Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

A Modular Approach for Deadlock Avoidance in FMS

Luigi Piroddi, and Luca Ferrarini, *Senior Member, IEEE*

***Abstract*—In Petri net models of FMS, deadlock avoidance policies based on siphon control may require an excessive computational load and result in over-sized control sub-nets. In this work, a simple approach is proposed for the design of sub-optimal but compact controllers. The approach is based on the separate control of two sub-sets of resources, and an anticipated booking mechanism for one of the two sets of resources that decouples the two sub-models. An illustrative example is provided, which shows how the partition design influences the control quality and performance.**

I. Introduction

FLEXIBLE manufacturing systems (FMS) are automated, multi-product systems, where any system resource can be shared by more operations of the same or different production sequences. The presence of such shared resources can cause the FMS to *deadlock* [1, 3, 13, 14]. In a deadlock state, in order to complete the routing of a subset of products, a subset of resources is requested, but cannot be used because it is blocked by the same subset of products. Notice that though deadlocks may involve only a subset of the resources of an FMS, they represent highly disadvantageous and potentially dangerous situations that cause system performance to decrease seriously, both in terms of total products processed and global processing time.

In the literature, the deadlock problem has been widely studied and many algorithms have been proposed to deal with it. These can be grouped in the following categories: *deadlock detection/recovery* (DR) (see e.g. [18]), *deadlock prevention* (DP) [2, 9, 17, 20], and *deadlock avoidance* (DA) [1, 4, 7, 14, 19] methods. DR methods allow the occurrence of deadlocks and specify procedures to recover the correct process functioning. In DP methods, the system model is modified off-line so that the resulting controlled model turns out to be deadlock-free. DA algorithms check the system flow on-line to see that the system does not fall in a deadlock state. Both DP and DA approaches finally amount to setting up a suitable control policy to regulate the shared resource allocation in the system.

Regardless of the type of approach chosen, the elimination of deadlocks is a complex activity, because of the state explosion in discrete event systems. Referring to Petri net (PN) models [10, 12, 22], deadlocks are related to siphon structures getting devoid of tokens. Briefly, a siphon is a set of places which remains permanently unmarked once it loses all tokens [10]: then, if a siphon gets unmarked during net evolution, some transitions become permanently disabled, causing a partial or total system deadlock. Deadlocks can be avoided by resorting e.g. to P-invariant based control [9], a method which implements a constraint for every siphon preventing it from getting unmarked. Unfortunately, the number of minimal siphons grows exponentially with size, so that even small size problems may turn out to be intractable (see the example in Section IV). In addition, every new constraint may actually generate additional siphons, that may also potentially cause deadlock. In general, the DA problem is known to be *NP*-hard, if only maximally permissive solutions are sought [3, 11]. Clearly, for a practical and successful application of design and control methodologies based on PNs, a pragmatic approach must be adopted to deal with the deadlock problem, in order to obtain reliable and applicable solutions for an industrial context. Sub-optimal control solutions that are not maximally permissive, but require a finite computational time and amount to a limited number of additional control places (as compared to the size of the original PN model) may be preferable.

In this paper a sub-optimal modular approach to DA is proposed, based on the separate control of two sub-sets of resources, and an anticipated booking mechanism for one of the two sets of resources that decouples the two sub-models. A similar technique specialized to batch processes is discussed in [16], where two separate sub-models are considered, describing the processing and transporting units, respectively. This division of the model is trivial, since in the models discussed in [16] there are no operations involving resources of both types. Each sub-model is then controlled to avoid deadlocks related to the specific sub-set of resources. Provided the processing resources required after a transporting operation are booked *before* the same transporting operation is started, the combination of the two DA controllers is sufficient to guarantee that the overall model is deadlock-free. This basic idea is extended in this paper to the more general framework of FMS. It is shown that two arbitrary sub-sets of resources of the modeled system can be used for the sub-model division, thus allowing for an extra degree-of-freedom that can be effectively used for optimization purposes. Finally, the assumption that operations using both types of resources are absent is relaxed.

Manuscript received March 1, 2005.

L. Piroddi is with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, 20133 Italy (phone: +39-022399-3556; fax: +39-022399-3412; e-mail: piroddi@elet.polimi.it).

L. Ferrarini is with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, 20133 Italy (e-mail: ferrarin@elet.polimi.it).

0-7803-9568-9/05/$20.00 ©2005 IEEE

II. Petri Net Modeling of Discrete Event Processes

The specification of the model of an FMS requires the definition of a recipe model, describing the sequencing and synchronization of the operations, as well as a resource allocation model, conditioning the execution of operations to the availability of specific resources. In the following, a fairly standard design approach will be briefly illustrated (see e.g. [12, 22]).

A. Modeling of recipes

A logical operation is an aggregated abstract activity that is performed in the process using system resources. It is generally modelled as a PN with one place (*operation place*), which can be interpreted as *Task in execution*. At least one *beginTask* transition and one *endTask* transition are present in the model, as in Fig. 1. However, depending on the connection structure between operations, the transitions in the pre-set and post-set of the operation place may be more than one.

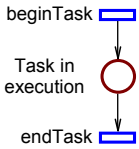

Fig. 1. A PN representing a single operation of the plant

Recipes are modelled by suitably connecting the individual operations involved. Fig. 2 lists the five elementary connection rules, which can be employed to define synchronization and sequencing between the individual operations. Generalization to more than two tasks and to more complex connections is straightforward [5].

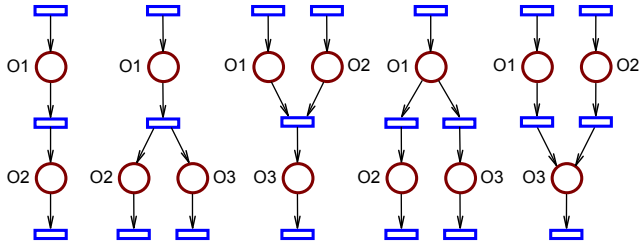

Fig. 2. Connection rules: a) sequential, b) concurrent, c) synchronization, d) alternative, e) end of alternative.

B. Modeling of the resource allocation mechanism

The concept of *resource* [2, 21] is used in the literature to represent the generic device needed to perform an operation. Each operation must use at least one resource. The basic model for resource allocation requires one *resource place* for each resource, initially marked with the resource availability, and suitably connected to the transitions in the pre-set and post-set of the operation places as in Fig. 3.a [2, 21]. This basic representation model can be easily extended to account for non elementary resource allocation mechanisms, such as resource retention between subsequent operations (Fig. 3.b), and resource sharing (Fig. 3.c).

A more complicated resource allocation model may be required to describe the concurrent/alternative usage of a single resource by different sequences of operations, as in join/split material transfers in batch processes [6].

In the following we will denote by N the recipe model, and by N_R the overall model, including resource allocation.

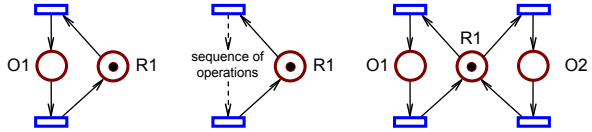

Fig. 3. Resource representations: a) basic resource allocation model, b) resource retention, c) resource sharing.

III. The Modular Deadlock Avoidance Approach

Given the reasonable assumption that the recipe model N is live, deadlocks can appear in the overall model N_R only because of shared resources, if one or more recipes are blocked in a *circular wait* condition [15], each waiting for a resource that is being used by the next recipe of the chain.

Let the resource set be partitioned in two sets $R = R_A \cup R_U$, where R_A groups resources that will be booked *before* actual use (anticipated booking) and R_U the remaining resources (un-anticipated booking), briefly referred to as *A-resources* and *U-resources*, respectively. Different criteria may be considered for this partition. For example, the resource set could be divided according to the type of resources, e.g. processing and transporting resources, or depending on the interaction of resources in the deadlock states, as can be pointed out by a structural analysis of the PN siphons (see the discussion in Section IV on the selection of convenient resource partitions). Deadlocks can be categorized as follows depending on the resources involved in the circular wait condition:

Definition 1 – A-, U- and M-deadlocks

With respect to the overall model N_R, and the partition $R = R_A \cup R_U$, deadlocks involving only A-resources (i.e. such that the associated minimal siphons contain only resource places belonging to R_A) are called A-deadlocks. Similarly, U-deadlocks involve only U-resources. The remaining *mixed* deadlocks, which involve both A- and U-resources, are called M-deadlocks. □

A. Model simplifications that preserve the basic structure of deadlocks

Some simple structural model simplifications will be used in the following to obtain the required sub-models. This section lists such operations and proves that though they may reduce the actual number of siphons in a PN model (thus greatly simplifying analysis and control design), they preserve the basic structure of deadlocks in terms of resources involved in circular wait conditions.

1) Elimination of resource places

The DA method is based on a suitable partition of the resources in two sub-sets, and two sub-models are accordingly derived eliminating one of the two resource sub-sets. Eliminating a sub-set of resources preserves all deadlocks involving only resources of the other sub-set (the corresponding siphons are unchanged).

2) Elimination of unconstrained operation places

In a correct model all operation places are constrained in their marking by the available units of the resources they employ. However, as a result of an elimination of resources, some places may turn out to be unbounded. Now, all operation places in a minimal siphon get empty in a deadlock state because they require unavailable tokens from empty resource places belonging to the same siphon. An unconstrained operation place does not use any resource, and therefore cannot be in any minimal siphon, so that its elimination from the model does not alter the set of minimal siphons of the net. More in detail, if an unconstrained operation place belonging to a siphon is unmarked, then all the transitions in its preset are disabled by way of previous operations which are not in execution (empty operation places which are also part of the siphon). Then there exists a smaller siphon not containing the unconstrained place.

3) Aggregation of concurrent operations using the same resources

Consider two concurrent operations (O_1 and O_2) employing the same resource R_1, as in Figure 4. An equivalent model can be obtained by grouping together O_1 and O_2 in a single logical operation O_{12}.

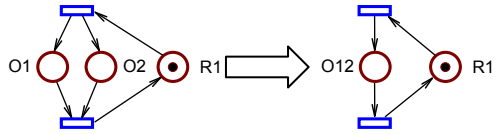

Fig. 4. Aggregation of concurrent operations.

4) Aggregation of sequential operations using the same resources

Consider a sequence of two operations (O_1 and O_2) employing a single resource R_1 (Figure 5). Assume that no allocation or deallocation of resources takes place at the intermediate transition. An equivalent model can be obtained by grouping together O_1 and O_2 in a single logical operation O_{12}. This simple model modification reduces the size of siphons in the PN model. Every siphon S of the original net which contains p_{O2} contains also p_{O1}. On the contrary, in the reduced model S will contain only p_{O12}.

Notice in fact that the two operation places p_{O1} and p_{O2} are either both marked or both unmarked, at any time, and that, when places p_{O1} and p_{O2} are marked [unmarked], then place p_{O12} is marked [unmarked]. This simple model modification reduces the number of siphons in the PN model: if there exists a siphon $S' = \{p_{O1}, \overline{P}\}$, where $\overline{P}$ is a suitable set of net places, then there exists also another siphon $S'' = \{p_{O2}, \overline{P}\}$, since $\bullet p_{O1} = \bullet p_{O2}$, and $p_{O1}\bullet = p_{O2}\bullet$. S' and S'' are not independent, since they always get unmarked together. The reduced model has a single siphon instead, $S = \{p_{O12}, \overline{P}\}$, which is unmarked when S' and S'' are unmarked in the original net. Therefore, both S' and S'' can be easily controlled by designing the DA control for S.

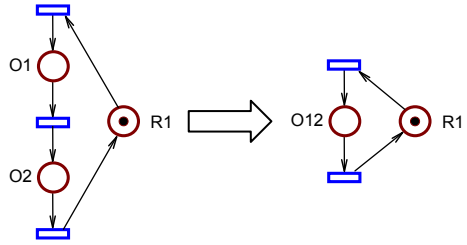

Fig. 5. Aggregation of sequential operations.

More in general, any transition not associated to allocation or deallocation of resources can be safely eliminated, suitably aggregating operation places of each individual sequence.

B. Anticipated booking of resources

In order to decouple the DA problem, the overall model N_R must be modified by suitably anticipating the allocation of A-resources as described in the following.

Definition 2 – U-sub-recipes

Remove from the recipe model N all operation places associated to operations using *only* A-resources, as well as any unconnected transitions resulting from this pruning process. The resulting unconnected sub-models of N are denoted U-sub-recipes. □

Definition 3 – Anticipated booking overall model

Modify the overall model N_R by substituting every arc directed from an A-resource to a transition of one U-sub-recipe with an arc directed from the same A-resource to the *initial* transition(s) of the U-sub-recipe. The resulting model is denoted N_R^{ab} and termed anticipated booking overall model (AB-model). □

Lemma 1 – The AB-model is free of M-deadlocks. □

Proof. Suppose, by contradiction, that there exists an M-deadlock in the AB-model. This implies that there exists a mutual wait condition in which both A- and U-resources are involved. More in detail, there exists at least one recipe using a U-resource which is waiting for an A-resource, which is in turn occupied by another recipe. But, since in the anticipated booking overall model U-resources are granted to start a U-sub-recipe only if the A-resources that are needed during or immediately after the U-sub-recipe are available (and in that case, they are booked), this event is impossible. □

Remark – Generally speaking, the anticipation of resource

allocation adds further constraints to the model evolution, but it does not necessarily represent a pejorative control solution with respect to maximally permissive DA methods.

Consider for example the simple PN model shown in Figure 6, representing a simple 3-operation FMS with 2 resources, R_1 and R_2. A minimal siphon $S = \{p_{O3}, p_{R1}, p_{R2}\}$ can be easily spotted in the system. To prevent the siphon from getting unmarked and causing deadlock in the system, a control place (p_C) can be added. As observed in Figure 6, the effect of the control place is equivalent to an anticipation of the allocation of resource R_1 at operation O_1.

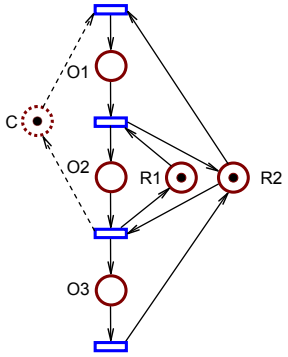

Fig. 6. PN model of a non-deadlock-free FMS (continuous line) and respective control (dashed line).

Notice finally that a solution to the deadlock problem in an FMS can always be found on the basis of the reallocation of shared resources. In the worst case one can allocate all the needed resources at the beginning of every production sequence. □

C. Control of A- and U-deadlocks

The design of the control sub-net for the remaining deadlocks of the anticipated booking overall model can be separated in two steps, each requiring the analysis of a suitable simplified PN.

Definition 4 – U-model

Let N_R^{ab} be the *AB*-model. Remove from N_R^{ab} all *A*-resource places. Eliminate all unconstrained operation places, as well as any unconnected transitions resulting from this pruning process. Compact the model by aggregating parallel or sequential operations using the same *U*-resources. The resulting sub-model is denoted *U*-model. □

The *U*-model can be interpreted as the *AB*-model in the case that *A*-resources are always available, and therefore describes exclusively the usage of *U*-resources in the *U*-sub-recipes. Therefore, the *U*-model preserves only the *U*-deadlocks of the *U*-sub-recipes.

Definition 5 – U-control

Let N_R^{ab} be the *AB*-model and N_U the corresponding *U*-model. Denote *U*-control a DA control sub-net N_{UC} for N_U, calculated e.g. by using the P-invariant based control method. □

Lemma 2 – The *AB*-model augmented with the *U*-control sub-net is free of *U*-deadlocks. □

Proof. The *U*-control ensures that if a *U*-sub-recipe is activated then it can be completed, in what pertains to the availability of *U*-resources (all the required *U*-resources will be available as the process evolves, and will all be made available again at its end). Now, suppose, by contradiction, that there exists a *U*-deadlock in the anticipated booking overall model with the *U*-control sub-net. This implies that there exists a mutual wait condition in which only *U*-resources are involved. In other words, there exists at least a recipe using a *U*-resource which is waiting for another *U*-resource, which is in turn occupied by another recipe. But *U*-resources are used only by the *U*-sub-recipes of the model, i.e. in the mutual wait condition there must be a blocked *U*-sub-recipe waiting for a *U*-resource held by another *U*-sub-recipe, which is forbidden by the *U*-control sub-net. □

Definition 6 – A-model

Let N_R^{ab} be the *AB*-model. Remove from N_R^{ab} all *U*-resource places. Eliminate also all unconstrained operation places and any unconnected transitions resulting from this pruning process. Compact the model by aggregating parallel or sequential operations using the same *A*-resources. The resulting model is denoted *A*-model. □

The *A*-model represents in compact form the anticipated allocation model for *A*-resources, and preserves the *A*-deadlocks of the anticipated booking overall model.

Definition 7 – A-control

Let N_R^{ab} be the *AB*-model and N_A the corresponding *A*-model. Denote *A*-control a DA control sub-net N_{AC} for N_A, calculated e.g. by using the P-invariant based control method. □

Lemma 3 – The *AB*-model augmented with the *A*-control sub-net is free of *A*-deadlocks. □

Proof. Since the *A*-model preserves all *A*-deadlocks of the *AB*-model, applying the *A*-control to the *AB*-model eliminates all its *A*-deadlocks. □

Theorem 1 – The overall model with anticipation of *A*-resources according to Def. 3 and augmented with the *A*- and *U*-control sub-nets is deadlock-free. □

Proof. The *AB*-model is free of *M*-deadlocks by Lemma 1. Augmenting it with the *A*- and *U*-control sub-nets eliminates the remaining *A*- and *U*-deadlocks, by Lemmas 2-3. The control places introduced by the *A*- and *U*-control sub-nets can be viewed as additional virtual *A*- and *U*-resources, and therefore cannot generate new *M*-deadlocks in the *AB*-model, thanks to the decoupling effect of the resource anticipation mechanism. □

IV. An Illustrative Example

Consider the following machining station (Fig. 7) where four machines (M_1, M_2, M_3 and M_4) are served by two robots (R_1 and R_2), and parts are transported on a conveyor belt (C). Two products are manufactured in the station, corresponding to two different routings: M_1-M_3-M_4 and M_4-M_2-M_1.

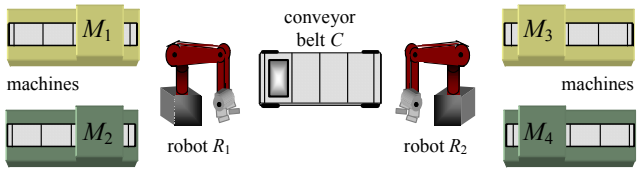

Fig. 7. FMS for the example.

A simple PN model of the FMS is shown in Fig. 8.

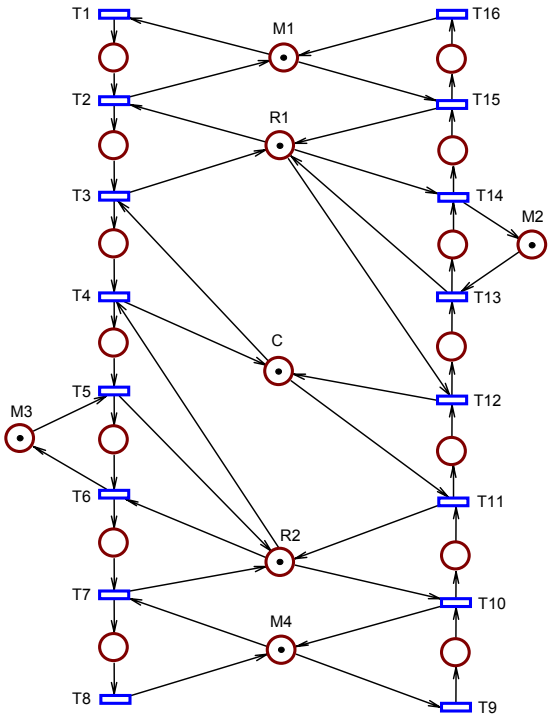

Fig. 8. Overall model for the FMS.

Though the model is apparently simple, consisting of two pure sequences using some shared resources, several deadlocks can occur. In particular, as many as 30 minimal siphons are obtained (excluding those that are supports of P-invariants). Directly applying the P-invariant based control approach would then result in 30 additional places, more than doubling the original model size. What is more, the controlled net would still be not deadlock-free because of new siphons involving the control places (which act as virtual resources), so that the control design should be iterated. Though convergence of the P-invariant based control method can be guaranteed (under certain conditions, such as net boundedness [8]), the awkwardness of the procedure is apparent. Some additional figures obtained by reachability analysis may clarify the picture: the model of Fig. 8 has 715 possible states, only 333 of which do not lead to one of the 27 deadlocks. Since each monitor of the P-invariant based controller typically prevents a single state, over 300 additional control places would probably be required to completely solve the problem.

Now, the first step of the proposed approach requires the partition of the resource set: 2^7–2 possible partitions can be envisaged. The size of the control sub-net and the permissivity degree of the control critically depend on the partition adopted. Also, the complexity of the A- and U-models resulting from the partition influence the number of necessary iterations to complete the P-invariant based control. A possible criterion for choosing a sound partition could be to maximize the number of M-deadlocks, since these are eliminated for free by simple anticipation of the A-resources.

Compare e.g. the 4 alternative partitions suggested in Table I. The first two are based on a subdivision of the resources in transporting and processing units, whereas the last two are based on the preceding criterion. It is apparent that the last two partitions lead to much more (though not maximally) permissive control solutions, with respect to the first two. However, the 3rd partition demands a high cost in terms of control design and size (3 iterations of the P-invariant based algorithm are required and the control sub-net consists of 10 places), so that the 4th one is selected as the best compromise.

TABLE I
COMPARISON OF POSSIBLE PARTITIONS OF THE RESOURCE SET

R_A ($R_U = R - R_A$)	M-DLs	Total states	Forb. states	Safe states	Alg. iter.	Control places
M_1, M_2, M_3, M_4	27	184	27	157	2	8
R_1, R_2, C	27	111	16	95	3	10
M_1, M_2, M_3, M_4, C	30	284	47	237	3	10
M_1, M_2, M_3, C	29	240	27	213	1	2

Setting $R_A = \{M_1, M_2, M_3, C\}$ and $R_U = R - R_A = \{M_4, R_1, R_2\}$, the AB-model and the A- and U-models are easily determined (Figs. 9-11). Analysis of the A- and U-models reveals the existence of one minimal siphon per model. The respective DA controllers are then computed with a single iteration (see Figs. 10-11). Applying the same A- and U-control sub-nets to the AB-model results in a deadlock-free model.

V. Conclusion

A modular approach for DA has been presented, which employs a resource anticipation mechanism to divide in two computationally much simpler problems the control design. The resource set is partitioned in two and, correspondingly, two sub-models are obtained.

It is shown that the respective DA controllers, together with the resource anticipation mechanism, guarantee that the resulting model is deadlock-free. The obtained controller is sub-optimal, but compact in size and easy to compute. Some criteria for optimal splitting of the resource set are suggested, though this is matter for further research. The method

can also be generalized to multiple resource sub-sets.

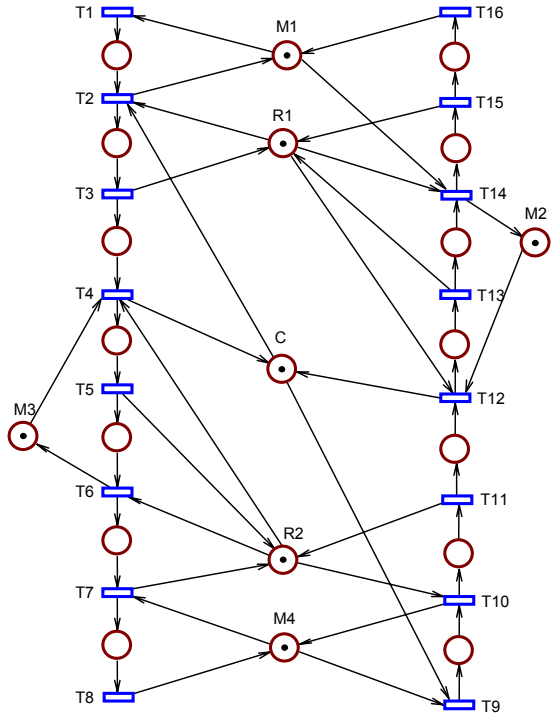

Fig. 9. *AB*-model for the FMS.

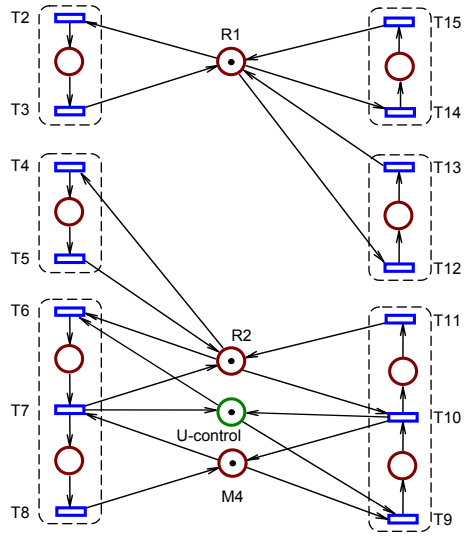

Fig. 10. *U*-model and *U*-control for the FMS. The *U*-sub-recipes are enclosed in dashed lines.

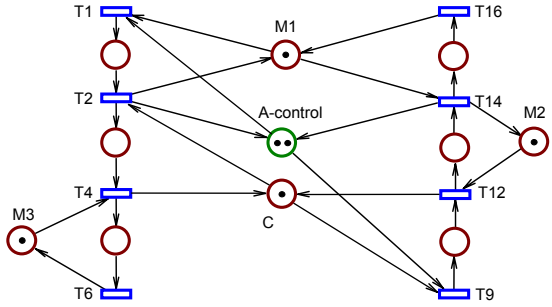

Fig. 11. *A*-model and *A*-control for the FMS.

REFERENCES

[1] Z. Banaszak, and B. H. Krogh, "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows," *IEEE Trans. on Robotics and Automation*, vol. 6, n. 6, pp. 724-734, 1990.

[2] J. Epzeleta, J.-M. Colom, and J. Martinez, "A PN Based Deadlock Prevention Policy for Flexible Manufacturing Systems," *IEEE Trans. on Robotics and Automation*, vol. 11, n. 2, p. 173, 1995.

[3] M. P. Fanti and M. C. Zhou, "Deadlock Control Methods in Automated Manufacturing Systems," *IEEE Trans. on Systems, Man, and Cybernetics – Part A*, vol. 34, n. 1, pp. 5-22, 2004.

[4] L. Ferrarini, and M. Maroni, "Deadlock Avoidance Control For Manufacturing Systems With Multiple Capacity Resources," *Int. J. of Advanced Manufacturing Technology*, *Special Issue on PNs and Manufacturing Systems*, vol. 16, n. 1, August 1998.

[5] L. Ferrarini, and L. Piroddi, "Modeling and control of transporting systems in batch processes with multiple aggregated resources," *IEEE Int. Symposium on Intelligent Control*, Mexico City, Mexico, September 2001, pp. 258-263.

[6] L. Ferrarini, and L. Piroddi, "Automatic Synthesis of Multiple Place Resource Models with Petri Nets," *American Control Conference*, Boston, USA, 2004, pp. 5096-5101.

[7] F.-S. Hsieh, and S.-C. Chang, "Dispatching-Driven Deadlock Avoidance Controller Synthesis for Flexible Manufacturing Systems," *IEEE Trans. on Robotics and Automation*, vol. 10, n. 2, p. 196, 1994.

[8] M. V. Iordache, J. O. Moody, and P. J. Antsaklis, "Synthesis of Deadlock Prevention Supervisors Using Petri Nets," *IEEE Trans. on Robotics and Automation*, vol. 18, n. 1, pp. 59-68, 2002.

[9] J. O. Moody, K. Yamalidou, M. D. Lemmon, and P. J. Antsaklis, "Feedback Control of PNs based on place invariants," *Proc. of the 33rd IEEE Conf. on Decision and Control*, vol. 3, Lake Buena Vista (FL), USA, 1994, pp. 3104-3109.

[10] T. Murata, "PNs: properties, analysis and application," *Proc. of the IEEE*, vol. 77, n. 4, pp. 541-580, 1989.

[11] J. Park, and S. A. Reveliotis, "Deadlock Avoidance in Sequential Resource Allocation Systems With Multiple Resource Acquisitions and Flexible Routings," *IEEE Trans. on Automatic Control*, vol. 46, n. 10, pp. 1572-1583, 2001.

[12] J. L. Peterson, *Petri net theory and the modeling of systems*, Prentice Hall International, 1981.

[13] S. E. Ramaswamy, and S. B. Joshi, "Deadlock-Free Schedules for Automated Manufacturing Workstations," *IEEE Trans. on Robotics and Automation*, vol. 12, n. 3, p. 391, 1996.

[14] S. A. Reveliotis, and P. M. Ferreira, "Deadlock Avoidance Policies for Automated Manufacturing Cells," *IEEE Trans. on Robotics and Automation*, vol. 12, n. 6, pp. 845-857, 1996.

[15] A. S. Tanenbaum, *Modern Operating Systems*, Prentice Hall International, 1992.

[16] M. Tittus and K. Åkesson, "Deadlock avoidance in batch processes," *14th IFAC World Congress*, Beijing, P. R. China, 1999, pp. 397-402.

[17] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using PN Models," *IEEE Trans. on Robotics and Automation*, vol. 6, n. 6, p. 713, 1990.

[18] R. A. Wysk, N.-S. Yang, and S. B. Joshi, "Resolution of Deadlocks in Flexible Manufacturing Systems: Avoidance and Recovery Approaches," *J. of Manufacturing Systems*, vol. 13, n. 2, p. 128, 1994.

[19] K.-Y. Xing, B.-S. Hu, and H.-X. Chen, "Deadlock Avoidance Policy for Petri-Net Modeling of Flexible Manufacturing Systems with Shared Resources," *IEEE Trans. on Automatic Control*, vol. 41, n. 2, p. 289, 1996.

[20] K. Yamalidou, J. O. Moody, M. D. Lemmon, and P. J. Antsaklis, "Feedback Control of PNs based on place invariants," *Automatica*, vol. 32, n. 1, pp. 15-28, 1996.

[21] M. C. Zhou, and F. DiCesare, "Petri Net Modeling of Buffers in Automated Manufacturing Systems," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 26, n. 1, pp. 157-164, 1996.

[22] M. C. Zhou, and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems. A Petri Net Approach*, World Scientific Publishing, 1998.