \max(c_1, \dots, c_n)$ & $i < j$ & $i, j \in \{1, 2, \dots, n\} / \{k\}$. According to (5) $s_{kj} = a_j(0) - a_k(0) = a_j(0)$ and $s_{ki} = a_i(0) - a_k(0) = a_i(0)$. Hence, taking into account (5) $s_{ki}, s_{kj} \in [0, c_k]$. The following conditions hold:

$$\begin{aligned} s_{ij} &= s_{kj} - s_{ki} \ \& \ s_{kj} \geq s_{ki} \ \& \ s_{ji} = s_{ki} - s_{kj} \ \& \ s_{ki} \geq s_{kj} \\ s_{ij}, s_{ji}, s_{ki}, s_{kj} &\in [0, c_k] \ \& \ c_k = \max(c_1, \dots, c_n) \end{aligned} \quad (6)$$

Local starting times $t_{ij}(0) \in [0, c_i]$ (1) and $t_{ji}(0) \in [0, c_j]$ (1) can be derived using formulas $t_{ij}(0) = s_{ij} \bmod c_i$, and $t_{ji}(0) = s_{ji} \bmod c_j$.

Let $u_{ij} = (s_{ij} \text{ div } c_i)$, $u_{ij} \in N \cup \{0\}$. There is $a_{ij} = (s_{ij} \text{ div } c_i)c_i + (s_{ij} \bmod c_i) = (u_{ij})c_i + t_{ij}(0)$. Hence, using (1) $s_{ij} = (u_{ij})D_{ij}m_{ij} + f_{ij}(0)D_{ij} + y_{ij}(0) = [u_{ij}m_{ij} + f_{ij}(0)]D_{ij} + y_{ij}(0) = v_{ij}D_{ij} + y_{ij}(0)$, where $v_{ij} = [u_{ij}m_{ij} + f_{ij}(0)]$. Finally, taking into account constraint for u_{ij} and m_{ij} , $f_{ij}(0)$ (1) there are $s_{ij} = v_{ij}D_{ij} + y_{ij}(0)$ and $v_{ij} \in N \cup \{0\}$. A domain of variable v_{ij} can be reduced. For $s_{ij} \in [0, c_k]$ (6) there is $0 \leq v_{ij}D_{ij} + y_{ij}(0) < c_k$ & $y_{ij}(0) \in [0, D_{ij}]$ (1), hence, $0 \leq v_{ij} < c_k/D_{ij}$. Taking into account (2) and denoting $y_{ij} = y_{ij}(0)$ the following formulas hold, based on necessary and sufficient conditions, defining a distance between starting time of the process P_j and starting time of the process P_i in a conflict-free schedule:

$$s_{ij} = v_{ij}D_{ij} + y_{ij} \ \& \ v_{ij} \in [0, c_k/D_{ij}] \ \& \ y_{ij} \in [r_i, D_{ij} - r_j] \quad (7)$$

Formula defining $s_{ji} \in [0, c_k]$ can be given analogously, by changing i, j .

The model presented defines the following CP problem: Let P_k be the slowest cyclic process. Find $s_{ki} \in [0, c_k]$ for $i \in \{1, 2, \dots, n\} / \{k\}$, and s_{ij} , $s_{ji} \in [0, c_k]$ for $i < j$ & $i, j \in \{1, 2, \dots, n\} / \{k\}$, such that constraints (6) and (7) hold. This problem will be solved using CP programming tool Mozart (Saraswat, 1994).

5.3 Computational Example

For a 3-process system shown in Fig.1, all possible conflict-free schedules will be derived using the constraint-based model presented. A standard *first fail (ff)* distribution strategy available in the Oz language is selected to distribute the constraints on the variables (Saraswat, 1994).

Let us consider a system $S_3 = (P_1, P_2, P_3)$ with the following parameters: $r_1 \in [1, 1]$, $o_1 \in [16, 17]$, $ZT_1 = (r_1, o_1)$; $r_2 \in [1, 1]$, $o_2 \in [10, 11]$, $ZT_2 = (r_2, o_2)$; $r_3 \in [4, 4]$, $o_3 \in [2, 3]$, $ZT_3 = (r_3, o_3)$. Analysis of all possible values from the domains of the operation times leads to the following eight cases defined by vectors $(r_1, o_1, c_1; r_2, o_2, c_2; r_3, o_3, c_3)$: (1,16,17; 1,10,11; 4,2,6), (1,16,17; 1,10,11; 4,3,7), (1,16,17; 1,11,12; 4,2,6), (1,16,17; 1,11,12; 4,3,7), and (1,17,18; 1,10,11; 4,2,6), (1,17,18; 1,10,11; 4,3,7), (1,17,18; 1,11,12; 4,2,6), (1,17,18; 1,11,12; 4,3,7).

In all cases P_1 is the slowest process, i.e. $c_k = \max(c_1, c_2, c_3) = c_1$, and $k=1$. Hence, starting times s_{12} , s_{13} (6) of the processes P_2 , P_3 are defined in relation to time

$a_1(0)=0$, i.e. $s_{12}, s_{13} \in [0, c_1]$. Time shifts $s_{23}, s_{32} \in [0, c_1]$ can be calculated using s_{12}, s_{13} defined by (6). According to (7), the following constraints hold for the variables $s_{12}, s_{13} \in [0, c_1]$, and $s_{23}, s_{32} \in [0, c_1]$:

$$\begin{aligned} s_{12} &= v_{12}D_{12} + y_{12} \ \& \ v_{12} \in [0, c_1/D_{12}] \ \& \ y_{12} \in [r_1, D_{12} - r_2]; \\ s_{13} &= v_{13}D_{13} + y_{13} \ \& \ v_{13} \in [0, c_1/D_{13}] \ \& \ y_{13} \in [r_1, D_{13} - r_3]; \\ s_{23} &= s_{13} - s_{12} \ \& \ s_{13} \geq s_{12} \ \& \ s_{23} = v_{23}D_{23} + y_{23} \ \& \ v_{23} \in [0, c_1/D_{23}] \ \& \ y_{23} \in [r_2, D_{23} - r_3]; \\ s_{32} &= s_{12} - s_{13} \ \& \ s_{12} \geq s_{13} \ \& \ s_{32} = v_{32}D_{32} + y_{32} \ \& \ v_{32} \in [0, c_1/D_{32}] \ \& \ y_{32} \in [r_3, D_{32} - r_2]. \end{aligned} \quad (8)$$

These constraints define the CP problem with the following solution vectors: $(s_{12}, s_{13}, s_{23}, y_{12}, y_{13}, y_{23})$ and $(s_{12}, s_{13}, s_{32}, y_{12}, y_{13}, y_{32})$. Assuming values for $(r_1, o_1, c_1; r_2, o_2, c_2; r_3, o_3, c_3)$ it is possible to solve the problem using predefined abstractions available in the Oz language – a programming tool of the Mozart system.

It can be shown that solutions of the problem exist only for case (1,17,18; 1,11,12; 4,2,6). In this case: $D_{12}=D_{21}=g.c.d.(c_1, c_2)=6$, and $D_{13}=D_{31}=g.c.d.(c_1, c_3)=6$, and $D_{23}=D_{32}=g.c.d.(c_2, c_3)=6$. Since, $c_1 = \max(c_1, c_2, c_3) = 18$, hence process P_k , where $k=1$, is the slowest one. According to (7), (8) the following constraints hold: $s_{12}, s_{13}, s_{23}, s_{32} \in [0, 18]$; $v_{12} \in [0, 3]$, $y_{12} \in [1, 5]$; $v_{13} \in [0, 3]$, $y_{13} \in [1, 2]$; $v_{23}, v_{32} \in [0, 3]$, $y_{23} \in [1, 2]$, $y_{32} \in [4, 5]$. The executable script given below can find solution vectors defined by $(s_{12}, s_{13}, s_{ij}, y_{12}, y_{13}, y_{ij})$, where $i \neq j$ & $i, j \in \{2, 3\}$. The following program generates solutions for $(s_{12}, s_{13}, s_{32}, y_{12}, y_{13}, y_{32})$.

```

local Find in
  proc {Find Root}
    S12 S13 S32 Y12 Y13 Y32 V12 V13 V32 D12 D13 D32 in
    Root=sol(s12:S12 s13:S13 s32:S32 y12:Y12 y13:Y13 y32:Y32)
    %variable domains
    S12::0#17 S13::0#17 S32::0#17 D12::6#6 D13::6#6 D32::6#6
    V12::0#2 V13::0#2 V32::0#2
    Y12::1#5 Y13::1#2 Y32::4#5 %for Y23 change into 1#2
    %constraints for the variables
    S32=:S12-S13 S12>=:S13
    S12=:V12*D12+Y12 S13=:V13*D13+Y13 S32=:V32*D32+Y32
    %start propagation and distribution
    {FD.distribute ff Root}
  end
  {Browse {SearchAll Find}} %find all solutions
end

```

In the case considered 9 solutions have been found. Solutions with the same values of (y_{12}, y_{13}, y_{32}) define starting times, which belong to the same conflict-free schedule. Two different schedules exist for the 3-process system considered. Starting times of the processes corresponding to a schedule shown in Fig.3:

sol(s_{12} :**5** s_{13} :**1** s_{32} :**4** y_{12} :**5** y_{13} :**1** y_{32} :**4**); sol(s_{12} :**11** s_{13} :**1** s_{32} :**10** y_{12} :**5** y_{13} :**1** y_{32} :**4**);
 sol(s_{12} :**11** s_{13} :**7** s_{32} :**4** y_{12} :**5** y_{13} :**1** y_{32} :**4**); sol(s_{12} :**17** s_{13} :**1** s_{32} :**16** y_{12} :**5** y_{13} :**1** y_{32} :**4**);
 sol(s_{12} :**17** s_{13} :**7** s_{32} :**10** y_{12} :**5** y_{13} :**1** y_{32} :**4**); sol(s_{12} :**17** s_{13} :**13** s_{32} :**4** y_{12} :**5** y_{13} :**1** y_{23} :**4**).

In the case $(s_{12}, s_{13}, s_{23}, y_{12}, y_{13}, y_{23})$ a number of solutions is the same. All derived vectors define states belonging to the waiting-free schedules, which have been found for the case $(s_{12}, s_{13}, s_{32}, y_{12}, y_{13}, y_{32})$.

