# A deadlock avoidance approach for Non–Sequential Resource Allocation Systems

J. Ezpeleta, L. Recalde

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
María de Luna 3, 50018 Zaragoza (SPAIN)
{ezpeleta,lrecalde}@posta.unizar.es

*Abstract*— **Many solutions have been proposed for deadlock related problems in systems where a set of sequential processes must be concurrently executed competing for the set of system resources (these systems are usually named as Sequential Resource Allocation Systems -S–RAS-). In the case in which the involved processes have a non–sequential nature (Non–Sequential Resource Allocation Systems -NS–RAS-) the problem becomes more complex. In this paper we propose a deadlock avoidance algorithm for this last class of systems. The solution is obtained by means of an adaptation of a Banker's approach for S–RAS. We also show the usefulness of the proposed solution by means of its application to a real system.**

**Keywords:** Deadlock avoidance, Banker's algorithm, Concurrent systems, Assembly–disassembly Systems

## I. Introduction

The point of view adopted in this paper looks at a manufacturing system as a resource allocation system (RAS) [Rev98], using Petri nets as formal tool to model and control the system.

In the adopted perspective, each in–process material (either a complete part, a component to be assembled or a component coming from a disassembly operation) will be modelled as a token that moves through the Petri net. A production order corresponds to the processing of a part. When being executed, such production order will correspond to an active process. A process is defined by the set of states it can reach, modelling the different operations to be carried out over the components as well as the actions executed to change such state. During the production process, each component of a final product needs to use some system resources (a machine and/or set of tools that are needed for the current operation on a component, or the buffer capacity needed for an intermediate storing operation, or the robot that is moving it between two locations, etc.). Each state change of a component can be seen as an operation by which a set of new resources is engaged and a set of resources is released.

As in any concurrent system where processes share common resources deadlocks can appear during the production process. In most cases deadlock states must be imperatively avoided. This paper concentrates on the development of a deadlock avoidance method applicable to systems with the following characteristics: 1) production orders are allowed to have assembly/disassembly operations (which gives the non-sequential nature in the production of a part); 2) the use of system resources must be conservative (resources are neither created nor destroyed); 3) actions related to the use of resources are controllable; 4) flexible part routing is allowed.

A lot of work has been done related to the control of such systems when no assembly/disassembly operation is allowed (see, for instance, [VNJ90], [BK90], [LRF97], [RR92], [ECM95], [CX97], [Jen96], [ETGC] for different approaches). The case of assembly/disassembly systems, from a Petri net perspective, has been dealt with in [RW92], [XJ99].

In general, it is assumed that the isolated execution of a production order must be possible in the system (on the contrary, the production plan is not properly defined). In the case of systems with no assembly/disassembly operation, to check if this is possible is easy to do. In the case of assembly/disassembly systems the problem has been proved to be NP–complete [RW92]. Assuming a correct system (correct here means that each production order can be executed in isolation), a second problem is to know if the processing of all the active orders can be terminated. The optimal solution (characterising exactly the set of system states from which all the active processes can terminate) is NP–hard [RW92]. A deadlock prevention/avoidance policy has to forbid reachable system states for which it is not possible to ensure that the active processes can terminate (or, equivalently, those states from which the initial state cannot be reached). In the general case, non–optimal solutions based on sufficient conditions can be provided. Moreover, when a deadlock avoidance approach is adopted (which means that the decision of allowing or forbidding a state transition must be taken on–line) efficient algorithms for checking a sufficient condition are necessary.

In this paper we adopt a mixed approach based on the computation of the reachability graph corresponding to the isolated execution of each type of production order (avoiding the computation of the reachability graph of the whole model). We show how the reachability graphs of the different production orders can be composed obtaining a Petri net belonging to the class of the $S^*PR$ nets, for which a polynomial time complexity adaptation of the Banker's algorithm for deadlock avoidance can be applied [ETGC]. We illustrate the interest of the proposed control method by its application to a real as-

sembly system.

The paper is organised as follows. In Section II a deadlock avoidance control policy for S–RAS, based on the Banker's algorithm, is presented. The classes of non–sequential resource allocation process (NS–RAP) and non–sequential resource allocation system (NS–RAS) are introduced in Section III. Section IV shows how the previous deadlock avoidance algorithm can be used in the control of NS–RAS. The idea is further illustrated in Section V, where it is applied to a manufacturing system.

## II. A deadlock avoidance control policy for Sequential Resource Allocation Systems

Let us introduce a class of Petri nets that allows to model, in a natural way, many S–RAS with serially reusable resources, $S^*PR$. These nets can model both flexible routing and the use of several resources at each processing step [ETGC].

*Definition 1:* Let $I_{\mathcal{N}} = \{1, 2, \ldots, m\}$ be a finite set of indices. An $S^*PR$ net is a connected self–loop free generalised Petri net $\mathcal{N} = \langle P, T, C \rangle$ where:

(1) $P = P_0 \cup P_S \cup P_R$ is a partition such that: (a) $P_S = \bigcup_{i \in I_{\mathcal{N}}} P_{S_i}$, $P_{S_i} \neq \emptyset$ and $P_{S_i} \cap P_{S_j} = \emptyset$, for all $i \neq j$. (b) $P_0 = \bigcup_{i \in I_{\mathcal{N}}} \{p_{0_i}\}$. (c) $P_R = \{r_1, r_2, \ldots, r_n\}$, $n > 0$. (2) $T = \bigcup_{i \in I_{\mathcal{N}}} T_i$, $T_i \neq \emptyset$, $T_i \cap T_j = \emptyset$, for all $i \neq j$ (3) For all $i \in I_{\mathcal{N}}$, the subnet $\mathcal{N}_i$ generated by $P_{S_i} \cup \{p_{0_i}\} \cup T_i$ is a strongly connected state machine. (4) For each $r \in P_R$ there exists a minimal P–Semiflow, $Y_r \in \mathbb{N}^{|P|}$, such that $\{r\} = \|Y_r\| \cap P_R$, $Y_r[r] = 1$, $P_0 \cap \|Y_r\| = \emptyset$, and $P_S \cap \|Y_r\| \neq \emptyset$. (5) $P_S = \bigcup_{r \in P_R} (\|Y_r\| \setminus \{r\})$.

An initial marking is *acceptable* for $\mathcal{N}$ iff (1) $\forall\, i \in I_{\mathcal{N}}$, $\mathbf{m_0}[p_{0_i}] > 0$. (2) $\forall p \in P_S$, $\mathbf{m_0}[p] = 0$. (3) $\forall r \in P_R$, $\mathbf{m_0}[r] \geq \max_{p \in \|Y_r\| \setminus \{r\}} Y_r[p]$.

Places of $P_S$ are called *process state places* (or *process places*). At a reachable marking, a token in a place $p \in P_{S_i}$ models an in–process part whose processing state is modelled by means of place $p$. Each place $p_{0_i}$ is called *idle state place* (or *idle place*), and represents the state in which the corresponding processes (or parts) are idle. Each strongly connected state machine in Definition 1.3 represents the states that a part, initially in $p_{0_i}$, can reach during its processing. Places of $P_R$, called *resource places*, and associated arcs model how resources are used by the active processes.

Definition 1 imposes the existence of a minimal P–Semiflow $Y_r$ such that $Y_r[r] = 1$. For a given $p \in P_S$, $Y_r[p] = k$ ($\geq 0$) means that $k$ copies of resource $r$ are used by each process (token) in the state modelled by means of place $p$. Moreover, the invariant imposed by $Y_r$ ($\forall\, \mathbf{m} \in \mathcal{R}(\mathcal{N}, \mathbf{m_0})$, $Y_r \cdot \mathbf{m} = Y_r \cdot \mathbf{m_0} = \mathbf{m_0}[r]$) represents the fact that resources can be neither created nor destroyed. These P–Semiflows impose the resources to be serially reusable. Considering only acceptable initial markings ensures that the processing of each part in isolation is possible. For a given state place $p \in P_S$, $Y_R(p)$ denotes the multi–set of resources used by a process (a token) in the state modelled by place $p$: for every $r \in P_R$, $Y_R(p)(r) = Y_r[p]$. In the $S^*PR$ in Figure 1, the following elements can be found: $P_{0_1} = \{p1\_0\}$, $P_{S_1} = \{p1\_1, p1\_2, p1\_3, p1\_4\}$, $P_R = \{r1, r2, r3, r4\}$, $T_1 = \{t1, t2, t3, t4, t5, t6\}$, $Y_{r3} = 1 \cdot r3 + 2 \cdot p2\_1 + 1 \cdot p2\_2 + 1 \cdot p1\_2 + 1 \cdot p1\_3 + 1 \cdot p1\_4$, $Y_R(p2\_2) = 1 \cdot r3 + 1 \cdot r4$ [1].

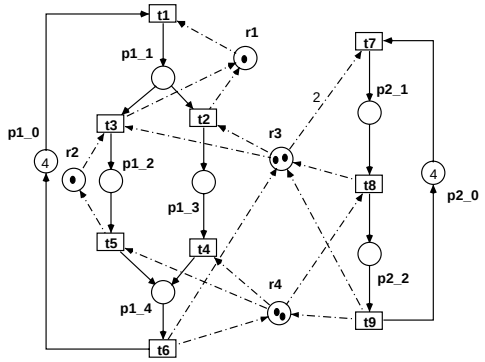


Fig. 1. A $S^*PR$.

An adaptation of the Banker's algorithm for deadlock avoidance in the class of $S^*PR$ nets is presented in [ETGC], whose run–time cost is polynomial in the Petri net model size. The algorithm looks for an ordering in the set of active processes such that the first process can terminate using its granted resources plus the free ones, the second process can terminate using the resources it holds plus the ones free upon termination of the first process, and so on. The basic step is to know if a process is able to terminate. Let us consider a reachable marking such that $\mathbf{m}[p] > 0$, with $p \in P_{S_i}$. A token in $p$ models an active process. If the rest of active processes do not move, the process modelled by one of the tokens in $p$ can use, in order to terminate, the resources granted to it ($Y_R(p)$) plus those resources free at $\mathbf{m}$.

A sufficient condition for the considered process to be able to terminate is the following:

- for every $q \in P_{S_i} \cup p_{0_i}$, mark $q$ if for every $r \in P_R$, $Y_r(q) \leq \mathbf{m}[r] + Y_r(p)$
- find a path of marked places joining $p$ and $p_{0_i}$

If such path exists, the process can terminate. Let us consider, for instance, marking $\mathbf{m} = 2 \cdot p1\_0 + 1 \cdot p1\_1 + 1 \cdot p1\_2 + 1 \cdot r3 + 2 \cdot r4 + 4 \cdot p2\_0$, reachable in the $S^*PR$ in Figure 1. Let us consider the token in place $p1\_1$ (corresponding to an active process in the state modelled by means of place $p1\_1$). If the previous method is applied to this process, places $p1\_0, p1\_1, p1\_3, p1\_4$ w ill be marked; since a path using these places exist from $p1\_1$ to $p1\_0$, the considered process can terminate. Notice also that, once this process terminates, also the process in $p1\_2$ can terminate, which allows us to conclude that the state corresponding to the chosen marking is safe.

## III. A class of Non–Sequential Resource Allocation Systems

A $S^*PR$ net is basically a set of sequential processes (state machines) that share a set of resources. This

[1] The multi–set notation will be used for this kind of vectors, as also for markings, P-Semiflows, etc.

structure can be easily generalised, allowing more general process structures.

*Definition 2:* A *Non–Sequential Resource Allocation Process* (NS–RAP) is a marked Petri $\langle \mathcal{N}, \mathbf{m_0} \rangle$, $\mathcal{N} = \langle P_S \cup P_0 \cup P_R, T, \mathbf{C} \rangle$, such that:
1. $P_S \cap P_R = P_S \cap P_0 = P_R \cap P_0 = \emptyset$,$P_S \neq \emptyset$, $P_R = \{r1, \ldots, r_k\} \neq \emptyset$
2. $P_0 = \{p_0\}$
3. $\mathcal{N}_{P_0 \cup P_S \cup T}$ is an ordinary, self–loop free, conservative and consistent Petri net
4. For every $r \in P_R$ there exists a non-empty set $\mathcal{Y}_r = \{Y_r^1, \ldots, Y_r^{k_r}\}$ of minimal P-Semiflows such that every $i \in \{1..k_r\}$ verifies that: (1) $\|Y_r^i\| \cap P_0 = \emptyset$ (2) $\|Y_r^i\| \cap P_R = \{r\}$
5. $P_S = \bigcup_{r \in P_R} \bigcup_{i \in \{1..k_r\}} (\|Y_r^i\| \setminus \{r\})$ [2]
6. $\mathbf{m_0}[p_0] = 1$; for every $r \in P_R$, $\mathbf{m_0}[r] > 0$; for every $p \in P_S$, $\mathbf{m_0}[p] = 0$

The conditions on the definition avoid undesirable behaviours. In particular, these conditions guarantee that the system is consistent and conservative. Otherwise it would be non-repetitive, and the resources would be consumed or generated under its execution.

Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a NS–RAP, let RG($\langle \mathcal{N}, \mathbf{m_0} \rangle$) be its reachability graph and RS($\langle \mathcal{N}, \mathbf{m_0} \rangle$) its reachability set (the set of nodes of RG($\langle \mathcal{N}, \mathbf{m_0} \rangle$)).

- $\mathcal{CC} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ denotes the set of strongly connected components (scc) of RG($\langle \mathcal{N}, \mathbf{m_0} \rangle$). Moreover, given a marking $\mathbf{m} \in$ RS($\langle \mathcal{N}, \mathbf{m_0} \rangle$), $\mathcal{C}^{\mathbf{m}}$ denotes the scc to which $\mathbf{m}$ belongs.
- Given a marking $\mathbf{m} \in$ RS($\langle \mathcal{N}, \mathbf{m_0} \rangle$), $Y_R^{\mathbf{m}}$ denotes the multi–set of resources the process is using at $\mathbf{m}$, and it is defined as follows: for every $r \in P_R$, $Y_R^{\mathbf{m}}[r] = \mathbf{m_0}[r] - \mathbf{m}[r]$.

Observe that the reachability graph of an NS–RAP may have more than one strongly connected component. Moreover, the process cannot be repetitive unless there is a non-trivial strongly connected component that contains the initial state. This is a necessary and sufficient condition for the production order to be executable in the system. This kind of NS–RAP will be said to be *well-defined*.

*Definition 3:* Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a NS–RAP. It is said to be *well–defined* if, and only if, $|C^{\mathbf{m_0}}| > 1$.

In the following, when talking about a NS–RAP it will be assumed to be well-defined. Given a NS–RAP, a $S^*PR$ can be obtained which is equivalent to $\mathcal{C}^{\mathbf{m_0}}$. The idea is to associate a place to each state (in particular the initial marking corresponds to the idle place), and a transition to each state change. The set of resources of the NS–RAP are also resources of the $S^*PR$, and are connected to the net in such a way that their marking evolves according to the reachability graph.

*Definition 4:* Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a (well–defined) NS–RAP. The *associated unfolded system* is the $S^*PR$ $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle$, $\mathcal{N}^* = \langle P_S^* \cup P_0^* \cup P_R^*, T^*, \mathbf{C}^* \rangle$, defined as follows:

- $P_S^*$ contains a place $p_{\mathbf{m}}$ for each $\mathbf{m} \in \mathcal{C}^{\mathbf{m_0}} \setminus \{\mathbf{m_0}\}$
- $P_0^*$ contains a unique place $p_{\mathbf{m_0}}$

[2]Every result presented here is also valid if this assumption is withdrawn.

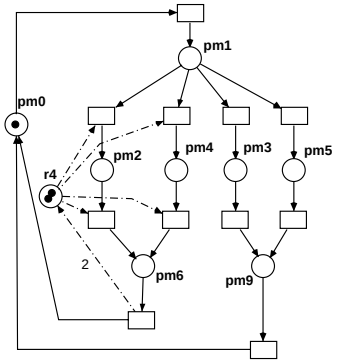


Fig. 3. The unfolded system corresponding to the well defined NS–RAS in Figure 2. For the sake of clarity, only resource $r4$ has been drawn. Notice that no place has been generated corresponding to markings $\mathbf{m}_7$ and $\mathbf{m}_8$ (since they do not belong to $C^{\mathbf{m_0}}$).

- $P_R^*$ contains a place $p_r$ for each $r \in P_R$
- $T^*$ contains a transition $t_{\mathbf{m}_1,\mathbf{m}_2}$ for each $\mathbf{m}_1[t'\rangle\mathbf{m}_2$ belonging to RG($\langle \mathcal{N}, \mathbf{m_0} \rangle$) such that $\mathbf{m}_1, \mathbf{m}_2 \in \mathcal{C}^{\mathbf{m_0}}$
- $\mathbf{C}^*$ is defined as follows:
  - $\mathbf{C}^*[p_{\mathbf{m}_1}, t_{\mathbf{m}_1,\mathbf{m}_2}] = -1$
  - $\mathbf{C}^*[p_{\mathbf{m}_2}, t_{\mathbf{m}_1,\mathbf{m}_2}] = 1$
  - for every $p_r \in P_R^*$, $\mathbf{C}^*[p_r, t_{\mathbf{m}_1,\mathbf{m}_2}] = Y_R^{\mathbf{m}_1}[r] - Y_R^{\mathbf{m}_2}[r]$
  - $\mathbf{C}^*[p_{\mathbf{m}}, t_{\mathbf{m}_1,\mathbf{m}_2}] = 0$ for any other place
- $\mathbf{m_0}^*[p_{\mathbf{m_0}}] = 1$; for every $p_r \in P_R$. $\mathbf{m_0}^*[p_r] = \mathbf{m_0}[r]$, while $\mathbf{m_0}^*[p_{\mathbf{m}}] = 0$ for every $p_{\mathbf{m}} \in P_S^*$

*Proposition 5:* [ER02] Let $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle$, $\mathcal{N}^* = \langle P_S^* \cup P_0^* \cup P_R^*, T^*, \mathbf{C}^* \rangle$, be the associated unfolded system of $\langle \mathcal{N}, \mathbf{m_0} \rangle$, $\mathcal{N} = \langle P_S \cup P_0 \cup P_R, T, \mathbf{C} \rangle$. Then, every $p_r \in P_R^*$ is an implicit place

*Proposition 6:* [ER02] Let $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle$, $\mathcal{N}^* = \langle P_S^* \cup P_0^* \cup P_R^*, T^*, \mathbf{C}^* \rangle$, be the associated unfolded system of $\langle \mathcal{N}, \mathbf{m_0} \rangle$, $\mathcal{N} = \langle P_S \cup P_0 \cup P_R, T, \mathbf{C} \rangle$. Then, $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle$ is an $S^*PR$ with an acceptable initial marking.

It is important to remark the fact that the $S^*PR$ obtained in the previous proposition is composed by just one type of process (this means that the set of indices $I_{\mathcal{N}}$ in Definition 1 is composed by just one element). Let us now introduce the class of NS–RAS, corresponding to the concurrent processing of a set of NS–RAP. A NS–RAS is obtained by means of the composition of a non–empty set of NS–RAP by fusion of the common places modelling resources. First, let us introduce what do we mean by composition.

*Definition 7:* Let $\langle \mathcal{N}_i, \mathbf{m}_{0_i} \rangle$, $\mathcal{N}_i = \langle P_i, T_i, \mathbf{C}_i \rangle$, $i \in \{1..2\}$, be two Petri nets. They are said to be *composable* if, and only if, $P_1 \cap P_2 \neq \emptyset$, $T_1 \cap T_2 = \emptyset$ and $\forall\, p \in P_1 \cap P_2$, $\mathbf{m}_{0_1}[p] = \mathbf{m}_{0_2}[p]$.

The resulting composed net is defined as follows: $\langle \mathcal{N}, \mathbf{m_0} \rangle$, $\mathcal{N} = \langle P, T, \mathbf{C} \rangle$ where:

- $P = P_1 \cup P_2$
- $T = T_1 \cup T_2$
- for every $p \in P$ and every $t \in T$, $\mathbf{C}[p,t] =$ If $t \in T_1, p \in P_1$ Then $\mathbf{C}_1[p,t]$ Else If $t \in T_2, p \in P_2$ Then $\mathbf{C}_2[p,t]$ Else 0.

The previous definition can be extended in the natural way to the case of any finite number of components.

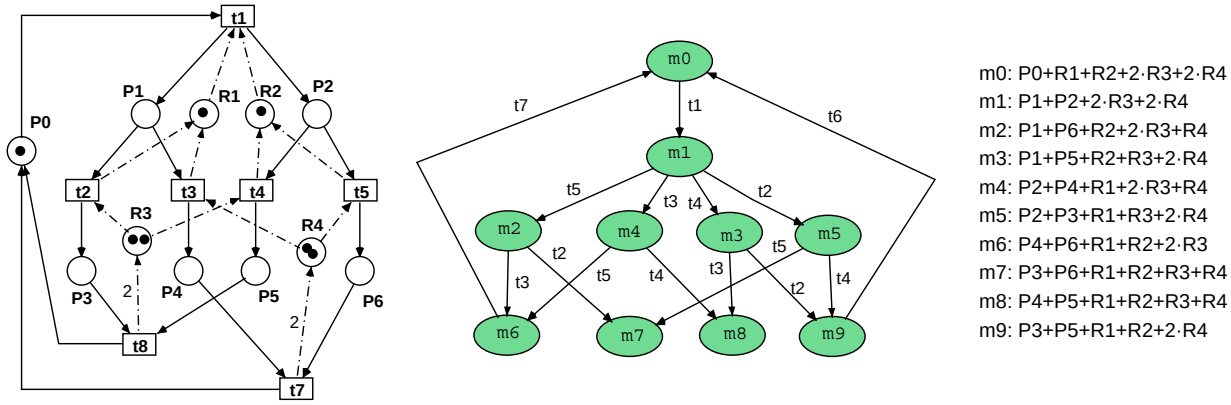


Fig. 2. A well defined NS–RAS and its reachability graph.

In the following, and being $\langle \mathcal{N}_i, \mathbf{m}_{0_i} \rangle$, $i \in \{1..n\}$, a set of composable Petri nets, $\bigcirc_{i=1}^{n} \langle \mathcal{N}_i, \mathbf{m}_{0_i} \rangle$ will denote the composed net. In the case of NS–RAP, the "composability" notion will be constrained to the case in which the common places belong to the set of resources. The interpretation of this constraint is clear: each one of the involved NS–RAP will model the production of a part. The interaction among the parts is due to the fact that they have to compete for the set of common resources; then, only places modelling resources can be shared among different NS–RAP.

*Definition 8:* A *non–sequential resource allocation system* (NS–RAS) is defined recursively as follows:

- A NS–RAP is a NS–RAS.
- The composition of two NS–RAS via a set of common resource places is also a NS–RAS.
- All the NS–RAS are obtained using the previous rules.

In the following, any NS–RAS will be assumed to be of the form $\bigcirc_{i=1}^{n} \langle \mathcal{N}_i, \mathbf{m_0}_i \rangle$, for some $n \geq 1$.

*Proposition 9:* Let $\langle \mathcal{N}, \mathbf{m_0} \rangle = \bigcirc_{i=1}^{n} \langle \mathcal{N}_i, \mathbf{m_0}_i \rangle$ be a NS–RAS composed of a set of well–defined NS–RAP. Then, $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle = \bigcirc_{i=1}^{n} \langle \mathcal{N}_i^*, \mathbf{m_0}_i^* \rangle$ is a set of $S^*PR$. Notice that $\langle \mathcal{N}^*, \mathbf{m_0}^* \rangle$ can be, in general, a set of $S^*PR$ instead of just one. This is due to the fact that it is possible for the resulting net to be a set of disjoint strongly connected nets. This is not a drawback for our purposes. Since we are looking for a deadlock avoidance algorithm, in the case of having a set of separated systems (which means there is no interaction among them) it is enough to control each one of them. Therefore, in the following we are going to assume that the composition of the unfolded system forms one unique $S^*PR$.

## IV. A deadlock avoidance control policy for NS–RAS

In [ETGC] an adaptation of the Banker's algorithm for deadlock avoidance was developed for the class of $S^*PR$. Here we will use this algorithm for deadlock avoidance in NS–RAS systems. The idea is to have an schema as shown in Figure 4.

Initially, an NS–RAS model that represents the system in an adequate detail is developed. This means, for example that local faults that can be managed by the local controllers are not explicitly represented. Then, the associated $S^*PR$ system is obtained. In the execution, the controller uses the information about the real state of the system and updates the associated state in the NS–RAS model to deduce the different possible system evolutions. From that set of possible transition firings the controller has to extract a subset that guarantees that no deadlock will be achieved. That will be done as follows: for each transition firing, obtain the marking $\mathbf{m}$ that would be reached. This marking has an equivalent marking $\mathbf{m}^*$ in the associated $S^*PR$. Apply the Banker's algorithm developed in [ETGC] to this marking. If deadlock-freeness can be guaranteed, the transition is effectively enabled. Otherwise, since it cannot be safely fired it should be removed from the list of possible evolutions. The controller should be completed by adding any other module (scheduler, strategies to optimise the production,...) that may provide information to choose the the most adequate action among the enabled ones.

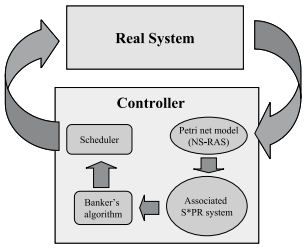


Fig. 4. Schema of the proposed controller.

Observe that it would be possible to remove the NS–RAS model of the system, and use the associated $S^*PR$ to decide the possible actions. However, the NS–RAS model is more intuitive and can be more easily checked by a human operator to supervise the behaviour or decide what should be done in a certain moment.

## V. An application example

Figure 5 shows the plant of a manufacturing cell consisting of six machines (M1 to M6) that process the

components, one buffer with place to store up to 16 intermediate products, and two robots (R1 and R2). The process is organised in two rings, with the buffer connecting them. The final product (Figure 6) is composed of a base on which three cylinders are set. The base may be black or white, and there are two types of cylinders: cylinders that are composed of a case, a piston, a spring, and a cover (that we will call "complete") and cylinders with just a case and a cover (that we will call "hollow"). In both kinds of cylinders the cases may be red, black or metallic.

Fig. 6. The kind of product that the system in Figure 5 produces.

As raw materials there are bases, pistons, springs, covers, cases, and cases with a cover. We assume that an unbounded amount of raw material feed the system.

The processing goes as follows: machine M1 takes a case from a feeder, and verifies that it corresponds to the order, that is, if the colour is correct and whether it has a cover or not. If it is not correct, then it is discarded, otherwise, it is put on a pallet, and the kind of processing that the part needs is written on the pallet. If the part already has a cover, a switch is activated to carry it directly to M4. Otherwise it goes to M2. Machine M2 puts the piston and the spring, if the cylinder needs them, and then the part goes to M3, which adds the cover. In M4 the parts are verified, the pallets are released and the parts are put on a conveyor that moves them to the entrance of the buffer. Machine M5 can temporarily store the cylinders in the buffer. When needed to assemble the final product, M5 puts them in a conveyor that takes them to robot R1. Machine M6 puts a base of the right colour on a pallet, and it is carried to robot R1. The robot takes the three cylinders one by one and puts them on the base. The product is then complete, and goes to robot R2, which takes it out of the system. A set of different 312 products can be composed using these materials.

An NS–RAP can be used to model each kind of product For example, in Figure 7 a Petri net that models the assembly of a product made of three complete cylinders is shown. The resource places have been distinguished by a "tag", so removing those places what remains is the process plan. It can be seen that there is a transition with several output "process places", the one that divides the order into sub-orders of different pieces and bases. There are also transitions with several input "process places", when the pieces are put on the base. In this example the resources are of two kind. On the one hand there are machines, robots, and space in the intermediate buffer (i.e, physical constraints). On the other, there are constraints that are not strictly necessary but are advisable for the correct evolution of the system, for example not to allow more than one pallet on each conveyor segment.

If all the different products are composed fussing the places of the common resources, what we have is an NS–RAS, and so the technique described in Section IV can be applied to it.

In order to check the usefulness of the proposed approach, a prototype of the program checking for safeness has been implemented. The associated $S^*PR$ associated to the NS–RAP in Figure 7 has been computed. It has 2442 state places and 7814 transitions. Checking if the processing of such type of parts can terminate with the free resources has been implemented using the set of graph libraries of the LEDA package (distributed by Algorithmic Solutions Software GmbH). In mean, its takes about 0.012 CPU seconds using a PIII processor at 1.0 GHz with a Linux RedHat 7.2 operating system (this computation uses a Depth First Search algorithm, which is linear in the size of the unfolded system). Given that no more than 26 components can stay at the same time in the system (considering the 10 pallets plus the 16 storage places in Figure 6) and that a direct implementation of the algorithm in [ETGC] grows in a quadratic way with respect to the number of active processes, the whole time to know if a system state is safe takes about 8 CPU seconds.

## VI. Conclusions

The paper has proposed a deadlock avoidance solution to the case of NS–RAS systems. It is based on two main elements. First, the computation of the reachability graph of the isolated execution of each production order. Second, the application of a Banker's approach for deadlock avoidance to the $S^*PR$ resulting from the composition of the unfolded systems. The usefulness of the method has also been shown by means of its application to a real example. It is important to remark the fact that the size of the reachability graphs of the isolated execution of each production order is tiny compared to the size of the reachability graph of the whole system, which makes the approach to be promising to solve the deadlock problem for this kind of systems. The experimental computations have been carried out using libraries implementing standard graph algorithms. We feel that the use of data structures and algorithm implementations specifically adapted to the kind of systems we are dealing with can give better performance results.

*Acknowledgments*

This work has been supported by the Spanish research project CICYT and FEDER TIC2001-1819

## References

[BK90] Z.A. Banaszak and B.H. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. on Robotics and Automation*, 6(6):724–734, December 1990.

[CX97] Feng Chu and Xiaolan Xie. Deadlock analysis of Petri nets using siphons and mathematical pro-

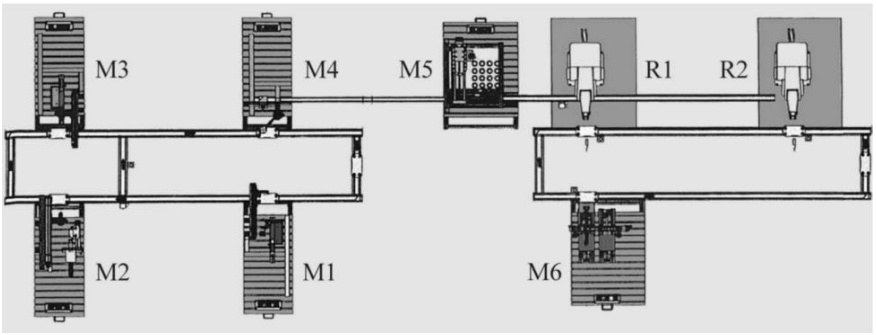


Fig. 5. A plan of the physical system.

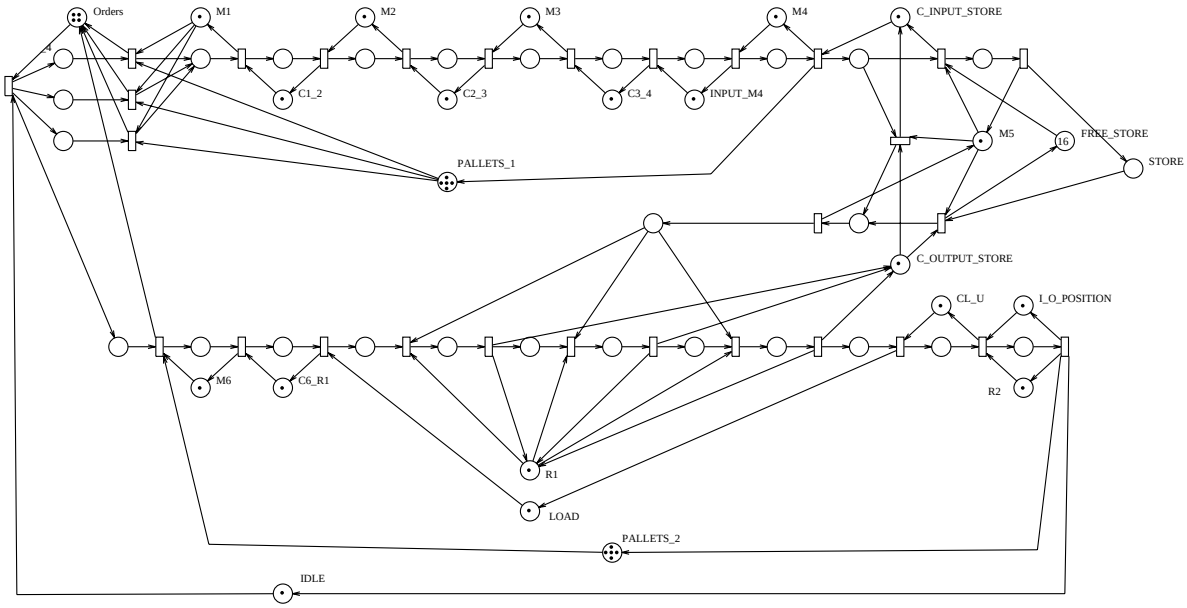


Fig. 7. The NS–RAP modelling the assembly of a product made of three complete cylinders and a base.

gramming. *IEEE Transactions on Robotics and Automation*, 13(6):793–804, December 1997.

[ECM95] J. Ezpeleta, J.M. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. on Robotics and Automation*, 11(2):173–184, April 1995.

[ER02] J. Ezpeleta and L. Recalde. A deadlock avoidance approach for non-sequential resource allocation systems. Research Report RR-02-05, Dept. of Computer Science. University of Zaragoza, María de Luna,3 - 50018-Zaragoza. Spain, May 2002.

[ETGC] J. Ezpeleta, F. Tricas, F. García-Vallés, and J.M. Colom. A Banker's solution for deadlock avoidance in FMS with routing flexibility and multi–resource states. *To appear in IEEE Transactions on Robotics and Automation.*

[Jen96] M.D. Jeng. A Petri net synthesis theory for modeling flexible manufacturing systems. *IEEE Trans. on Systems, Man and Cybernetics-Part B: Cybernetics*, 27(2):169–183, April 1996.

[LRF97] M. Lawley, S. Reveliotis, and P. Ferreira. FMS structural control and the neighborhood policy: Part 1 correctness and scalability. *IIE Transactions*, (29):877–887, 1997.

[Rev98] S. Reveliotis. Accommodating FMS operational contingencies though routing flexibility. In *Proceedings of the 1998 Int. Conference on Robotics and Automation*, pages 573–579, Leuven, Belgium, may 1998. IEEE.

[RR92] Z.Banaszak R.Wojcik and E. Roszkowska. Automation of self–recovery resource allocation procedures synthesis in FMS. In K. Leiviska, editor, *IFAC CIM in Process and manufacturing Industries*, pages 127–132, Espoo, Finland, 1992. Oxford: pergamon press.

[RW92] E. Roszkowska and R. Wojcik. Problems of process flow feasibility in FAS. In K. Leiviska, editor, *IFAC CIM in Process and manufacturing Industries*, pages 115–120, Espoo, Finland, 1992. Oxford: pergamon press.

[VNJ90] N. Viswanadham, Y. Narahari, and T.L. Johnson. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models. *IEEE Trans. on Robotics and Automation*, 6(6):713–723, December 1990.

[XJ99] Xiaolan Xie and MuDer Jeng. ERCN–merged nets and their analysis using siphons. *IEEE Transactions on Robotics and Automation*, 15(4):692–703, aug 1999.