DESIGN OF DEADLOCK AVOIDANCE COMPENSATORS FOR ANTHROPOCENTRIC PRODUCTION SYSTEMS

P.E. Miyagi, D.J. Santos Filho, W.M. Arata
University of São Paulo, Escola Politécnica, São Paulo, Brazil
pemiyagi@usp.br diolinos@usp.br

The advances in networked industrial enterprises to meet the demands for high productivity level as well as for quality and operational flexibility has driven the development of anthropocentric system that emphasize the importance of human operator. However, greater flexibility and autonomy means greater complexity in the flow of items (material and information) in a system. Thus, this work discusses the concept of Anthropocentric Production Systems (APS) and how it should be implemented do avoid deadlock problem. In particular, the techniques based on Petri nets and Request-Allocation Graphs (RAG) are introduces to the design of deadlock avoidance compensators, employed to support the task of system control program redesign, the only solution in case of extreme changes in the environmental conditions.

1. INTRODUCTION

Anthropocentric Production Systems (APS) (e.g.: Camarinha Matos, 1995; Erbe, 1998) are characterized by an intensive interaction between human and machines and there exist a great diversity of human commands that are issued to machines accomplish the production processes. One important part of these commands is related to the use of resources considering the aspects such as productivity and quality. Due the proper nature of human, APS often involves conflict arbitration and changes in the conditions of the dynamics of the system. In this context, the advances in networked industrial enterprises reinforce the need of an adequate method to model and implement dynamic resource allocation management procedures, i.e., it is necessary to develop an efficient control strategy. A deadlock in the flow of items (including material and information) in a system occurs when it reaches a situation where the system blocks due to (normally considered) defective logic in the coordination of operations. Since it is related to the interruption of operations, this is the main problem to be solved by of the control strategy.

Our approach to deal with deadlocks in APS is a technique to support the human operator in the task of reprogramming the control system in response to extreme changes in the operating environment (i.e., the kind of change that makes the current control logic inconsistent with the new system parameters, demanding a structural reformulation or even a complete redefinition of this logic). Another goal is also to improve system responsiveness, one of the main goals of APS.

This paper presents, at first, the architecture of APS control systems. Following, a discussion about deadlocks in APS and how Petri nets and Request/Allocation Graphs are used to model the system are described. An example illustrates the effectiveness of the proposal in the design anti-deadlock compensators for APS.

2. CONTROL SYSTEM OF APS

The basic APS control architecture is illustrated in figure 1. The concept considers that the ability of reactive behavior of such architecture is improved from the presence of the human element that be possible the dynamic control reconfiguration.

Figure 1 Control architecture of APS

However, evidently, the effectiveness of such architecture is strongly dependent on the performance of the human element.

- (a) primarily, the human element is the responsible for the monitoring of the system global state;
- (b) once a change in the constraints (e.g., in the state or quantity of available resources) is detected, the human element should be capable to start up an analysis of the effects of such changes;
- (c) after the analysis, the human element should reprogram the control system adequately, i.e., by incorporating the changes necessary to follow the desired dynamic evolution of the system.

In this procedure, we note that the human element interacts with the machines or devices of the control object and of the control systems based on the decision about the resource allocation. And the main problem in system is the occurrence of a deadlock due an inappropriate resource allocation decision.

3. SPECIFICATION OF THE DEADLOCK PROBLEM

Deadlock has been studied in the areas of operating systems of computer systems and flexible manufacturing systems (e.g.: Banaszak, 1990; Ezpeleta, 1995; Hasegawa, 1996; Xing, 1996), but, unfortunately, due to characteristics of APS (e.g., the presence of sequencing constraints) the solutions developed for them are not efficient¹. Therefore, a new way to solve deadlock problem in APS is

¹ There are four necessary conditions to the occurrence of deadlock in concurrent processes systems (Banaszak, 1990; Hasegawa, 1996a): mutual exclusion, retention while waiting, non preemption and circular waiting. To avoid deadlocks, it is sufficient to ensure that at least one of the above conditions will never occur.

introduced. It is adopted a deadlock freeness concept by construction approach based on the specification of a control strategy based on Condition Event nets (C/E nets), a class of Petri net (Miyagi, 1996).

Through an interpretation of the structural elements of the C/E nets it is possible to describe the activities, events, conditions of the APS and through the marking concept and firing rules of the transition elements, the resulting graph model the dynamic behavior of the APS (Hasegawa, 1999).

3.1 Request/Allocation Graphs

Despite the fact that C/E nets models the sequence of operations of the production process and the resources utilization of the system, it does not give any information about the order of the allocation of such resources. Then the Request/Allocation Graphs (RAG) are introduced to model the sequence of the resource allocation in APS.

Given a set of resources $R=\{r_1, r_2, \dots, r_R\}$, where each of the r_k can provide multiple units, a set of products $Q=\{q_1, q_2, \dots, q_Q\}$, and, for each $q \in Q$, a sequence of process steps S_q , with $s_q(k)$ denoting the k -th step in the sequence to make a product q , and $S = S_{q_1} \cup S_{q_2} \cup \dots \cup S_{q_Q}$, a RAG is an oriented graph (R, S) with nodes $r \in R$ and interconnected by arcs corresponding to process steps $s \in S$. Being $r_q(k)$ the resource necessary to execute the k -th step of the process of product q , an allocation pair in a RAG is a pair (r, s) formed by a node $(r \in R)$ and one of its outgoing arc $(s \in S)$ such that $r_q(k)=r$ and $s_q(k)=s$, i.e., the k -th step of sequence S_q needs resource r . A request pair in a RAG is a pair (s, r) formed by an arc $(s \in S)$ and the node at its arrowed end $(r \in R)$ such that $r_q(k+1)=r$ and $s_q(k)=s$, i.e., the $(k+1)$ -th step of sequence S_q needs resource r . Figure 2 illustrates both types of pairs.

Figure 2 Allocation and request pairs relative to the same product, in a certain step in production sequence

In the general case, the RAG model of a C/E net is obtained by directly applying the definitions of allocation and request pairs.

3.2 Segments and closed loops in a RAG

A continuous connection is a sequence (a path) of allocation pairs in a RAG such that it represents, partially or in whole, a sequence of production steps relative to one product $q \in Q$. Considering a RAG corresponding to an APS with resource sharing, a closed loop can be found if a continuous connection crosses itself (at an *intersection node*) before its last allocation pair or if the continuous connections of different processes intersects themselves before their respective last allocation pairs, a closed waiting loops (CWL). In a continuous connection with CWL, we define IAP (intersection allocation pair) the allocation pair whose node is the intersection node;

the allocation pair that points at an intersection node is called TAP (terminal allocation pair). *Segment* is the continuous connection from an IAP to the following IAP. In the general case, a CWL is made up of various segments, so the connection of these segments is classified as *non-continuous connection* due to the involvement of multiple production sequences. Figure 3 shows a CWL, IAP and TAP of a RAG.

Figure 3 RAG model containing CWL, IAP and TAP

3.3 Deadlock avoidance procedure

A CWL can reach a state where all the steps along the loop are holding resources while waiting for the release of the resources for the next step (retention while waiting), characterizing a *retention state* (analogous to the circular wait state in Hasegawa, 1996). So, a priori, any CWL can assume a retention state.

Given that a CWL is in retention state if and only if all of its resources are busy, then, in this case, retention state means deadlock in the flow of items and, therefore, all segment in this circuit are blocked, waiting for resource releasing. In this context, there are the following theorem (Hasegawa, 1996):

- Theorem 1. Assuming that a system has exactly one CWL, the system is in a retention state if and only if all the allocation pairs in the CWL are blocked.

Since the operations within a CWL necessarily have to get into an intersection node (an IAP), it can be noted that compensating the resource of the IAP so that it does not block, it is possible to prevent the segment to block. This strategy of control to avoid deadlocks is called Flow-in Suppression, FIS (Banaszak, 1990).

3.4 Detection of Closed Loops in a RAG

The detection of CWL in a RAG model becomes increasingly onerous as the degree of complexity of the structure and dynamic behavior of the modeled production system. Therefore, it is introduced a systematic way of identifying CWLs in RAGs based on the works of (Hasegawa, 1996).

- Definition 1. Let $RAG=(R,A)$ be an oriented graph, where $R=\{r_1, r_2, \dots, r_N\}$ is a set of resources and A is a set (r_i, r_j) pairs, with $i \neq j$. The RAG algebraic matrix $M=(m_{ij})$ is such that:
 - $M_{ij}=s_{ij}$ if $(r_i, r_j) \in A$
 - $M_{ij}=0$, otherwise

where s_{ij} is a Boolean algebraic term; a non null s_{ij} means that there is a process k -th step $s_q(k)$ (relative to a product $q \in Q$) which is annotated on the arc from r_i to r_j in the RAG.

Replacing the Boolean terms by 1 in matrix M yields matrix G , that turns out to be the associated matrix of the RAG (Harari, 1971; Callioli, 1978).

- Theorem 2. Let $RAG=(R,A)$ be an oriented graph and G , its associated matrix. Then, the number of paths with length p between nodes R_i and R_j is the element $g(i,j)$ in the matrix G^p (Callioli, 1978).

With matrix M and using Theorem 2, we elaborate a procedure to determine the closed loops in a RAG in terms of process steps by calculating M^p for successive values of p (where Boolean matrix multiplication arranges the operations and operands in the same way as the conventional form, but, instead of addition and multiplication, we apply the AND and OR logic operations). Then, each element $(m^{(p)})_{ij}$ of matrix M^p is a Boolean algebraic expression of the form

$$(m^{(p)})_{ij} = \sum_{u=1,n} (\sum_{v=1,p} s_q(k))$$

where p is the length of the paths, n is the number of paths with length p and $s_q(k)$ is a Boolean algebraic term whose presence represents that a process step $s_q(k)$ is part of the path from node r_i and r_j in the RAG. The expression $(m^{(p)})_{ij}$ represents all the p -step paths connecting R_i and R_j in the RAG. So, as a non null element $(m^{(p)})_{ii}$ on the principal diagonal is obtained, the expression contained in this element represents all the possible paths that start and end at R_i with length p . This is how the closed loops are obtained. However, in order to identify such closed loops, at each step of the matrix multiplication to calculate the power of M , the elements on the principal diagonal must be eliminated so that reentrant loops are not computed thereafter. The powers of M are calculated until M^p becomes a null matrix or until $p = N$ (the dimension of M). It is not necessary to increment p beyond N because that would compute circuits with length greater than the number of (R_i, R_j) pairs, i.e., reentries would be computed. Then, we derive the following algorithm to obtain the a list of CWLs from RAGs:

```

begin_program
  obtain matrix M from RAG;
  W = I; // initialize matrix W with identity matrix; W holds the powers of M
  j = N; / (dimension of matrix M and W)
  while (W ≠ 0 and j > 0) do
    W = W * M;
    j = j - 1;
    i = 0;
    while (i ≤ 0) do
      if (wii ≠ 0) then
        add wii to the list of CWL;
        set element (i,i) of W to null;
      end_if;
      i = i + 1;
    end_while;
  end_while;
end_program;

```

5. EXAMPLE

In this section, we present an example of the application of the proposed technique for deadlock avoidance. Consider figure 4 that illustrates a C/E net of a APS dealing with three kinds of products (A, B, and C) and three process sequences, whose steps are represented by conditions (boxes) denoted by $s_A(k)$, $s_B(k)$, and $s_C(k)$ (with k indicates the sequence). Its resources are modeled by places (circle): for each product type, input buffers (B-INA, B-INB, and B-INC) and output buffers (B-OUTA, B-OUTB, B-OUTC) and processing stations (EST-1, EST-2, EST-3, and EST-4); the marking (tokens) in each place represents the number of units of the respective resource. Places C2 and C1 and the arcs that connect them to the graph represent the compensators that are added to the C/E net after working with the RAG model, as we discuss now.

Figure 5 C/E net relative to the processes to be executed (including the compensation elements C_1 and C_2 that are added after RAG analysis)

From the C/E net model (without compensation elements C_1 and C_2), a RAG model is built, as shown in figure 5 (the meaning of the dashed lines are explained soon), and whose algebraic matrix (M) is given by:

$$M = \begin{bmatrix} 0 & s_A(1) & s_B(2) & s_C(3) \\ s_B(1) + s_C(2) & 0 & s_A(2) & 0 \\ 0 & s_C(1) & 0 & s_A(3) \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(note that "+" stands for the logical OR). The algebraic Boolean expression that represents all the CWL in the RAG, resulting from the algorithm in section 3.4 is: $s_B(1).s_B(2).s_C(1) + s_C(2).s_B(2).s_C(1) + s_B(1).s_A(1) + s_C(2).s_A(1) + s_A(2).s_C(1)$.

Figure 5 RAG model (including the compensators)

Using this relation, we design of the deadlock compensation. The allocation pair(s) must be defined to compensate a special type of resources (remembering that one non-blocking allocation pair in a CWL suffices to avoid deadlock) and the idea is to constrain the resource allocation process (or Flow-in Suppression, FIS). For instance, let us consider process steps $s_A(1)$ and $s_C(1)$: observing the algebraic expression, it suffices that we ensure that the respective allocation pairs do not reach a “retain while waiting” situation so that the retention state in all of the CWL identified in the RAG is avoided.

Focusing on $s_C(1)$, that allocates EST-3, the loop $\{s_C(1), s_A(2)\}$ will be in a retention state if all units of EST-3 are in $s_C(1)$ and all units of EST-2 in $s_A(2)$ (analogous statements can be made about the other CWL). Then, it is sufficient to ensure that the allocation pair with $s_C(1)$ never holds all units of EST-3, and we do that by adding another constraint in the allocation of EST-3 by $s_C(1)$: we state that $s_C(1)$ can only allocates one resource at a time and only if there is no process steps $s_C(1)$ and $s_C(2)$ in progress (described by the dashed line label C_1 in figure 5). This is a case of *compensation per segment*.

Considering $s_A(1)$, making the same considerations, we apply a *complex compensation*, involving more than one segment. Now, we only allow that either

$s_A(1)$ or $s_B(1)$ (not both) can allocate their respective resources (but never all of the units) at a time. Then we obtain the compensated C/E net as shown in figure 4.

6. FINAL REMARKS

This work characterizes the deadlocks in APS, showing the necessary conditions to their occurrence. Then, the modeling of APS using C/E nets and RAG including the concepts of segment and closed waiting loop (CWL) were presented in the focus of the design of the deadlock avoidance mechanism: by analyzing the CWL, the relevant IAPs (intersection allocation pairs) to compensate are identified and, thus, the adequate compensator elements are selected. In (Santos Filho, 1998), there is several examples of cases that illustrates how the proposed method can be associated to the design of the control system of a production system using vehicles driven by human beings able to make decisions about the trajectories in a APS context.

Acknowledgements

The authors would like to thank the governmental agencies FAPESP and CNPq for the financial support.

7. REFERENCES

1. Camarinha Matos LM, Afsarmanech H (ed.) *Balanced Automation Systems*, Chapman & Hall., 1995.
2. Banaszak ZA, Krogh BH. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. on Robotics and Automation*, 1990, 6(6): 724-34.
3. Callioli CA et al. *Álgebra Linear e Aplicações*. São Paulo: Atual Editora, 2nd ed., 1978. (In Portuguese)
4. Erbe HH. "Assistance systems supporting operators in manufacturing". In: *Intelligent Systems for Manufacturing*, L.M. Camarinha Matos et al. (ed.), Kluwer Ac. Publ., 1998.
5. Ezpeleta J et al. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. on Robotics and Automation*, 1995, 11(2): 173-84.
6. Harari F. *Graph Theory*. Addison-Wesley Publ. Company, 2nd ed., 1971.
7. Hasegawa K. Modeling, control and deadlock avoidance of flexible manufacturing systems. *Proc. of 11th Brazilian Congr. of Automatics*, São Paulo: SBA, 1996, 1: 37-51.
8. Hasegawa K et al. Graphical analysis and synthesis of in flexible manufacturing systems. *Proc. of 17th Intern. Conf. on Application and Theory of Petri Net*, Osaka: SICE, 1996a, 161-176.
9. Hasegawa K et al. Transform of discrete production net request/allocation graph. *Proc. of SICE'96 Joint Symp. on System and Information*, Matsuyama: SICE, 1996b, 213-18 (in Japanese).
10. Hasegawa K et al. Deadlock avoidance policy based on request/allocation graph. *Proc. of SICE'96 Joint Symp. on System and Information*, Matsuyama: SICE, 1996c, p.219-24 (in Japanese).
11. Hasegawa et al. On resource arc for Petri net modelling of complex resource sharing system. *Journal of Intelligent and Robotic Systems*, 1999, 26: 423-437.
12. Miyagi PE. *Controle Programável - Fundamentos do controle de sistemas a eventos discretos*. São Paulo: Ed. Edgard Blücher Ltda., 1996.
13. Santos Filho DJ. *Controle de Sistemas Antropocêntricos de Produção Baseado em Redes de Petri Interpretadas*. Doctor Thesis, São Paulo: University of São Paulo, 1998. (In Portuguese)
14. Xing K et al. Deadlock avoidance policy for Petri-net modeling of flexible manufacturing systems with shared resources. *IEEE Trans. on Robotics and Automation*, 1996, 41(2): 289-95.