A Petri Net based approach for the Synthesis of Parts’ Controllers for Reconfigurable Manufacturing Systems

Eun Joo Lee, Armand Toguyéni, Nathalie Dangoumau
LAGIS – CNRS UMR 8146, Ecole centrale de Lille
Cité scientifique BP 48, 59651 Villeneuve d’ASCQ, France
(Tel: +33-3-20-33-54-55; E-mail: lee.eun.joo@ec-lille.fr, Armand.Toguyeni@ec-lille.fr, Nathalie.Dangoumau@ec-lille.fr)

Abstract: The goal of this work is to address the problem of the synthesis of Petri Nets controllers to design Reconfigurable Manufacturing Systems. Our approach is inspired by supervisory control but proposes to use Petri nets (PN) to model both the plant model and the user specifications. Since Ramadge and Wonham original works [1], many authors have investigated supervisory control problems based on automata formalism. However, finite automata present some drawbacks such as the difficulty to model parallelism, synchronization and resource sharing. The approaches based on this formalism are generally limited by the combinatorial explosion that occurs when attempting to model complex systems. On the other hand, Petri Net is a more powerful tool for modeling parallelization or synchronization, but its uses for synthesis has not been largely investigated. The method developed here is based on the Ghaﬀari’s Region Theory [2] to synthesis control places that are added to the original plant model in order to obtain a closed-loop PN that respect the user specifications. In this study one assumes that the system production is based on cyclic scheduling performed by sharing resources. More specifically the goal is to synthesis on-line resources allocators based on the scheduling of operations that is established off-line by the scheduling task.

Keywords: Petri net, Reconfigurable Manufacturing System, Cyclic scheduling, Resource sharing, Constrained Synchronous Reachability Graph.

1. INTRODUCTION

In the eighties, the concept of flexible manufacturing systems (FMS) was introduced to develop new systems of manufacturing production able to produce the small ones and average series of products. But today, the capacity of reconfiguration is become a major problem to improve the functioning of industrial processes. Indeed, actually a main objective is to adapt quickly the system in order to start a new production or to react to failures. This leads some authors to introduce the concept of reconfigurable manufacturing systems (RMS) as a final solution to maintain industrial productions in westerns countries [3, 4]. Such type of system is characterized by its reconfigurability capacity that is obtained both by the flexibility of the plant and the adaptability of its control software. These features can be ensured by introducing redundancies between the operations held by the plant resources. This lead to design control software that is modular, generic and whose parameters can be modifiable depending on the plant functioning mode.

This study concerns more specifically the design of parts’ controllers to allow a deterministic controlling that corresponds to off-line scheduling of the production. Parts’ controllers correspond to the integration of operating sequences and machines’ allocators. The second section defines the problematic and our study context. The third section introduces cycling scheduling. We assume that it is the type of scheduling applied to considered systems. The fourth section presents our approach of control places synthesis based on Ghaﬀari’s Region Theory [2]. The last section will concern the proposed method to design parts’ controllers.

2. STUDY CONTEXT AND PROBLEMATIC

To control a manufacturing system, the LAGIS’s approach can be summarized by the design of two main functions: sequential control and piloting control (Fig. 1).

Sequential control aims to define the sequence of order to send to the plant to make a finished product from raw parts. To reach this goal, one can decompose this function in two tasks corresponding to two viewpoints: the Coordination Graph of Transport System (CGTS hereafter) and the Operating Sequences [5]. They are both Petri nets (PN) based models. To take account of the plant flexibilities they integrate indeterminisms that are solved in real time by the piloting function. CGTS controls the transfers and the routing of part between the different resources that compose the plant. At this level, the problems are both to solve the constraint introduced by buffers placed between the machines and by the allocation of transport resources. For more details, the readers can see our previous publications [6, 7].

![Diagram of RMS Control Architecture](image_url)

Fig. 1. Hierarchical view of RMS Control Architecture

89-950038-5-5 98560/06/$10 © 2006 ICASE
Operating sequences are PN models that describe the operations to apply to transform raw parts in finished products. These operations are tasks performed by transformation or assembling machines. Because of the parallelism of parts’ production, some resources can be required simultaneously by several parts: it is a conflict situation. To solve conflict situations it is necessary to design resources’ allocators.

The second part of control architecture is the piloting function. The function of piloting consists both in defining the sequence an dates of raw parts entry in the plant and also in solving in real time the residual indeterminism of the models in sequential control. Our approach considers of course conflict situations but also the choice of variant alternatives in operating sequences models and part routing in the CGTS. This paper addresses the problem of parts’ controller design based on Petri net formalism. Our idea is to translate a cyclic scheduling in a set of control places that are added to initial operating sequences to transform them in deterministic controllers implementing exactly the scheduling frame.

3. PRESENTATION OF CYCLIC SCHEDULING

Cyclic scheduling is generally known to be suitable to the productions of big demand [8]. Indeed, these methods constitute a good approach to avoid the combinatorial complexity explosion due to the scheduling of all the operations while concentrating on the determination and the optimization of a repetitive frame (cycle). Recent works showed that they are also well adapted to small and medium demand by studying and optimizing associated transient states. Among these methods, we can quote the critical machine scheduling, K-cyclic scheduling and 1-cyclic scheduling: these approaches aim to respect the optimal cycle time while minimizing the Work in Progress (WIP). The 1-cyclic schedules are characterized by the fact that each machine does the same operations at the same dates in each cycle. In this paper we will focus on this scheduling type. In a pedagogic purpose of clarity, we will disregard transfer operations.

In order to illustrate our approach, we introduce an example from the literature (see [8]). Three machines \(\{M_i, i \in \{1, \ldots, 3\}\}\) compose the plant and allow making 2 types of parts OS1 and OS2. Each part is characterized by its operating sequence. An operating sequence is defined here by the sequence of operations and their respective duration (given in generic time unit or t.u. hereafter) on each machine.

- OS1: OP11 (M3, 2 t.u.), OP12 (M1, 3 t.u.), OP13 (M2, 2 t.u.);
- OS2: OP21 (M1, 1 t.u.), OP22 (M3, 2 t.u.).

The cyclic production consists in producing, in each cycle, 3 parts of OS1 and 2 of OS2. Hence, 5 parts are produced per cycle (see Fig. 2). Even if the three parts of OS1 are identical, it is necessary to identify them and to distinguish them. Indeed, since each part has a different output date, it is necessary to know which part leaves the production system to survey if production achieved corresponds to the planned one.

![Fig. 2. Operations precedence constraints](image)

The cyclic schedule gives the result of Fig. 3 for the plant steady state: the optimal cycle time is equal to 11 time units and the bottleneck machine (slowest machine: working 100% of time) is M1. The WIP used is optimal and equal to 5 (1 work-in-process by each part to be manufactured). The result is represented by a dual Gantt chart. Its first section shows the scheduling of the operations of each part (point of view of the parts). The second section illustrates the scheduling of the operations performed by each machine during the cycle (point of view of the machines). In section 5, these two viewpoints will be exploited to define the initial marking and the sequences of our Petri Net models.

![Fig. 3. Cyclic scheduling of the steady state](image)

4. SYNTHESIS OF PETRI NET CONTROLLER

4.1 Theory of the regions

The synthesis problem for nets consists in deciding whether a given automaton is isomorphic to the reachability graph of a net and then constructing it. This problem has been solved in the literature for various types of net ranging from elementary nets to Petri nets. Its objective here is to determine a pure PN
\((N, M_0)\), characterized by an incidence matrix \(C\) and having a set of transitions \(T\), such as his reachability graph is isomorphic to \(RG(N, M_0)\) (an automaton which arcs are labeled by the elements of \(T\)).

Ghaffari proposes a new adaptation of the theory of regions that enables to add control places to an initial PN controller in order to control forbidden behavior [2]. The Ghaffari’s approach is summarized by the following theorem.

**Theorem 1** [2]: There exist a PN, \((N, M_s)\) which reachable graph is \(RG(N, M_0)\), if there exists a set \(P_c\) of control places \(\{M_0(p_c), C(p_c, \cdot)\}\) as:

1. Each place \(p\) of \((N, M_s)\) satisfies the cycle equation
   \[(3.1) \text{ of } RG(N, M_0): \sum_{\sigma \in \Delta} C(p, \sigma)[T] = 0, \forall \sigma \in \Delta \]  
   \[(1)\]
2. Each place \(p\) of \((N, M_s)\) verifies the reachability equation
   \[(3.2) \text{ of } RG(N, M_0): \]  
   \[M_0(p) + C(p, \cdot)^T M = 0, \forall M \in \mathcal{R} \]  
   \[(2)\]
3. To each pair \((M, \tau)\) such that \(\tau\) does not fire from \(M\), it exists at least one control place \(p_c\) which satisfies the equation (3) of state separation,
   \[M(p_c) = M_0(p_c) + C(p_c, \cdot)^T M + C(p_c, \cdot) < 0 \]  
   \[(3)\]

**4.2 Determination of theory of the regions equations**

We propose the constrained synchronous reachability graph (CSRG) as a tool that helps to synthesize plant PN models and specification PN models [6, 7]. Specification PN models contain transitions that corresponds to some transitions of the plant PN models. To obtain the legal behavior of the plant that implements the user specifications our algorithm merges the transitions that are synchronized. In this context, CSRG results in a smaller size than the reachability graph obtained by a Cartesian product of the reachability graphs of the two types of models. To construct the CSRG, following two rules are essential.

**Rule 1**: A transition of the specification model can only be fired when it is enabled in the two models. 

**Rule 2**: An enabled transition of plant model must be inhibited if this transition exists in the two models and if it is not simultaneously enabled in the specification model.

Our goal is not really to construct the CSRG. We propose an algorithm that interprets the CSRG to give three sets that characterize the theory of the regions equations of a given plant and specification PN models.

**Algorithm: Extraction of theory of the regions equations**

\(AM\) is the set of authorized marking,

\(PM\) is the set of processed marking,

\(AS\) is the set of authorized sequences,

\(FS\) is the set of forbidden sequences of state-transitions,

\(CS\) is the set of Cycles.

\(LT\) is the set of the transitions in the PN models

\(P(N^p, M_0^p)\) is the plant PN model and \(S(N^s, M_0^s)\) is the specification PN model

The marking \(M_i\) is composed of the places of the different Petri Nets models: \(M_0 = \left[ M_0^p, M_0^s \right] \) is the initial marking

**Inputs:** \(M_0, LT, P(N^p, M_0^p), S(N^s, M_0^s)\)

**Outputs:** \(AS, FS, CS\)

**Step 1** \(AM \leftarrow \{M_0\}\);  
**Step 2** while \((AM \neq \emptyset)\) do,  
**Step 2.1** \(M_i = \text{next} \text{-marking of } (AM)\); \(AM \leftarrow AM \setminus \{M_i\}\)  
**Step 2.1.1** \(T_{SA} \leftarrow \text{transitions from } (M_i, LT, P(N^p, M_0^p), S(N^s, M_0^s))\);  
**Step 2.1.2** while \((T_{SA} = \emptyset)\) do,  
**Step 2.1.2.1** \(MT = \text{next successor in } (T_{SA})\); \(M_i = (M_0, M_i)\); \(T_{SA} \leftarrow T_{SA} \setminus \{MT\}\);  
**Step 2.1.2.2** \(T_i \leftarrow \text{transition } (MT)\);  
**Step 2.1.2.3** \(M_k \leftarrow \text{marking } (T_i)\);  
**Step 2.1.2.4** (*Applying rule 2*) if enabled in plant \((t_{ik})\) and not (enabled in specification \((t_{ik})\)) then  
**Step 2.1.2.4.1** \(\forall \sigma \in AS \text{ if } M_k > M_i\); \(FS \leftarrow FS \cup \{t_{ik}\}\);  
**Step 2.1.2.4.2** \(FM \leftarrow FM \cup \{M_k\}\); end if;  
else if enabled in plant \((t_{ik})\) then  
**Step 2.1.2.4.3** If not \((M_k \in PM)\) then  
**Step 2.1.2.4.3.1** \(\forall \sigma \in AS \text{ if } M_k > M_i\); \(AS \leftarrow AS \cup \{t_{ik}\}\);  
**Step 2.1.2.4.3.2** \(AM \leftarrow AM \cup \{M_k\}\); end if;  
else  
**Step 2.1.2.4.3.3** \(\forall \sigma \in AS \text{ if } M_k > M_i\); \(CS \leftarrow CS \cup \{t_{ik}\}\); end if;  
end if;  
end while;  
**Step 2.1.3** \(PM \leftarrow PM \cup \{M_k\}\); end while;  
end.

In this algorithm, the function ‘transition from’ is supposed to give the list of all transitions that start from the marking \(M_i\). In other words, its means that its function constructs incrementally the CSRG corresponding of the system PN models.
5. SYNTHESIS OF PARTS’ CONTROLLERS

Assuming that we have the operating sequences the problem is now to synthesize the resources allocators that achieve the cyclic scheduling (see Fig. 3). Our approach is based on generalized PN. So, the designer does not take account of time specifications given by the scheduling. He just has to express the relative order of each operation regarding the two viewpoints given by the cyclic scheduling.

To apply our synthesis approach, operating sequences are considered as the plant model. First, it is necessary to specify an initial marking of these operating sequences that corresponds to the WIP of each manufactured part. After that, the designer must constructs a PN that models the manufacturing order of each part, by a given machine. These PN models specify how each machine is allocated to parts. They are cyclic to reflect the cyclic nature of the scheduling. Their initial marking define the capacity of each machine and the status of the machine at the beginning of a production period. Their transitions are synchronized with transitions of the operating sequences because they model the same operation. To understand how we build a machine allocator model that implements a specific scheduling, it is important to notice here that an operating sequence is a part viewpoint instead machine allocator model which is a resource viewpoint.

Fig. 4. Products’ operating sequences with illustration of machining resource sharing

Fig. 4 gives the operating sequences of each type of part corresponding to the cyclic scheduling of Fig. 3. In this model, the places from P1 to P10 model the status of the different parts regarding the machining operations. The places P1, P3, P5, P7 and P9 model transfers operations. The machining operations modeled by the other places are represented by their name on the left of the place. The two operating sequences are connected by three places M1, M2 and M3 that model the machines. When a place Mi is marked, it means the corresponding machine (machine ‘i’) is free. The transitions model the beginning and the end of an operation executed by a machine. As an example, ‘t1’ is the beginning of manufacturing of OS1 by machine 3 (operation OP11) and ‘t2’ the end of this operation.

It is important that the initial marking of the operating sequences model the state of the parts at the initial time of each cycle. Let us consider the superior part of the Gantt diagram of Fig. 3. It defines that the first part W1 is supposed to have been already machined by machine 3 in the previous cycle. Its next operation is OP12 so place P3 is marked with one token. The two other parts OS1 (W2 and W3) has not yet started their operating sequence. So two token are placed in the place P1 to reflect their status. At the opposite, the two parts OS2 (W4 and W5) has already be processed by machine 1. W4 is currently machined by machine 3 so a token is put in place P8. W5 is waiting to be transformed by machine 3, so a token is placed in P7.

For pedagogic reasons we have reduced the complexity of this example by removing machine 2 because this machine is used only by operating sequence OS1. To simplify the calculations we have transformed the initial OS PN models with multiple token in so many OS PN of each type than the WIP of each parts. In consequence, our plant model is represented by Fig. 5.

Fig. 5. The plant model corresponding to the process given by Fig. 3.

To specify machines’ allocators, the designer must consider the inferior part of the Gantt diagram (machines viewpoint) as the specifications of the expected behavior of the plant. So these informal specifications must be translated in PN models. The
transformation is systematic. The first place of each allocator models the machine in an idle mode waiting the request of the first operation scheduled on it. Thus, considering how each operation is modeled in dual operating sequences PN, the remainder is an alternation of operation frame (beginning transition, a place, and an end transition) and an idle state to wait the request of the next part (Fig. 6).

To define their initial marking, one assumes here that the capacity of each machine is one. At the initial time only machine 3 is working. So a token is initially placed in the place ‘machine3.works(W4)’ (Fig. 6).

![Fig. 6 Specification models of machines’ allocators](image)

The next step consists in synthesis the operating sequences model and the machine allocators’ model. This synthesis is based on the theory of the regions which allows determining the control places to add to initial operating sequences [6]. To perform this synthesis, we have developed a computer tool that proceeds in two stages. The first stage consists in establishing a set of equations that are given by the theory of the regions. This stage implements the algorithm given in section 4. It is implemented in PetriMaker version 3.1 a free PN software developed by the LISAC at Anger (in France). This software is developed in Visual C++ and we have added new features such as ‘Get the Equations’ that gives the equations of the theory of the regions (Fig. 7).

![Fig. 7 Reachability equations, events separations equations and cycle equation](image)

The second stage of the synthesis consists in using a solver like CPLEX to solve the system of equations obtained at the first stage. This solving gives a set a control places (with their initial marking) that must be added in initial operating sequences PN model.

Finally the synthesis is the solution of Fig. 8 (on the last page) that is a deterministic controller that implements exactly the sequence of operations defines by the Gantt chart in Fig. 3.

But because of the plant model we select, our synthesis algorithm works here like a merging algorithm of PN sharing common transitions. In fact, our algorithm is more powerful. Let us consider the case of the production of two parts OS1 and one part OS2. Let us assume that the sequence on machine 1 is OS1-OS1-OS2 and the sequence on machine 2 is OS1-OS2-OS1. All the parts are first machined by machine 1 and then by machine 2. For OS1, we select a PN model with two tokens modeling WIP. The specification of the sequence of parts OS1 on a machine implies the presence of several occurrences of the same transition in the PN model. In this case our algorithm cannot works as a merging algorithm and it gives the result illustrated by Fig. 9.

![Fig. 9 Control model corresponding to a given cyclic scheduling](image)
6. CONCLUSION

We desire that our work contribute to the approaches to synthesize DES controllers with respect to user specifications based on the supervisory approach using the Petri net formalism. In this context we propose a systematic approach to design a deterministic controller that implement a cyclic scheduling of a plant owning shared resources. The main interest of this approach is to allow adapting part controllers depending on the production objective. The writing of resources' allocators is enough systematic to be automated. We plan in the future to use technique of transformation models to translate automatically the results of cyclic scheduling in specification PN models.

We have developed a PN tool that assists a designer to build the PN controllers corresponding to the closed-loop behavior that guarantees user specifications. When this tool will be automated to work directly with cyclic scheduling tool it will become possible to reconfigure on-line manufacturing systems based on their production objectives.

REFERENCES


Fig. 8. The PN controller that synthesizes the close-loop behavior of the plant given by Fig. 3.