Logic control law design for automated manufacturing systems

Henry Sébastien, Éric Zamaï, Mireille Jacomino

Logic Control law design for Automated Manufacturing Systems

Sébastien HENRY[a], Eric Zamaï[b], Mireille JACOMINO[b]

[a]*University of Lyon, Claude Bernard Lyon 1, IUT Lyon 1, DISP Laboratory, Villeurbanne, 69627, France*
[b]*Grenoble-INP, G-SCOP Laboratory, 46 avenue Felix Viallet, 38031 Grenoble Cedex 1, France*

Abstract

To respond rapidly to the highly volatile market, the reconfigurable manufacturing systems (RMS) have brought forward challenging issues. First of all there is a need to build a formal model of a manufacturing configuration. Second it has to be rather easy to derive the models associated to the manufacturing configuration changes (Reconfiguration) from such an initial model. An off-line method of rapid design of an optimal logic control law (configuration) based on Petri net (PN) is presented in this paper. From a controlled system modeling point of view, the main characteristics of the level 1 of the CIM architecture are depicted. Subsequently, the formal tool used in the Automated Planning field is extended to provide a controlled system model. The concept of operation is structured in order to introduce the behavioral properties of the operations. A four-step method is then proposed to design a logic control law that satisfies several goals: reduction of the lead time, satisfaction of the work orders objectives, minimization of the time cycle. Finally, the proposed design method is illustrated on a manufacturing cell.

Keywords:
Logic control law, Design, Manufacturing systems, Programmable Logic Controllers, Petri Nets

1. Introduction

Manufacturers are facing nowadays a market with frequent fluctuations in product demand. To be competitive in such a context the flexibility and reactivity of manufacturing systems have to be properly involved in order to guaranty the productivity and quality of the production system. To reach this objective, not only the reactivity of the higher levels of the control system (Entreprise Resource Planning and Manufacturing Execution System) but also the real time control level (PLC control level) has to be improved. This paper is located at this control level. PLC control level of an automated manufacturing system is very crucial to ensure the execution of desired manufacturing behaviors by coordinating various machines and operating a large number of processes. Each time a new product is requested or each time a corrective maintenance is operated, a new logic control law has to be designed in order to match the current manufacturing abilities. Then the model reconfiguration problematic should be solved to make change the models along with manufacturing configuration changes (Li et al., 2009). Today, from an industrial point of view, practitioners in industry pursuit more intuitionally standardized modelling tools, e.g., IDEF family (IEEE-SA Standard Board, 1998), GEMMA -Study Guide of the Starting and Stopping Modes (of an automated process), and one or several languages of the IEC-611313-3 norm (IEC-61131-3, 1993), to design new logic control code when required. The developer uses the implicit model of the physical system that is his own knowledge about the system. Generally speaking this knowledge syntheses the resources abilities and their possible effects on the product flow as well as the physical constraints of the system to prevent disruptive events such as collisions between resources or degradation of products. The constraints assure the integrity of people and goods. Depending on the objectives defined by the scheduling level, the developer designs a logic control law that satisfies the physical constraints and the objectives.

By the end, the following drawbacks of this handmade method can be mentioned:

- The optimality of the resulting logic control law with regard to a given criteria cannot be ensured. It directly depends on the expertise level of the developer.
- Theoretical tools classically adopted by academia, such as automata (Linz, 2000), Petri nets (PNs) (Murata, 1989), statecharts (Harel, 1987), and finite state machines (E.W. Endsley & Tilbury, 2006) to check behavioral properties have usually not been used due to difficulties for industry to accept such tools. The languages traditionally used to code the logic control law (Sequential Function Chart (SFC) or Ladder Diagram languages (IEC-61131-3, 1993)) do not lend themselves to analytic verification during the design step. However, the behavioral properties like liveness, boundedness (or safety), and reversibility, are very important to operating and control of production systems.
- The global design time of a logic control law is the sum of the three-part time: the design, the verification/validation steps and the tuning phase. Even if today, the time of the last two parts can be reduced with the 3D virtual environment, the global design time is very long. So, the reduction of the lead time of logic control law, and so production time, is not guaranteed.

- When a PLC program developer leaves the company, all the knowledge on the physical system which is not formalized and which is known only by the PLC program developer is lost if a knowledge management is not properly spread into the company.
- With this time constraint, a logic control law is only designed off-line during the design or re-design of the manufacturing system. Then, on-line, a reaction to an event that requires a new logic control law cannot be answered (Herroelen & Leus, 2002), (Davenport & Beck, 2000).

To settle these challenging problems of logic control code changes and so to contribute at least, not only to reduction of the production lead time at the real time control level, but also to give means to test behavioral properties of the resulting logic control code, a generic logic control law design approach for industry is proposed. One of the originalities of the proposed approach is to fill the gap between academia and industry, starting from a guided and intuitionally modelling tool (issued from Automatic Planning field of research), dedicated industrial practitioners, and a sub-optimal resulting control code based on formal tool e.g. PNs, allowing behavioral properties processing.

The paper is organized as follows. The main approaches for logic control laws design are depicted in section 2. The proposed methodology is submitted in Section 3. The proposed control model is described as well as the operation concept on which it relies on. The formal definition and derived properties of operation are given. The design algorithm is displayed in Section 4. Section 5 is devoted to an illustrative case study. Finally, conclusions are given and further research is commented on.

2. Logic control law design in CIM level 1

2.1. Design need

Elementary functions (move a product from one point to another, stop a product, etc.) of an automated system are performed by functional chains. A functional chain includes all the components required to achieve this function: an actuator with its effector and elements associated to interface with the control system (pre-actuators, sensors).

From the functionnal chain level to the highest control level, the control system of a production system is generally divided into five levels as proposed by the CIM (CIM, 1989), (ISA-S95, 2000), (Trentesaux, 2009). Control equipment machines such programmable logic controllers (PLC) correspond to the CIM level 1. To design of logic control laws for the PLC, the level 1 is divided into two sub-levels to separate the constraints: functional chain and coordination.

At functional chain level, a logic control law is designed to provide services to coordination such as: extend cylinder, retract cylinder, start conveyor, change the conveyor speed, etc. A logic control law at a functional chain level has then to include all the technological constraints required to offer the service. These technological constraints depend on the type of actuator (single or double effect cylinder, synchronous motor, etc.), the type of pre-actuators (bistable or monostable solenoid valve, starter motor, etc) and information provided by the sensors.

At coordination level, the execution of logic control law changes the state of the products through the services offered by the functional chains. At this level, the controlled system is composed of functional chains and products. The addressed constraints are interactions between functional chains and products. The optimization of time cycle, that is the main criterion, is generally obtained by simultaneously performing several services.

At the coordination level, our approach aims to design off-line a logic control law in one IEC 61131-3 langage (Structure Functional Chart, Ladder Diagram, etc) to modify the state of functional chains and products from an initial state to a final state by optimizing the time cycle. The design approach is based on a controlled system model, an objective function and a design algorithm. The controlled system model must represent the evolutions of the functionnal chains with their effects on the products. The objective to achieve is the time cycle, the initial state and the final state of both the product and the functionnal chains. From these data, the design of a logic control law in coordination level boils down to determine the set of operations to be performed with their precedence constraints.

In literature, various models of the controlled system exist according to the initial data of the problem and the desired result. In order to establish the foundations of the controlled system model needed to design a control law in coordination level, the next section presents the existing design approaches.

2.2. Design approaches in the literature

We have identified as design approach every method that provides a law to control partially or entirely a controlled system. They are used in different CIM levels or in other application fields than manufacturing. Finally, from this analysis of available methods, the design principle proposed in this paper is defined to answer some drawbacks highlighted by the analysis.

2.2.1. Scheduling

From the set of operations to be performed with their constraints, the scheduling problem consists in finding the chronological order of operations which is the best feasible calendar with respect to a given performance criteria (Chretienne et al., 1997). In the manufacturing field, the scheduling is used in the CIM level 3. An operation represents the use of a resource (operator or machine) to produce a service. The set of operations to be performed is determined from manufacturing bill of materials (mBOM) of each product and the quantities of products to manufacture. This knowledge of all operations to be performed is the main difference with the design problem of logic control law in coordination level.

2.2.2. Supervisor synthesis

The supervisor synthesis is based on the Ramadge and Wonham theory (Ramadge & Wonham, 1987). From a state model of controlled system and the set of prohibited states, the output called supervisor is the set of prohibited events according to the

state of the controlled system. The supervisor observes the state of the controlled system and limits its evolutions by prohibiting some events (Charbonnier et al., 1999). From the Ramadge and Wonham theory, other approaches (Lee & Lee, 2002), (Qiu et al., 2003) exist that are used in the CIM level 2. This observer role of the supervisor which prohibits some events is the main difference with the design problem of the logic control law whose execution imposes evolution on the controlled system.

2.2.3. Automated Control Code Synthesis

This approach proposed by Holloway in (Holloway et al., 2000) is based on the condition systems that are closely related to Interpreted Petri Nets. A component model representing a functional chain is described with this formalism as well as the associated task blocks. For each output condition of the component model and from any marking of the component model, a task block defines all the control laws to reach the marking for which the output condition is true. In a condition system, the output conditions result from marked places. The effects on the product flow are not modeled in the component model which represents the state space of a component. Then, a sequence of product states is initially given to have the expected effects on the product flow. This approach designs a control code which defines for an actuator the evolution of the PLC outputs to reach successively the different component states. According to the authors, the approach is limited to functionnal chain level with technological constraints. The need to provide a sequence of states of the product is the main difference with the design problem of the logic control law.

2.2.4. Automated Planning

Automated planning is a control law design method devoted to path and motion planning for mobile systems like trucks or robots that can be easily extended to assembling operations (Valente & E.Carpanzano, 2011). Automated Planning consists in defining an organized collection of actions whose global effect on mobile systems or handling parts satisfies a given objective (Ghallab et al., 2004). The controlled system model consists in the set of all available actions, a description of the state of the controlled system, and some objectives defined by the final state of the mobile system or parts. The planning algorithm consists in finding the subset and string of available actions to reach the final state satisfying the constraints all along the planning. The automated planning is a sequence of actions that can depend on the current execution context. No parallelism mechanism is involved in the automated planning. Then the model and the algorithms used in automated planning cannot be directly used to design control laws in manufacturing systems in which several parts have to be processed in the same time. But the controlled system model based on the available actions is very interesting because it takes into account constraints on the controlled system state using the pre-constraint notion. Moreover the action model includes the impacts of the action's execution on the controlled system. Furthermore, with the action concept, the state space of the controlled system is not explicitly built. The result as a sequence without parallelism is the main difference with the design problem of the logic control law.

2.2.5. Analysis

The previous submitted formal approaches do not give a satisfactory solution (Zaytoon, 2002) to design a logic control law for CIM level 1. Depending on the initial data and results of the problem defined in paragraph 2.1, the table 1 summarizes the partial adequacy of the four approaches to design a logic control law in coordination level.

	Input				Set of operations to be performed	Output	
	Model		Objective				
	State evolutions	Interaction constraints	Initial and final states	Optimization criterion		Explicit control law	Parallelism
Scheduling		X		X	input	X	X
Supervisor synthesis	X	X					X
Code synthesis	X				input	X	X
Automated planning	X	X	X		output	X	

Table 1: Analysis of available design approaches.

In view of these elements, especially the outputs, the best approach with reagard to the need expressed in paragraph 2.1 is the automated planning. Indeed, this approach allows one to determine all the operations to be performed to achieve a final state of the controlled system. However, optimization of a criterion is not considered by this approach that generates a control law without parallelism. The design approach that is introduced in this paper is based on the automatic planning.

3. Proposed Controlled System Model

In the previous section, all requirements for the automated control law design are defined. The first requirement is the definition of a controlled system model adapted to the design of logic control laws. Then the set of information contained in this model must be specified. It connects directly with both the control law features, and the constraints satisfied by the controlled system evolutions imposed by a control law.

3.1. Constraint Levels Satisfied by a Control Law

The basic function of a manufacturing system (defined by the norm NF X50-100) can be expressed as increasing the value of the product flow. This basic function is achieved by performing effects on the product flow. The set of these effects modifies a product from its input state to its output state. An effect on the product flow is the result of the conversion by a functional chain

of a request emitted by the control system. A functional chain is generally made up of pre-actuator(s), an actuator, and an effector, as in a Fig. 1. In the proposed approach, an effector is always considered to be confused with an actuator. Depending on the desired effects on the product flow and with the notion of functional chains, the PLC program developer uses several points of view to synthesize a control law.

In the manufacturing systems, a control law is generally implemented in a Programmable Logic Controller (PLC). In this case, the requests are emitted by the PLC outputs depending on a control law run by the PLC. Then, a control law implemented in a PLC has a PLC output point of view. But a control law is never directly designed manually with this point of view. To design such a control law, the set of constraints (of the objectives and the controlled system) is taken into account progressively. Indeed, a PLC program developer begins by defining a control law from a product flow point of view. This control law defines the evolutions of the product flow state to modify it from its input state to its output state. The functional-chain constraints are progressively taken into account by designing a control law from an actuator point of view, then from a pre-actuator(s) point of view, and finally from a PLC output point of view. The components of a functional chain are generally standard. And in this case, they are chosen in a library. Therefore, when starting with a control law with an actuator point of view, designing a control law with a pre-actuator point of view and finally with a PLC output point of view, the new constraints taken into account are not specific to the manufacturing system. These new constraints depend only on the functional-chain technology like single or double cylinder action for the actuator, and one valve with three positions or two valves with two positions for the pre-actuator. Finally, all the specific constraints of the manufacturing system are taken in a control law from an actuator point of view into account. Thus, the most difficult work of a PLC program developer is to design a control law from an actuator point of view. It is the aim of the proposed approach.

The control law design is mainly based on the controlled system model. To model the controlled system, the correct point of view must be determined. The controlled system model must have a product flow point of view to be compatible with the objectives. But with only this point of view, the actuator evolutions without effect on the product flow cannot be found. Finally, to design a control law from the actuator point of view, the controlled system model must have a double point of view: actuators and product flow. And the model must give the links between both points of view. If an operation models the actuator evolutions and possible associated product flow evolutions, then the links defining the causal relations between two types of evolutions will be modeled.

This double point of view to model the controlled system is also required to know the executable operations for a state of the actuators and the product flow.

To achieve its basic function, the manufacturing system must control the product flow evolutions. But its control system does not drive directly the product flow evolutions; it drives the resource evolutions. Thus, we propose to represent the controlled system model by the set of the product flow evolutions, the resource evolutions and the links between the product flow and resource evolutions. With these links and from product flow evolutions respecting the objectives, the required resources will be deduced. Then a logic control law will be automatically designed. To define the link between the evolutions of resources and the product flow, the product flow and resource evolutions are modeled in the same operation. The operation structuring is based on the model proposed in automated planning field (Sandewall & Ronnquist, 1986) (Ghallab et al., 2004) and used by (Klein, 1999), (Castillo et al., 2000), (Aylett, 2001) in the manufacturing and batch-process field.

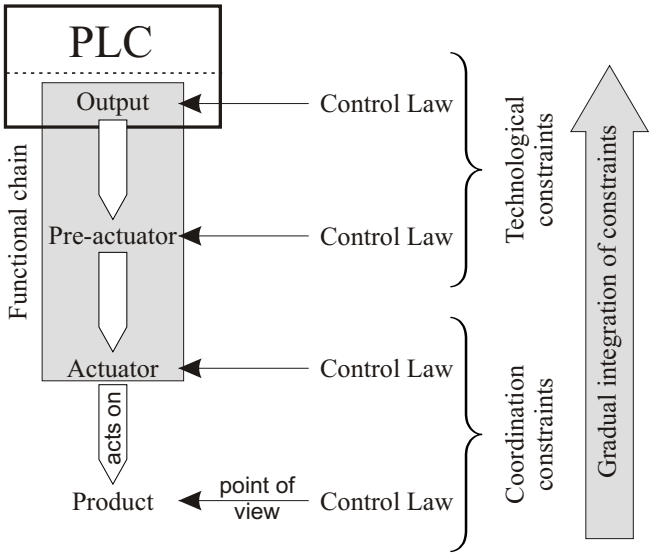

Figure 1: Functional chain and the different control laws designed by a PLC program developer.

3.2. Formal definition of an operation

Before formalizing the operation behavior, the required notations are defined.

3.2.1. The Automaton Notations

The controlled system is made up of all the resources and the product flow. The state of a resource or a product is defined by state variables (sv). A state q of the controlled system is defined by the value of each sv. The set of states of the Controlled System Model is denoted Q_{CSM}. We assume the following partition: $Q_{CSM} = Q_A \cup Q_F$, where Q_A and Q_F are the sets of Authorized and Forbidden states, respectively. From the control law design point of view in the SMC context (Combacau et al., 2000), the controlled system evolves in accordance with the occurrence of events: "start operation i" ,"end operation i" and "expected external events". The state evolution of the resources and the product flow may be seen as a four-tuple deterministic automaton: $CS = (Q_{CSM}, \Sigma, \delta, q_0)$ where: Σ is the alphabet of events defined above; $\delta : Q_{CSM} \times \Sigma \rightarrow Q_{CSM}$ is a partial transition function; and q_0 is the initial state. When the designer begins the modeling of the controlled system, no operation exists. Then, Q_A and Q_F are empty, and there is no event, and no δ.

3.2.2. The Operation Notations

First, the notation of the different sub-behaviors is submitted. Then, the notation of an operation behavior is presented. Finally the definition of the sub-behavior structure is given.

An operation i will be denoted O_i. The set of the operations is $O = \{O_i\}_{i\in[0,M]}$. An operation is made up of two kinds of sub-behaviors (see Fig. 2):

- One basic sub-behavior bb_i which defines the effect on the resource with the associated constraints.
- The extra sub-behaviors. An extra sub-behavior, denoted $eb_{i,j}$, defines an effect on the product flow with the associated constraints. The $eb_{i,j}$ number is not limited, so $j \in [0, N]$.

The set of the extra behaviors of O_i is

$$EB_i = \{eb_{i,j}\}_{j\in[0,N]}$$

When an operation is run from a state, the effects defined by bb_i and $eb_{i,j}$ are obtained simultaneously. The description of bb_i and $eb_{i,j}$ is given in the end of this part.

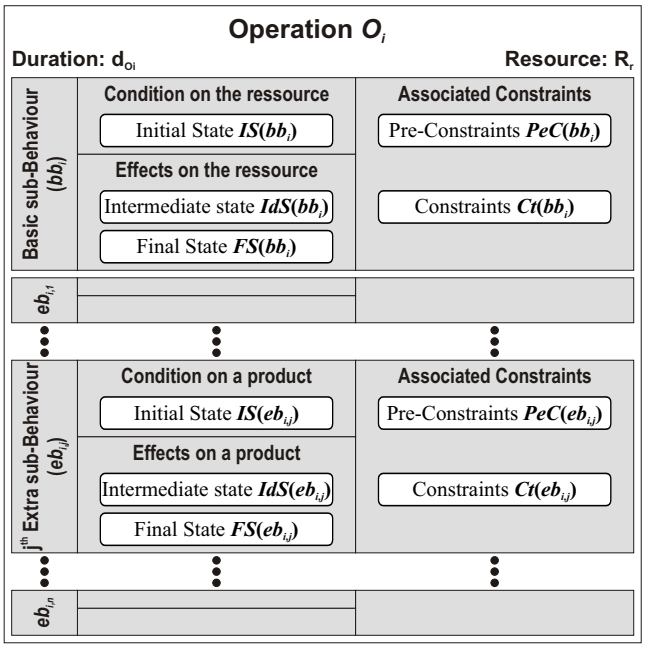

Figure 2: The components of the operations.

Depending on the controlled system state from which O_i is run, all the extra sub-behaviors are not obtained. However, two extra sub-behaviors can be not compatible, e.g. they can never be obtained simultaneously for all the controlled system states. For the above reasons, it is necessary to consider the sub-sets of EB_i, denoted $EB_{i,k}$. The number of $EB_{i,k}$ with p $eb_{i,j}$ elements is the combination of N elements taken p at a time (^NC_p). If all the extra sub-behaviors are compatible, the maximum number of $EB_{i,k}$ is equal to:

$$^NC_0 +^N C_1 + \ldots +^N C_N = 2^N$$

$EB_{i,k}$ is defined by:

$$EB_{i,k} = \{eb_{i,j}/j \in J_k \wedge k \in [0, 2^N - 1]\}, EB_{i,k} \subset EB_i$$

To have a concise notation to refer to an operation behavior, the k index value refers to a behavior. The binary value of k index is considered and each bit represents an extra sub-behavior. If the bit equals one, then the associated extra sub-behavior is included in the operation behavior. So, from the k index, the numbers of extra sub-behaviors included are formally defined in the J_k set by:

$$J_k = \{x/(k = \sum_{y=0}^{N-1} c_y 2^y / c_y \in [0,1] \wedge y \in \mathbb{N}) \wedge c_y = 1 \Rightarrow x = y+1\}$$

When an operation is run, the basic sub-behavior is always obtained. Finally, a behavior of an O_i operation is made up of the basic sub-behavior and a sub-set of EB_i. It is denoted $B_{i,k}$ and it is defined by:

$$B_{i,k} = \{bb_i\} \cup EB_{i,k}$$

The set of the O_i behaviors is denoted $B_i = \{B_{i,k}\}$. If all the extra sub-behaviors are compatible, the number of $B_{i,k}$ is equal to the cardinality of $EB_{i,k}$.

$$\text{Card}(B_i) = \text{Card}(EB_{i,k}) = \sum_{p=0}^{N} {}^NC_p = 2^N$$

For instance, the extra sub-behaviors of the EC1 operation (O_{EC1}) are:

$$EB_{EC1} = \{eb_{EC1,1}, eb_{EC1,2}\}$$

Then $k \in [0, 2^2 - 1]$ and the set of J_k is:

$$J_0 = \{\emptyset\},\ J_1 = \{1\},\ J_2 = \{2\},\ J_3 = \{1, 2\}$$

Finally, the set of the $EC1$ behaviors is:

- $B_{EC1,0} = \{bb_{EC1}\}$, no extra sub-behavior is simultaneously obtained with the basic sub-behavior,
- $B_{EC1,1} = \{bb_{EC1}, eb_{EC1,1}\}$, extra sub-behavior 1 is simultaneously obtained with the basic sub-behavior,
- $B_{EC1,2} = \{bb_{EC1}, eb_{EC1,2}\}$, extra sub-behavior 2 is simultaneously obtained with the basic sub-behavior,
- $B_{EC1,3} = \{bb_{EC1}, eb_{EC1,1}, eb_{EC1,2}\}$, the extra sub-behaviors 1 and 2 are incompatible, thus the $B_{EC1,3}$ behavior does not exist.

It is still impossible to define the operation behavior without giving details on the sub-behaviors (the basic sub-behavior (bb_i) and the extra sub-behaviors ($eb_{i,j}$)). The structure of the sub-behaviors is generic (see Fig. 2). They are made up of:

- The sub-behavior effect on the controlled system. The intermediate state (IdS) defines the effect that results from the "O_i start" event occurrence. After the "O_i start" event occurrence, $IdS(x)$ gives the value of each one of sv on which the sub-behavior has an effect. And the final state $FS(x)$ defines the effect that results from the "O_i end" event occurrence. After this event occurrence, $IF()$ gives the value of each one of sv on which the sub-behavior has an effect. The effects ($IdS(bb_i)$, $FS(bb_i)$) of the basic sub-behavior (bb_i) can be only on the sv of the Rr resource running the operation. And the effects ($IdS(eb_{i,j})$ and $FS(eb_{i,j})$) of an extra sub-behavior ($eb_{i,j}$) can be only on the sv of the product flow.

- The condition on the initial state. For a state q, if the condition (IS) of a sub-behavior is true, then the "start O_i" event occurrence will cause the effects defined by the sub-behavior. The condition is specified by a propositional formula with an elementary proposition defined by the value of a state variable. The elementary proposition can be expressed with the sv of the resources or the product flow. And the sv used to specify the sub-behavior effects must be used in the condition specification. The conditions of the basic sub-behavior (bb_i) and an extra sub-behavior ($eb_{i,j}$) are denoted $IS(bb_i)$ and $IS(eb_{i,j})$, respectively.

- The associated constraints. For the controlled system evolution resulting from the effects of a sub-behavior, they insure the respect of the security and environmental constraints to avoid harmful consequences. The pre-constraints (PeC) and the constraints (Ct) are specified by a propositional formula with an elementary proposition defined by the value of a state variable. The sv used here must not exist in IS, and IdS of the sub-behavior.

3.3. The Behavior of an Operation

The operation behavior $B_{i,k}$ is defined when the operation is the only one to be run. For a state $q \in Q_{CSM}$, the effects of the operation behavior are defined by the effect of the basic sub-behavior bb_i and the simultaneously obtained extra sub-behaviors $\{eb_{i,j}/\, j \in J_k\}$. From a state q after the "O_i start" event occurrence, the effect of a sub-behavior is obtained while respecting the security and environmental constraints, if the condition is true and the associated pre-constraints are satisfied. And from a state q after the "O_i start" event occurrence, the effect of an extra sub-behavior is not obtained if the condition is false.

Definition 1. *The set $Q_{I(B_{i,k})}$ of the initial $B_{i,k}$ states is the set of the states from which the "start of $B_{i,k}$" event occurrence is authorized. Formally, this set is represented by:*

$$Q_{I(B_{i,k})} = \{q \in Q_{CSM}/IS(bb_i) \wedge PeC(bb_i) \bigwedge_{j\in J_k}[IS(eb_{i,j}) \wedge PeC(eb_{i,j})] \bigwedge_{j\in\overline{J_k}} \neg IS(eb_{i,j}) = TRUE\}$$

Definition 2. *The set $Q_{Id(B_{i,k})}$ of intermediate $B_{i,k}$ states is the set of the states in which the controlled system can be after the "start of $B_{i,k}$" event occurrence and from which the "end of $B_{i,k}$" event occurrence is authorized. Formally, this set is represented by:*

$$Q_{Id(B_{i,k})} = \{q \in Q_{CSM}/IdS(bb_i) \wedge Ct(bb_i) \bigwedge_{j\in J_k}[IdS(eb_{i,j}) \wedge Ct(eb_{i,j})] \bigwedge_{j\in\overline{J_k}} \neg IdS(eb_{i,j}) = TRUE\}$$

Definition 3. *The set $Q_{F(B_{i,k})}$ of the final $B_{i,k}$ states is the set of the states in which the controlled system can be after the "end of $B_{i,k}$" event occurrence. Formally, this set is represented by:*

$$Q_{F(B_{i,k})} = \{q \in Q_{CSM}/FS(bb_i) \wedge PoC(bb_i) \bigwedge_{j\in J_k}[FS(eb_{i,j}) \wedge PoC(eb_{i,j})] \bigwedge_{j\in\overline{J_k}} \neg FS(eb_{i,j}) = TRUE\}$$

Definition 4. *From a state $q \in Q_{I(B_{i,k})}$ and after the occurrence of "start of $B_{i,k}$" event, denoted ($s_{B_{i,k}}$), the controlled system is in a state q'. The sv values of this state are the sv values of the state q which are modified by $IdS(bb_i)$ and $IdS(eb_{i,j})$ for $j \in J_k$. The partial transition function is defined by $q' = \delta(s_{B_{i,k}}, q)$. From the state q' and after the occurrence of "end of $B_{i,k}$" event, denoted ($e_{B_{i,k}}$), the controlled system is in a state q". The sv values of this state are the sv values of the state q which are modified by $FS(bb_i)$ and $FS(eb_{i,j})$ for $j \in J_k$. The partial transition function is defined by q" $= \delta(e_{B_{i,k}}, q)$. From a state $q \in Q_{I(B_{i,k})}$, the effect of the $B_{i,k}$ behavior on the controlled system can be represented as below:*

$$q \xrightarrow{s_{B_{i,k}}} q' \xrightarrow{e_{B_{i,k}}} q"$$

3.4. Three Types of Operations

In the preliminaries, we have highlighted two sets of operations: with or without effects on the product. This classification of operations is not sufficiently accurate towards the product flow state. Some state variables of the product flow result from the product specifications like geometric forms, etc. And other state variables of the product flow result from resources and their position in the manufacturing system. According to the state variables modified by an operation running, we have identified three kinds of operations, presented in Fig. 3.

4. Design algorithm

At the coordination level of the CIM architecture (Trentesaux, 2009), this section presents the proposed method to automatically design, for a single product, a control law able to satisfy one demand provided by the scheduling level.

The global structure of the proposed algorithm is directly driven by the operations classification presented above. For each kind of operations, we formulate processes as three optimal path searching problems are associated with (Lacomme

processing operation	move operation	preparation operation		
modifies	**cannot modify**	**cannot modify**	**SVP S**	State Variables of Product flow product Specifications
modifies	*modifies*	**cannot modify**	**SVP S**	State Variables of Product flow product Specifications
modifies	*modifies*	*modifies*	**SVR**	State Variables of Resources

Figure 3: The three types of operations.

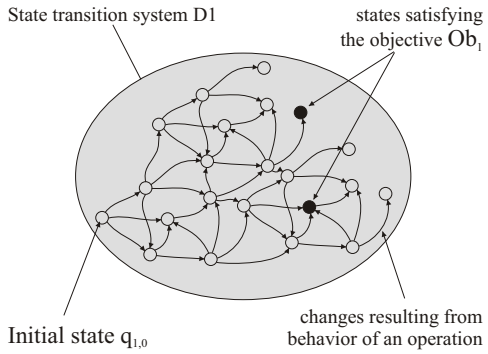

Figure 4: Reachable states space resulting from the processing of the product.

et al., 2003). At each step, an optimal path is provided including constraints depending first on the product processing, second on the product moving and third on the preparation of the functional chain.

By the end, a fourth step is added to optimize the cycle time of the sequence, studying the available parallelization between operations. As a result, we generate a control sequence in Petri Net formalism that can be directly translated into one of the IEC 1131-3 language, as Ladder Diagram (Lee et al., 2004) or SFCs that are a subset of the more complex Petri net.

4.1. Step 1: Product Processing

4.1.1. Principle

Here, it is proposed to focus on the physical states of the product to look for a path able to transform it from an initial state to the final one corresponding to the demand (objectives).

So, only operations with behavior that have an effect on variables states related to the physical state (form, color, etc) of products are used.

Let us define the state q_1 that represents the controlled system and its environment in the physical condition of the product belonging to the set Q_1. $Q_1 = \prod_{x \in VE_1} V_x$ where V_x represents all values of the state variable x.

A state $q_1 \in Q_1$ is defined by $q_1 = \{(x = c) | x \in VE_1\}$, where $c \in V_x$.

The states space considered in this first step describes the changes in the physical state of the product. It is a state transition system noted D_1 defined by a triple $D_1 = (Q_1, B_1, \delta_1)$ with Q_1 the set of states defined above, B_1 the set of behaviors transformation resulting from all operations and δ_1 the transition function of $Q_1 \times B_1$ in Q_1.

For an initial state $Q_{1.0}$, describing the physical state of a product just inserted in the factory process, and an objective (demand) Ob_1 defined by one (or more) specification on the physical state of the product, the first step is to find an optimal path in D_1 from the initial state $q_{1,0} \in Q_1$ to a target state $q_1 \in Q_1$ such as Ob_1 is satisfied (see Fig. 4).

To only build the useful part of the states space limited to product processing and to reduce the time process required to generate this space, the reachable states from the initial state $Q_{1.0}$ for a given objective Ob_1 are only considered. D_1 is defined by a quintuple $D_1 = (Q_{A1}, B_1, \delta_1, q_{1,0}, Ob_1)$ with $Q_{A1} \subseteq Q_1$ the restricted set of reachable states from $Q_{1.0}$ for the objective Ob_1.

4.1.2. Building of states space

To build the only reachable states space from an initial state $Q_{1.0}$, the proposed algorithm uses a greedy procedure (Lacomme et al., 2003) which generates new states to visit.

So, two sets of states are updated:

- the set of states to visit, noted Q_v,
- the set of processed states, noted Q_{A1}.

The principle of the algorithm is then as follows:

- Determining all processing behaviors from a state of the set of states to be visited. This state is then removed from this set of states and next added to the set of processed states;
- Calculate the status achieved through the application of each such behavior;
- If the reached state is a new state, it is added to the set of states to be visited.

Function ReachableStates($q_{1,0}$) : **System state transitions**
  $Q_v = \{q_{1,0}\}$
  $Q_{A1} = \{\emptyset\}$
  $\delta(q_1, b_1)$ est vide
  While ($Q_v \neq \{\emptyset\}$) **do**
    $B_{processing} = B_1(q_1)$ tel $q_1 \in Q_v$
    $Q_v = Q_v \setminus \{q_1\}$
    $Q_{A1} = Q_{A1} \cup \{q_1\}$
    While ($B_1(q_1) \neq \{\emptyset\}$) **do**
      $q"_1 = \delta(q_1, b_1)$ tel que $b_1 \in B_1(q_1)$
      $B_{processing} = B_{processing} \setminus \{b_1\}$
      If ($q"_1 \notin Q_{A1}$ et $q"_1 \notin Q_v$) **then**
        $Q_v = Q_v \cup \{q"_1\}$
      end If
    done
  done
  return $D_1 = (Q_{A1}, B_1, \delta_1, q_{1,0})$;
End

Algorithm 1: Algorithm to build the states space reachable by transforming the product from an initial state $Q_{1.0} \in Q_1$.

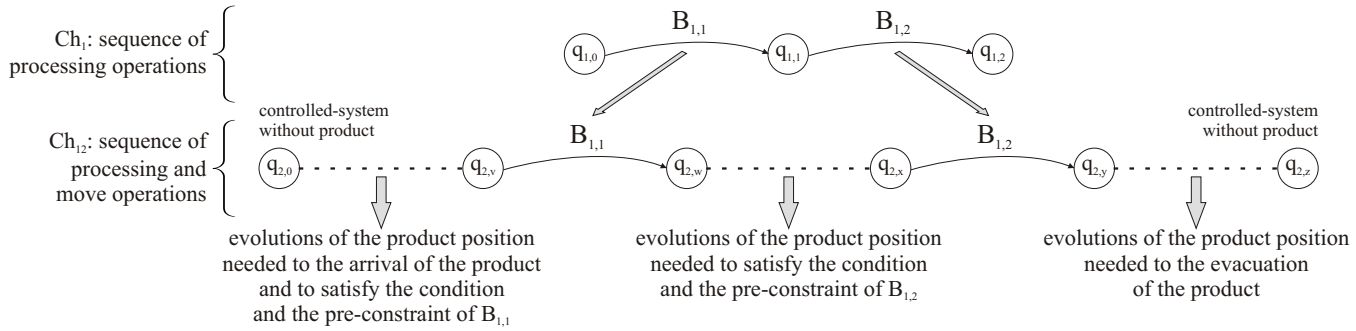

Figure 6: Principle of step 2.

The algorithm ends when the set of states to be visited, Q_v, is empty. The set of reachable states is then given by the processed states. As a result, the state transition system $D_1 = (Q_{A1}, B_1, \delta_1, q_{1,0})$ is built (see Fig. 5).

From the initial, the optimal path to reach the objective is look for in the reachable state space.

4.1.3. Optimal Path

Considering a criterion, the search process to find an optimal path to reach a goal state from an initial state is a classical graph theory problem (Lacomme et al., 2003). Among the many existing solutions and considering that the weight of arcs are positive or equal to zero, the use of Dijkstra's algorithm (Dijkstra, 1959) is submitted. It is applied to the system state transitions D_1 as a graph g_1. The vertices correspond to states of the system transitions and arcs to transitions between states. The initial state $Q_{1.0}$ becomes the top $s_{1.0}$ vertices and objectives $S_{1,Ob}$ correspond to objective states $Q_{1,Ob}$. Weighted arcs are defined from the characteristic of the operations (i.e. duration, see § 2) required to evaluate the optimization criterion.

The optimal path of processing product, noted Ch_1 is obtained by associating the vertices g_1 and the states transition system D_1.

At the end of this first step, only sequenced operations able to transform the physical states (form, color, ...) of the product according to the demand are submitted. Thus, the second step, presented in the next section, aims to act on the product location, to drive it from a processing operation to another one.

4.2. Step 2: Move the product

Consideration of changes in the position of the product aims firstly to satisfy the scheduling level request taking arrival and departure of the product in the manufacturing system, and secondly to meet conditions and constraints on the intermediates product's location with the path constructed above, Ch_1 into account.

Because of the significant similarities with step 1 proposed previously, the presentation of step 2 is voluntary reduced.

In the first step, only one single sequence is proposed, from the initial physical state of the product to the final one. In the Step 2, we have to build as many sequences as arcs between two vertices (processing operation) of Ch_1 (see Fig. 6).

The integration of these sequences to Ch_1 results in the generation of a new path denoted CH_{12}.

The constructive principle of such sequences shows a fundamental difference in the definition of the initial state $Q_{1.0}$ and objectives Ob_1 directly issued from the scheduling level demand. Here these two states noted $Q_{2,i,0}$ and $ob_{2,i}$ are directly derived from the application with the reachable states space noted $D_{2,i}$ (see Fig. 6).

Beyond this particular knowledge on the initial state and objectives, finding a path only representing changes of the location of the product, denoted $CH_{2,i}$ differs from the above algorithm on the following four points:

- Here, only move operations are considered,
- Only the location of the product evolves. Thus, a state q_2 of a path $CH_{2,i}$ belongs to the set defined by Q_2, where $Q_2 = \prod_{x \in VE_1 \cup VE_2} V_x$ with V_x corresponding to all the values of the variable state x. A state $q_2 \in Q_2$ is defined by $Q_2 = \prod_{x \in VE_1 \cup VE_2} V_x$, where $c \in V_x$,
- The path of processing and move operations, CH_{12} is built gradually with each new resulting sequences ($CH_{2,i}$),
- The complexity of the reachable states space in which are searched after the product's location is limited by restrictive conditions different from those used in step 1.

4.3. Step 3: Preparation of functional chains

The meaning of this step is to build sequences, noted $CH_{3,i}$, representing functional chains that do not affect the product. The integration of these sequences in the path CH_{12} will be called CH_{123}.

The corresponding algorithm for generating the path CH_{123} is quite similar to the one used to generate CH_{12} ; so it is not detailed here. This algorithm uses a function *OptimalPath* that built a specific path, denoted $CH_{3,i}$, limited to the changes of the functional-chain state. This path differs slightly from the sequence $CH_{2,i}$ looking for the evolution of the position of the product:

- Only operations affecting the state of the functional chains are considered. Then, a state q_3 of the path $CH_{3,i}$ belongs to the set $Q_3 = \prod_{x \in VE_1 \cup VE_2 \cup VE_3} V_x$ with V_x the set of values

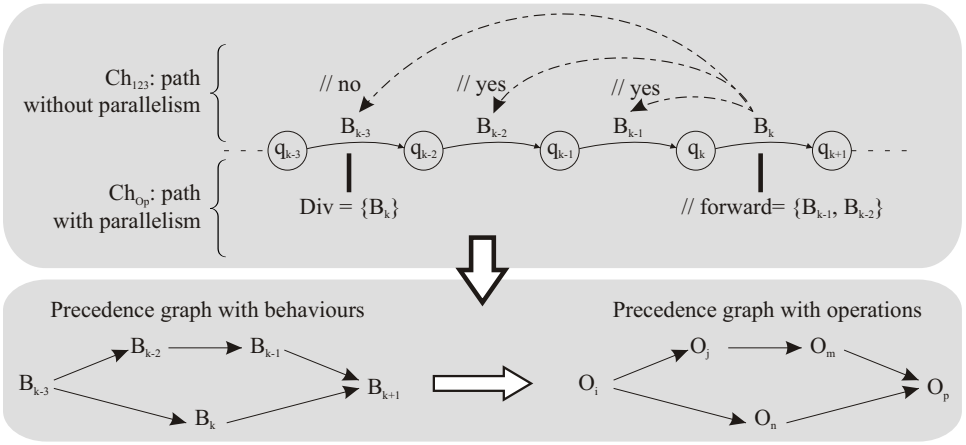

Figure 7: CH_{123} optimization

of the state variable x. A state $q_3 \in Q_3$ is defined by $q_3 = \{(x = c)|x \in VE_1 \cup VE_2 \cup VE_2\}$, where $c \in V_x$;

- Restrictive conditions specific to this third step limit the complexity of the reachable states space.

The search for the shortest path is performed in the same way as in steps 1 and 2. Obviously, the states space in which the path is searched is different.

4.4. Step 4: Optimization

The found path CH_{123} is not always optimal. Indeed, the cycle time of the control law corresponding to the sequence of operations specified by this path can be reduced by introducing parallelism between operations.

Let us consider a full path CH_{123} composed of N transitions which are each labeled with the following behaviors (B_k, $k \in [1, N]$). These behaviors transform the product, alter its position, or prepare functional chains.

Adding to the path CH_{123} information on parallel implementation leads to a path, called optimal control law, from a cycle time point of view. It is denoted CH_{Op}.

Both lists (*// forward* and *div*) of a B_k behavior is completed as follows:

list *// forward*. If the behaviors B_k and B_{k-1} can be executed in parallel, then the behavior B_{k-1} is added to the list of parallels authorized before B_k. We proceed in a same way for B_k and B_{k-i} (with i $\geq$ 2), until to find a behavior of an operation whose parallel execution is prohibited with B_k. In this case, B_k is then added to the list denoted *div*.

list *div*. For example, in Fig. 7 let us suppose that the behaviors B_k and B_{k-3} cannot be executed in parallel. B_k is then added to the list to be executed after B_{k-3}.

A full path CH_{123} has been designed for a product, then it is not possible to reverse the order of two operations whose effects would be different from the specified one by the full path CH_{123}.

5. Case study

The presentation of the case study is based on the principle of the control law design presented in Figure 4.1.1. After the description of the controlled system used for the case study, this section presents the model of the controlled system, the initial state and the objective. The control law provided by the design algorithm is then given.

5.1. Controlled system description

The controlled system used for the case study is based on the loading system (see Fig. 8) of an automated system. This system is dedicated to the assembly of camshafts, denoted products. A rotating storage with four places is used to receive up to six different kinds of products. The products are identified by a weight identification system. Once a product has been identified, a central conveyor drives it to a sorting device. A robot takes the different products to assemble them. A worker is in charge of filling the rotating storage and emptying the assembly station. The points A, B, C and D are fixed. So, if there is a product in A and if the rotating storage begins to turn clockwise, the product will be between A and B, then in B, then between B and C, and so on.

The next section presents the model of the loading system.

5.2. Controlled system model

The loading system model is made up of ten operations:

- **E**xtend **C**ylinder **1** from its retracted position (EC1) (see fig. 9),
- **R**etract **C**ylinder **1** from its extended position (RC1),
- **E**xtend **C**ylinder **2** from its retracted position (EC2),
- **R**etract **C**ylinder **2** from its extended position (RC2),
- **R**otate the rotating **S**torage **C**lockwise from an indexed position to the next (RSC),
- **R**otate the rotating **S**torage **C**ounter**C**lockwise from an indexed position to the next (RSCC),

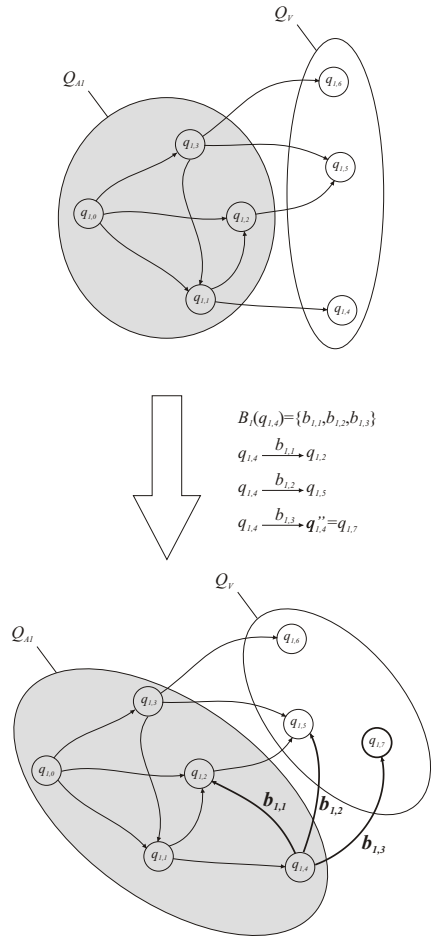

Figure 5: Design of the reachable states by processing the product from an initial state $q_{1.0} \in Q_1$.

- **I**dentify the **P**roduct on the weight identification system (IP),
- **D**etect the presence of a **P**roduct in **A** (DPA),
- **D**rop a **P**roduct in **C** by the operator (DPC),
- **E**vacuate a **P**roduct from **F** by conveyor (EPF) (see fig.10).

The EC1 operation (extend cylinder 1 from its retracted position) includes a basic sub-behavior and two extra sub-behaviors. The basic sub-behavior describes on the one hand the evolution of the resource (actuator 1) on which the operation is based and on the other hand constraints to be satisfied to ensure the integrity of the system. If the condition on the cylinder 1 is not satisfied, the operation cannot be executed. If the constraints are not satisfied, the operation execution is possible but the loading system or the products risk to be damaged. The constraints guarantee to avoid collisions between cylinder 1 and other elements (cylinder 2 or products). When the cylinder 1 is extended, the two extra sub-behaviors describe the possible effects on products according to the product state before the operation execution. The difference between condition and constraint is the same as in the basic sub-behavior.

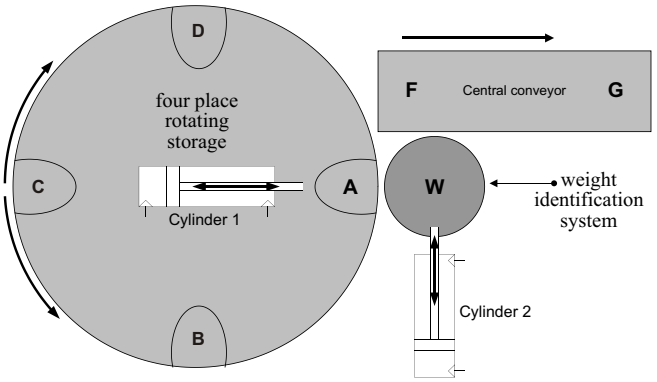

Figure 8: Loading system of automated system.

Extend Cylinder 1 from its retracted position (EC1)

Duration: 3s — Resource: C1

Basic sub-Behaviour	Condition on the resource: C1 retracted position	Associated constraints: C2 retracted position ∧ No P bet. A & B ∧ No P bet. A & D
	Effect on the resource: C1 intermediate position / C1 extended position	C2 retracted position ∧ No P bet. A & B ∧ No P bet. A & D
1st Extra sub-Behaviour	Condition on the product: P in A	RS indexed position ∧ RS null speed ∧ No P bet. A & W ∧ No P in W
	Effect on the product: P bet. A & W / P in W	RS indexed position ∧ RS null speed ∧ No P in W
2nd Extra sub-Behaviour	Condition on the product: P bet. A & W	RS indexed position ∧ RS null speed ∧ No P in W
	Effect on the product: P bet. A & W / P in W	RS indexed position ∧ RS null speed ∧ No P in W

Figure 9: "Extend Cylinder 1 from its retracted position" operation.

For the other operations, there are between zero and four extra sub-behaviors. For instance, the cylinder 1 retraction cannot have any effect on a product and therefore the operation RC1 has no extra sub-behavior. In contrast, the rotating storage holds up to four products and may have an effect simultaneously on these four products. Thus, the RSC and RSCC operations have four sub-behaviors each.

The DPC and EPF operations have no basic sub-behavior. For the DPC operation, the operator drops the product in C on the rotating storage when he wants. The operator can drop or remove the products in C. He does not inform the control system about his actions. And the product presence (or absence) in the rotating storage is known only when the product arrives in A. So in C, the presence of a product is uncertain. The operator is outside of the controlled system, the control system cannot run the DPC operation. In addition, the state of the operator is not necessary to write constraints or conditions for other operations, it is unnecessary to model the state and its evolution in

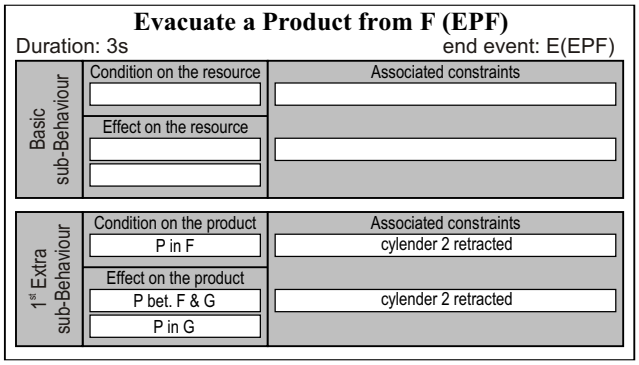

Figure 10: "Evacuate a product from F" operation.

the basic sub-behavior. For this operation performed by a resource outside of the controlled system, only information on its occurrence and its effect on the product is modeled in the extra sub-behavior.

Similarly for the EPF operation, the conveyor that acts on the product is outside of the controlled system. The state of the conveyor is not needed for writing requirements and constraints for other operations, the operation EPF has no basic-sub-behavior, see fig. 10.

With this controlled system model, the controlled system state space is not explicitly represented. The problem of the state space explosion is avoided. In addition to the controlled system model, the design algorithm also needs to know the initial state of the controlled system and the objective that are detailed in the next section.

5.3. Initial State and Objectives

The values of all variables of the controlled system are defined in the initial state. SVP∈S variables define the entry state of products into the loading system. For this system, the entry state is characterized by the single variable "Type". All the values of the variable "Type" are: unknown, 1-type to 6-type. For SVP∈S variables, the initial state is characterized by Type=unknown. For SVP∉S variables, the values are: no product in the controlled system. For SVR variables, the values are: C1 and C2 in the retracted position, the rotating storage in the indexed position with a null speed.

For the objective, the value of all variables is not necessarily defined. For the proposed example, the objective is to reach a state without product through a state with a product identified in F. This objective is defined only by the values of some variables. The optimized criterion is the cycle time.

The results of the algorithm that uses operations to design a control law according to the initial state and the objective are presented later in the next section.

5.4. Design algorithm

This section presents the main results of the four steps of the algorithm described in section 4.1.1.

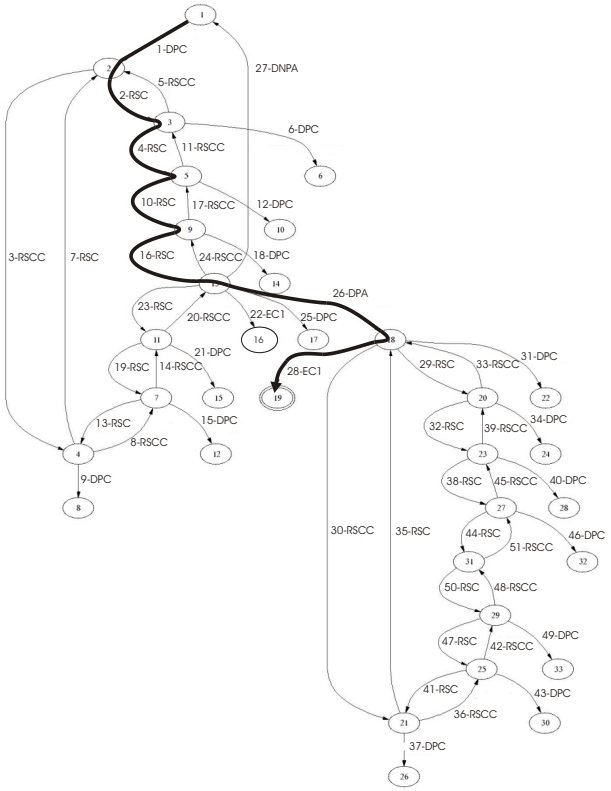

Figure 12: Reachable states space from initial state to the state allowing the processing operation.

5.4.1. Step 1: sequence with processing operations

The only processing operation provided by the loading system is the product identification (IP). Indeed, for the control system, this operation changes the features of the product. The features are known according to the value of variable *Type*. In the initial state, the features of the product are unknown. After the product identification, product features have changed state for the control system because they are known.

The Step 1 result is a sequence with a single operation whose execution modifies the values of SVP∈S variables in the initial state to the values defined by the objective. This sequence is very simple, the first step is no more detailed before presenting the second step.

5.4.2. Step 2: adding moving operations

Step 2 adds moving operations to the sequence generated in step 1 in order to satisfy the conditions and constraints on the product position of the processing operations. For the loading system, this step generates two sequences of moving operations. To allow the execution of the processing IP operation, the first sequence is designed to put a product on the weight identification system. After the IP operation, the second sequence is designed to put the controlled system in its initial state without product to generate a cyclic control law.

To design the first sequence, the problem is characterized by the initial state defined in paragraph 5.3 but limited to all the variables SVP. The objective of the first sequence is to sat-

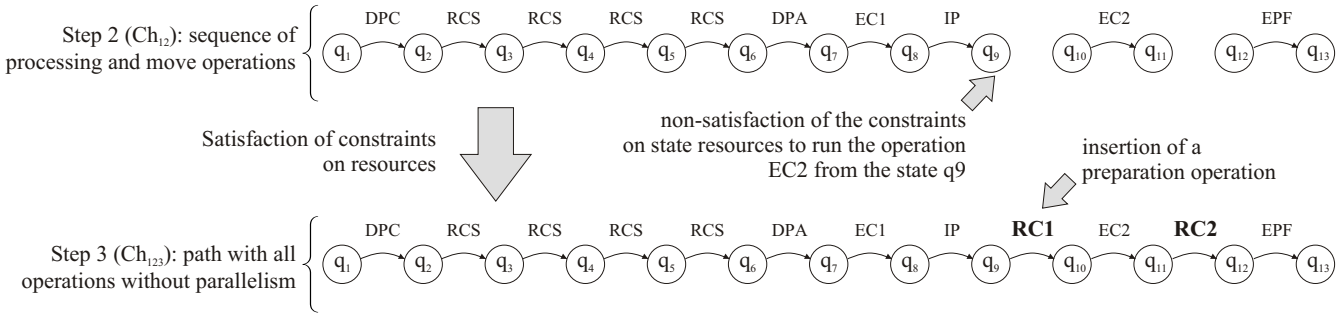

Figure 11: Preparation operations added to the sequence of move and processing operations.

isfy the conditions and constraints of IP operation on variables SVP∉S (one product on the identification system). As indicated in the presentation of the algorithm in section 4, the reachable state space is generated with the loading system model from the initial state before looking for the shortest path between the initial state and a state satisfying the objective. The shortest path is represented by a bold line in Figure 12 whose state 1 is the initial state.

To limit the size of the reachable state space, the number of products in the system is limited to one for this step 2. For the loading system, the number of products increases with the DPE operation when the operator drops a product in C. Apart from the states that respect the objective, this limitation is characterized by states reached with the DPE operation from which there is no transition for leave(see figure 12).

Of course, there is a risk of combinatorial explosion. For instance, the number of states and transitions with one product in the system is lower than 100 and and for three products the number of states exceed 3000 and the number of transition is closed to 8000.

From the graph, a sequence of moving operations is obtained with the operations belonging to the shortest path in figure 12. This sequence is added before the processing IP operation determined in step 1. The same principle is applied to design a sequence of moving operations added after the IP operation. After the assembly of these sequences, the result of Step 2 is a sequence of processing and moving operations shown in Fig. 11 in which preparation operations are added in step 3.

5.4.3. Step 3: adding preparation operations

The objective is to satisfy the conditions and constraints on resources for processing and moving operations. The principle of Step 3 is to add preparation operations in the sequence generated in step 2. For the loading system, the constraint on the state of the actuator 1 which must be returned to run the EC2 operation (Extend Cylinder 2) is not satisfied after the IP operation. To satisfy this constraint, a sequence of preparation operations is designed with the same principle as in steps 1 and 2. The preparation RC1 operation (retract cylinder 1) is added from the state q_9 after IP operation (see fig. 11).

Finally, step 3 leads to add the RC1 operation and the RC2 operation (Retract Cylinder 2). The sequence designed after the step 3 can reach the objective but it is not optimal in terms of cycle time. Operations that can be performed simultaneously have to be identified.

5.4.4. Step 4: parallelism

The Step 4 of the algorithm aims at finding authorized parallelism between operations in the sequence generated in step 3. From the principle of step 4 shown in Figure 7, the RC1 operation can be performed simultaneously with the IP operation. However the RC1 and EC1 operations cannot be performed simultaneously because they require the same resource, the cylinder 1. The parallelism between RC1 and IP corresponds first to perform the RC1 and IP operations after the EC1 operation and second to perform RC2 only after the end of the IP and RC1 operations. This information of AND divergence after EC1 operation and of AND convergence before EC2 operation is contained in both lists *// forward* and *div* which will translate the control law into a Petri Net or a language for PLC as SFC language.

Unlike these languages, the translation into a precedence graph of Figure 13 does not allow execution of the control law by a digital system (PLC, industrial computer) but it only allows to illustrate simply the result of Step 4. The precedence graph formalism is poorer in information than the state model with lists *// forward* and *div*. Indeed, the precedence graph does not contain information on the state of the controlled system after the execution of each operation. But, this information is required by the subsequent steps of the algorithm in the case of a multi-product problem. This problem is reflected by the presence at a time t of several products. The algorithm for solving this problem that contains additional steps is not presented in this article.

To complete the presentation of case studies, the result of the translation into a Petri net of the design control law is proposed in the case of multi-product problem.

5.4.5. Logic control law

From the viewpoint of the control system, the loading system is a system in which several products can be present simultaneously and for which there is one different objective for two product families. Indeed, after the rotation of the rotating storage, the DPC operation (Drop a product in C by the operator) is always executed for the design algorithm by lack of

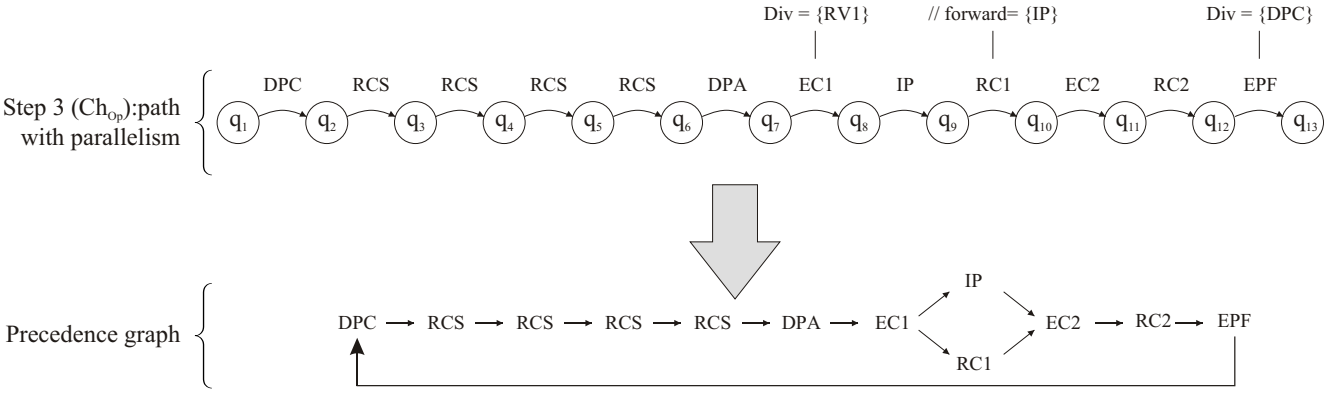

Figure 13: Result of parallelism search and translation into a precedence graph.

information provided by the operator. The presence of a product dropped in C is therefore uncertain. The real presence of a product is detected in A with the detection operation. Because of this uncertainty about the presence of products on the rotation storage, the loading system is characterized by the treatment of two product families: the present products and lacking products. In addition, the system can act on several products simultaneously which may include on the rotating storage in A, B, C and D. Although the steps of the design algorithm are not presented in this article, the figure 14 shows the result of the design algorithm for these two additional problems and translation of the result into a Petri net. A software environment to design control laws has been developed in Java under Eclipse.

5.4.6. Result with automated planning approach

With a classic automated planning approach (Valente & E.Carpanzano, 2011), a single kind of operation is defined in contrast to the proposed approach, which defines three kinds. With no difference between pre-conditions and pre-constraints in automated planning, coupling between functional-chain and product evolutions is lacking or permanent. The sequence of operations obtained with an automated planning approach is limited to non-optimal sequence of processing operations or a sequence of moving operations in case of lacking of process operation. For the proposed system, the controlled system will be able to go through step 21 in Figure 12 before to achieve the state goal 19. Finally, the result does not contain all the operations, the sequence of operations does not minimize the cycle time and parallelism do not exist between the operations. Finally, the system would treat the products one by one while it is possible to have more than six products simultaneously in the manufacturing cell.

6. Conclusion

In this paper, a methodology to help automation engineers to design off line a logic control law from the controlled system abilities is proposed. The modeling concept is based on the extension of the one proposed in Automatic Planning field of research: operation model. There are double interests of this controlled system modeling. First, there is no problem of combinatory explosion which is generally the limit of automaton representation usually used at this level. Next, the model structuring and the required information (effects of a service, conditions, constraints) facilitate the modeling task. Such a modeling approach presents also the advantage to capitalize a technical knowledge of the controlled system abilities resolving in part the departure of automation engineers from the company. Moreover, considering separately each of the operation of the controlled systems allows to simply update the model in case of changes due to physical reconfigurations or loss of controlled system capacities in failures context.

Based on the proposed decomposition, the resulting logic control law is not optimal. In return, the proposed algorithm is based on a compromise between the complexity of the build states space and the solution's performance. Finally, it is important to notice that the resulting solution support on a formalism of representation favorable with its translation into languages of standard IEC 61131-3, thus opening interesting application for PLC programming.

Future works will focus first on the way to develop a graphical interface to help designer to capitalize the required knowledge using a functional chain point of view. Secondly, the problem with several different types of products will be studied. The principle would be very similar to the steps defined to face parallelism for several identical products. By the end, this logic control law design approach will be extended to an uncertain execution context characterized by resource failures in order to provide some reactivity abilities to the manufacturing system. In this context, an automatic update process of the model of the controlled system must be studied. Indeed a failure has two consequences, first the evolution of the AMS can be in an unexpected state and second, a possible modification to the AMS capabilities. Then before to try to design a new adequate control law, the model must take these modifications into account.

References

Aylett, R. S. (2001). Planning plant operating procedures for chemical plant. *Engineering Applications of Artificial Intelligence*, *14*, 341–356.

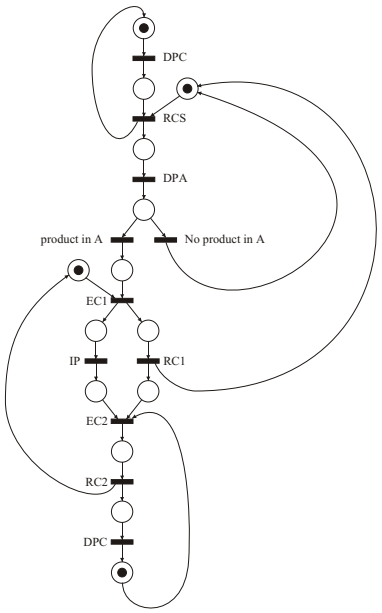

Figure 14: Petri net translation of designed control law for multiple products.

Castillo, L., Fdez-Oliveras, J., & Gonzales, A. (2000). Intelligent planning of grafcet charts. *Robotics and Computer Integrated Manufacturing*, *16*, 225–239.

Charbonnier, F., Alla, H., & david, R. (1999). The supervised control of discret-event dynamic systems. *IEEE Transactions on Control Systems Technology*, *7*, 175–187.

Chretienne, P., Coffman, E. G., Lensta, J. K., & Liu, Z. (1997). *Scheduling theory and its applications*. Chichester: John Wiley and Soons.

CIM (1989). A reference model for computer integrated manufacturing from the viewpoint of industrial automation. *International Journal of Computer Integrated Manufacturing*, *2*, 114–127.

Combacau, M., Berruet, P., Zamai, E., Charbonnaud, P., & Khatab, A. (2000). Supervision and monitoring of production systems. In *IFAC 2nd Conference on Management and Control of Production and Logistics (MCPL'00)*. Grenoble, France.

Davenport, A. J., & Beck, J. C. (2000). A survey of techniques for scheduling with uncertainty.

Dijkstra, E. W. (1959). A note in two problems in connexion with graphs. *Numerisches Mathematik*, (pp. 269–271).

E.W. Endsley, E. A., & Tilbury, D. (2006). Modular finite state machines: Development and application to reconfigurable manufacturing cell controller generation. *Control Engineering Practice*, *14*, 11271142.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning, Theory and Practice*. Elsevier.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, *8*, 231–274.

Herroelen, W., & Leus, R. (2002). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, .

Holloway, L. E., Guan, X., Sundaravadivelu, R., & Jr., J. A. (2000). Automated synthesis and composition of taskblocks for control of manufacturing systems. *IEEE Trans. On Systems, Man and Cybernetics, Part. B*, *30*.

IEC-61131-3 (1993). *Programming Languages - Providing the Basis*. Swiss.

ISA-S95 (2000). *Entreprise Control System Integration*.

Klein, I. (1999). Efficient planning for a miniature assembly line. *Artificial Intelligence in Engineering*, *13*, 69–81.

Lacomme, P., Prins, C., & Sevaux, M. (2003). *Algorithmes de Graphes*. Eyrolles.

Lee, G. B., Zandong, H., & Lee, J. S. (2004). Automatic generation of ladder diagram with control petri net. *Journal of Intelligent Manufacturing*, *15*, 245–252.

Lee, J. K., & Lee, T. E. (2002). Automata-based supervisory control logic design for a multi-robot assembly cell. *International Journal of Computer Integrated Manufacturing*, *15*, 319–334.

Li, J., Dai, X., Meng, Z., Dou, J., & Guan, X. (2009). Rapid design and reconfiguration of petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams. *Computers and Industrial Engineering*, *57*, 1431–1451.

Linz, P. (2000). *An introduction to formal languages and automata*. Sudbury, MA: Jones and Bartlett Publishers.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*, 541580.

Qiu, R., Wysk, R., & Xu, Q. (2003). Extend structured adaptative supervisory control model of shop floor controls for an e-manufacturing system. *International Journal of Production Research*, *41*, 1605–1620.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discret event processes. *Journal of Control and Optimization*, *25*.

Sandewall, E., & Ronnquist, R. (1986). A representation of action structures. In *National Conference on Artificial Intelligence (AAAI'86)* (pp. 89–97).

Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, *22*, 971–978.

Valente, A., & E.Carpanzano (2011). Development of multi-level adaptive control and scheduling solutions for shop-floor automation in reconfigurable manufacturing systems. *CIRP Annals-ManufacturingTechnology*, *in press*.

Zaytoon, J. (2002). On the recent advances in grafcet. *Production Planning and Control*, *13*, 86–100.