Manufacturing Tasks Synchronization by Algebraic Synthesis

Tom Ranger, Philippot Alexandre, Bernard Riera

Manufacturing Tasks Synchronization by Algebraic Synthesis

T. RANGER * **A. PHILIPPOT** ** **B. RIERA** ***

* *Université de Reims Champagne Ardenne, CReSTIC EA 3804, 51097 Reims, France (e-mail: tom.ranger@univ-reims.fr)*
** *Université de Reims Champagne Ardenne, CReSTIC EA 3804, 51097 Reims, France (e-mail: alexandre.philippot@univ-reims.fr)*
*** *Université de Reims Champagne Ardenne, CReSTIC EA 3804, 51097 Reims, France (e-mail: bernard.riera@univ-reims.fr)*

Abstract: The development of control programs for PLCs is mainly carried out by direct implementation. We propose to switch to formal methods to obtain these control laws. For this we use a task-based structural analysis approach assisted by algebraic synthesis. We obtain a control law in ST language allowing to ensure the functional behaviour defined in the specifications. This makes the generation of PLC programs more reliable and efficient.

Keywords: Tasks synchronisation, algebraic approaches, logic controllers, formal methods, discrete event systems, modular control

1. INTRODUCTION

Programmable logic controllers (PLCs) are frequently used as industrial automation components in a very large number of systems such as production systems for the control of complex systems. Control engineers today mainly handle the development of industrial automation applications by direct implementation of the control task based on the interpretation of informal specification and text documents. This effort is assisted by standardized engineering tools for the programming of PLCs (Zaytoon and Riera, 2017). However, the informal specification of the control software has to be manually and intuitively transferred into the control program as they are not formally defined in practise due to lack of time and expertise. The growing complexity of the control problem, the demand for reduced development time, and the possible reuse of existing software modules result in the need for formal approaches in logic control design and PLC programming. To reach this goal, several formal approaches have been proposed for the design of logic controllers. This area is mainly filled with works using Supervisory Control Theory (SCT) defined by (Ramadge and Wonham, 1989), 30 years ago. A large body of theoretical results has appeared since then, but there is still a gap between the theoretical development and the limited number of applications of SCT in the industry using PLC (Fabian and Hellgren, 1998) (Cantarelli and Roussel, 2008) (Zaytoon and Riera, 2017). In addition, the large scale of industrial systems requires to use modular approaches in order to reduce the problem complexity (de Queiroz and Cury, 2000). Applying a modular approach to SCT is not without difficulties. In the works of Hellgren (Hellgren et al., 2002) it is explained that it is necessary to have an internal synchronization of the controllers. With regard to these difficulties, it could be interesting to use formal non SCT based approaches like Algebraic Synthesis (Roussel and Lesage, 2014) able to support parts of the work of control engineers. In this paper a task-based structural analysis approach is proposed to obtain the control laws. The Algebraic Synthesis (AS) is used to manage the synchronization tasks problem.

The first part of the paper deals with algebraic synthesis. The second section of this paper consists on a presentation of a method using task-based structural analysis. The last part of the paper details how to synchronise the tasks thanks to AS.

2. ALGEBRAIC SYNTHESIS

All the equations in this paper are based on Boolean algebra $(\mathcal{B}, +, ., ^-, 0, 1)$ (Definition 15.5 of Grimaldi (2004)).

2.1 Overall presentation

Algebraic synthesis is used in the framework of discrete event systems (DES) as shown in the figure 1. The controller of such a system consists of p Boolean inputs (u_i), of q Boolean outputs (y_j) and of r Boolean states variables (x_l). Obtaining the control law for an algebraic model requires the generation of $(q + r)$ switching functions of $(p + q)$ variables. This generation must be automated because of the combinatorial explosion. The system controller presented in figure 1 can send 2^q output combinations that can express $(2^{2^{p+r}})^{q+r}$ sequential behaviors. This automation is made possible by Brown's results on Boolean algebra (Brown, 1990).

The principle is to translate each of the requirements of the specification into the form of a Boolean equation. The resolution of this system provides an expression of the outputs and state variables as a function of the inputs and state variables. However, initially the solution is in a parametric form. Indeed, the latter expresses the

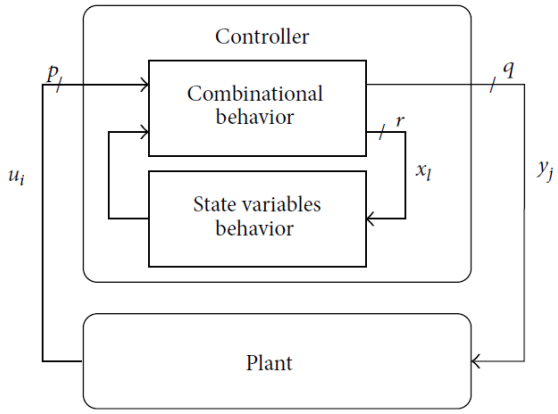

Fig. 1. A sequential DES

whole space of the admissible solutions. Since PLC control requires a deterministic behavior, the parametric solution must then be reduced to a single solution.

Equation mainly uses an inclusion form. It is a partial relation which allow to define conditions of sufficiency or necessity between 2 boolean expressions. Let us take a_1 and a_2 2 known variables and B_1 and B_2 2 unknown variables. To express the fact that it is enough to have a_1 to get B_1 the equation 1 is used. The equation 2 expresses the fact that it is necessary to have a_1 to get B_1. The equation 3 expresses the fact that in order to obtain B_1 when a_2 is true it is necessary to have a_1. Finally, the equation 4 expresses the fact that the variables B_1 and B_2 cannot be true simultaneously.

$$a_1 \leq B_1 \tag{1}$$

$$B_1 \leq a_1 \tag{2}$$

$$B_1.a_2 \leq a_1 \tag{3}$$

$$B_1.B_2 = 0 \tag{4}$$

Concerning the choice of a single solution, it is possible to use optimization criteria on Boolean expressions. The use of such criteria in Algebraic Synthesis has been presented in the work of Leroux and Roussel (Leroux and Roussel, 2012). An optimization criterion is a linear combination of variables in the problem and can be maximized or minimized. With each application of a criterion the space of solutions of the unknowns contained in this criterion is reduced. These criteria are applied in the order in which they are defined. With each application the solution space is reduced, so the order of definition has an impact on the solution. It should be noted that part of the expected behavior can be expressed via these optimization criteria.

2.2 Previous uses of Algebraic Synthesis

Originally, LURPA developed the algebraic synthesis (Roussel and Lesage, 2012) on the basis of Brown's work in order to generate control laws in a formal way. Their

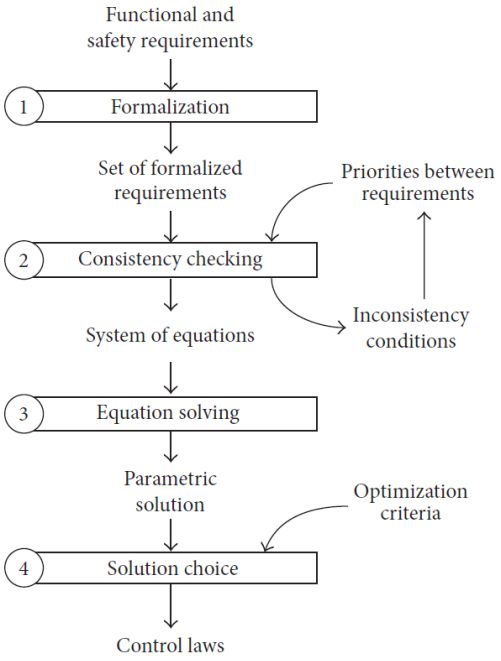

Fig. 2. Steps to obtain PLC code by algebraic synthesis

methodology is based on 4 steps presented in figure 2. The first step consists in formalizing the problem. Indeed, before using the algorithms for solving systems of Boolean equations, the functional and safety requirements must be available in the form of Boolean equations. At first it is necessary to define a set of specifications informally before transcribing them into Boolean equations. In the second step the coherence of the system of equations obtained is checked. This ensure that there is no contradiction between several equations. Inconsistencies may reflect two types of problems, errors in the definition of the specifications or conflicts between security constraints and functional requirements. Once all the inconsistency problems have been removed, the resolution can take place. This third step provides the parametric solution to our problem. The implementation into a PLC requires the choice of a solution among the proposed set. This choice can be assisted by the use of optimization criteria.

By applying this approach, the goal is to generate the entire control law, from its functional behavior to the management of safety aspects. However, it is difficult to describe sequential behaviors using Boolean equations. This difficulty comes from the fact that these equations define constraints and that it is complicated to define sequences of actions by constraints. This is why it is difficult to design the entire control law in algebraic synthesis. So we decided to use algebraic synthesis only for some parts of the control law generation. In the next section shows where the algebraic synthesis can be used in the handling of a synchronization problem.

3. TASK-BASED STRUCTURAL ANALYSIS

A methodology constituted by a task-based structural analysis presents the management of the synchronisation problem using Algebraic Synthesis. The idea of this method is to obtain the PLC code by partitioning the

functional command into elementary tasks which then are synchronized in order to meet the requirements of the specifications.

3.1 Overall method

This method consists of the following 4 steps:

(1) Identification of basic system tasks ;
(2) Specification of task behavior ;
(3) Definition of task launch conditions ;
(4) Generation of a controller to ensure task synchronization.

Step 1: Identification of basic system tasks The first step in a task-based structural analysis is to subdivide the system into elementary tasks that it can perform. Each of these tasks must remain independent, and it is the synchronization of these tasks that makes it possible to respect the functional specifications.

It is necessary that the tasks are independent of each other during their execution. That is why each task must respect the following rules:

- Must have a launch condition ;
- Must not require additional conditions during its execution ;
- Must not emit authorizations during its execution ;
- Must generate one or more authorizations at the end of execution.

A functional analysis must be performed to identify the different tasks. The more tasks are added, the more flexible the command can be and the more complex behaviors can be defined. However, the complexity of synchronization may increase. It is therefore necessary to find the right level of granularity to meet the functional specifications without unnecessarily multiplying the number of tasks. This step is the weak point of the method because a poorly performed task identification can greatly complicate the rest of the method.

Step 2: Specification of task behavior After having identified the different tasks of our system their behavior must be specified. Once this step is done the system situation at the start of the task, the added value of the task to the overall process and the system situation at the end of the execution must be known for each task. The behavior can then be modeled by different tools but we are not focusing on this part in this paper.

Step 3: Definition of task launch conditions Now that the behavior of the tasks is specified, it is necessary to determine their launch conditions. All the conditions for launching the tasks are gathered in a table such as the one presented in table 2 and 3. Before authorizing the launch of a task 3 types of conditions need to be checked. The first kind of condition are the initial conditions which are indicated in the second column of table 2. They are defined based on the system inputs and are used to ensure that a task have access to the right resources before starting. Then the anteriority conditions are be defined. Their role is to make sure that the tasks are properly sequenced. The definition of the anteriority conditions is done in three steps.

(1) Determination of the induced tasks list (third column of table 2);
(2) Definition of the tokens generated at the end of the execution of each task (fourth column of table 2) ;
(3) Generation of the anteriority conditions by gathering the tokens into logical expressions (second column of table 3).

Concerning the step 2, different kinds of token generation can be distinguished. In case a task always causes the same tasks to be carried out, the same set of tokens is generated at each of its end of execution. However in other cases the induced tasks depends on the state of the system at the end of the task. In this case the generation of the tokens is done under condition.

The other conditions are those of access to shared resources. Indeed, in many industrial systems this type of resources can be found. Whether it is a conveyor, a robotic arm or access to a material flow, it is necessary to manage the conditions of access to these resources during the design of the control system. The first step is to identify the shared system resources and determine which tasks should have access to them. All tasks that have access to the same R_i resource must then be ordered by priority of access to this resource. A task with an order of 1 has priority over one with an order of 2. This information is provided in the third column of table 3.

Step 4: Generation of a controller to ensure task synchronization Finally, based on the table obtained from step 2, a controller must be generated to synchronize the tasks. Even with a correctly constructed synchronisation table it can be extremely complex to obtain such a controller. However, with the help of algebraic synthesis we propose a methodology to automatically generate such a controller.

3.2 Mathematical framework and notations

We are using the following notations:

- T_i: Task i ;
- $Start\_Ti$: Starting event of the task i ;
- End_T_i: Ending event of the task i ;
- GT_{i_j}: Generated token by task i for task j ;
- GT_{i_jk}: Generated token by task i for task j or k ;
- $AntC\_T_i$: Condition of anteriority of the task i ;
- AUT_T_i: Authorisation for the launch of the task i ;
- REQ_T_i: Request for the launch of the task i ;
- X^{-1}: Variable X at the previous cycle of the PLC ;
- R_i: Shared resource i ;
- R_{i_j}: Shared resource i assigned in order of priority j.

3.3 Application of the method to an academic benchmark

In this section, the objective is to illustrate the methodology presented above by applying it to an example. Here only the first three steps of the method are dealt with, the fourth will be dealt with in the section 4. Concerning

step 2 the task modeling is not presented because it has no impact on synchronization management. Moreover, it should be noted that the goal here is not to have the most optimized control in terms of production cycle time but to illustrate the use of the method on an example, the number of tasks is therefore deliberately limited.

The studied system (presented figure 3) aims at filling a box with 3 pieces that can be supplied by two different conveyors. The observer C_1 indicates if the box is ready to be evacuated. As long as there are less than 3 pieces in the box, C_1 is false. Each of the parts conveyors has a robotic arm equipped with a suction cup to place the parts in the box. Table 1 presents the input/output variables of the benchmark:

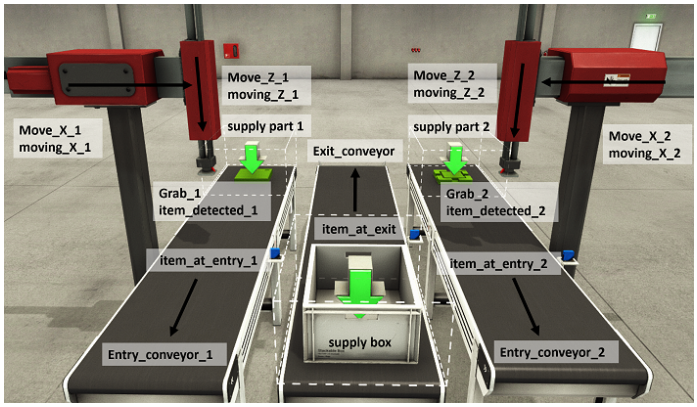

Fig. 3. Studied system

Table 1. System Inputs/Outputs

Inputs	Outputs
item_at_entry_1	*Entry_conveyor_1*
item_at_entry_2	*Entry_conveyor_2*
item_at_exit	*Exit_conveyor*
moving_X_1	*Move_X_1*
moving_X_2	*Move_X_2*
moving_Z_1	*Move_Z_1*
moving_Z_2	*Move_Z_2*
item_detected_1	*Grab_1*
item_detected_2	*Grab_2*

For this system we identify the following tasks:

- T1: Supply part 1 ;
- T2: Supply part 2 ;
- T3: Take part 1 ;
- T4: Take part 2 ;
- T5: Supply box ;
- T6: Deposit part 1 ;
- T7: Deposit part 2 ;
- T8: Box evacuation.

Tasks 1 and 2 correspond to bringing a part to the sensors *item_at_entry_1* and *item_at_entry_2*. Tasks 3 and 4 start with the empty arm above the entry conveyors and end with a part gripped by the suction cup above the entry conveyors. Task 5 is the feeding of a box to the *item_at_exit* sensor. Tasks 6 and 7 start with a part picked up by the suction cup above the entry conveyor and end in the same position after depositing the part in the box.

Table 2. Task synchronization table (1)

Task	Initial conditions	Induced tasks	Tokens generated
T_1	$\overline{item\_at\_entry\_1}$	T_3	GT_{1_3}
T_2	$\overline{item\_at\_entry\_2}$	T_4	GT_{2_4}
T_3	$item\_at\_entry\_1$	T_1 and T_6	GT_{3_1} and GT_{3_6}
T_4	$item\_at\_entry\_2$	T_2 and T_7	GT_{4_2} and GT_{4_7}
T_5	$\overline{item\_at\_exit}$	T_6 or T_7	GT_{5_67}
T_6	$item\_detected\_1$	T_3 and (T_8 if C_1 or T_6 if $\overline{C_1}$)	GT_{6_3} and (GT_{6_8} if C_1 or GT_{6_67} if $\overline{C_1}$)
T_7	$item\_detected\_2$	T_4 and (T_8 if C_1 or T_7 if $\overline{C_1}$)	GT_{7_4} and (GT_{7_8} if C_1 or GT_{7_67} if $\overline{C_1}$)
T_8	$C1$	T_5	GT_{8_5}

Table 3. Task synchronization table (2)

Task	Anteriority conditions	Order of access to resources
T_1	$AntC\_T_1 = GT_{3\_1}$	
T_2	$AntC\_T_2 = GT_{4\_2}$	
T_3	$AntC\_T_3 = GT_{1\_3}.GT_{6\_3}$	
T_4	$AntC\_T_4 = GT_{2\_4}.GT_{7\_4}$	
T_5	$AntC\_T_5 = GT_{8\_5}$	
T_6	$AntC\_T_6 = GT_{3\_6}.(GT_{5\_67} + GT_{6\_67} + GT_{7\_67})$	R_{1_1}
T_7	$AntC\_T_7 = GT_{4\_7}.(GT_{5\_67} + GT_{7\_67} + GT_{6\_67})$	R_{1_2}
T_8	$AntC\_T_8 = GT_{6\_8} + GT_{7\_8}$	

Finally task 8 ends once a falling edge has been detected on the *item_at_exit* sensor.

Tasks 6 and 7 must share access to the central conveyor. The latter is therefore a shared resource whose access priority must be managed. This priority is given to task 6 in this example.

Tables 2 and 3 presents the task synchronization table for the tasks.

4. USE OF ALGEBRAIC SYNTHESIS TO OBTAIN THE SYNCHRONIZATION CONTROLLER

The previous section did show how to define a problem using a task-based structural analysis approach. It ended with the writing of the synchronisation table. This sections aims to show how to switch from this table to a system of Boolean equations in order to be able to use algebraic synthesis to solve the synchronization problem.

4.1 Tasks synchronizations

At first we are looking at tasks that do not require access to a shared resource. Our problem presents 2 types of unknowns, the generated tokens and the authorizations to launch tasks. The execution of a task is only authorized when the initial and anteriority conditions are checked for this task. For a task to be authorized it is necessary that the initial conditions as well as the precedence conditions are true. The task authorizations are therefore constrained by an equation of the type 2. The authorization of the task T_i is thus managed by the equation 5.

$$AUT\_T_i \leq AntC\_T_i \ . \ IC\_T_i \tag{5}$$

The equation 6 represents the example of the task T_3.

$$AUT\_T_3 \leq AntC\_T_3 \ . \ IC\_T_3 \tag{6}$$

It is also necessary to manage the evolution of the generated tokens. In order to manage their evolution we need to define the sufficient conditions for a token to be true or false. For this, equations of the form 1 will be used. Let's consider the GT_{i_j} token, this token is generated when the ending event of the task i is observed. On the other hand, he is consumed when the task j is executed. This leads to the writing of the equations 7 and 8.

$$End\_T_i.\overline{GT_{i\_j}^{-1}} \leq GT_{i\_j} \tag{7}$$

$$Start\_T_j.GT_{i\_j}^{-1} \leq \overline{GT_{i\_j}} \tag{8}$$

The equation 9 presents the generation and consumption of the token $GT1\_3$.

$$\begin{aligned} End\_T_1.\overline{GT_{1\_3}^{-1}} &\leq GT_{1\_3} \\ Start\_T_3.GT_{1\_3}^{-1} &\leq \overline{GT_{1\_3}} \end{aligned} \tag{9}$$

In the case of conditionally generated tokens like GT_{6_8} only the generation of the token is modified. This generation is only done if the condition C_1 is verified as presented in the equation 10

$$End\_T_6.\overline{GT_{6\_8}^{-1}}.C_1 \leq GT_{6\_8} \tag{10}$$

Applying these rules to all authorizations and tokens results in a system of Boolean equations. As the resolution of this system gives parametric solutions, optimization criteria must also be defined in order to obtain a unique solution for each output.

With regard to authorizations, it is desirable that the execution of tasks be authorized as soon as possible. The expression of each authorization is therefore maximized as in the equation 11. Tokens should only be produced and consumed when necessary. Their evolution is limited by applying the criterion presented in the equation 12.

$$Crit\_AUT\_T_i = Max(AUT\_T_i) \tag{11}$$

$$Crit\_GT_{i\_j} = Min(GT_{i\_j}^{-1}.\overline{GT_{i\_j}} + \overline{GT_{i\_j}^{-1}}.GT_{i\_j}) \tag{12}$$

4.2 Management of shared resources

The generation of the task authorizations is changed to manage the access to shared resources. The idea is to add a step before issuing the authorization. Where it was sufficient to have the conditions of precedence and the initial conditions verified to authorize the launch of a task, it is now necessary to add an access authorization to the resource. However, it is not enough to have the resource free to be able to use it. Indeed, it must also be assigned to the task when making the access request. In this way it is ensured that two accesses are not granted to the same resource during an PLC cycle.

To manage these access authorizations are transformed into access requests. Then if the resource is assigned to the task by making the request an authorization is issued. The equation 13 presents the situation where the task i needs to have access to the resource R_1. Moreover, in order to ensure that the authorizations follows the value of the requests as much as possible, a similarity criteria between these two variables is defined. The equation 14 presents this criteria which has to be maximized.

$$\begin{aligned} REQ\_T_i &\leq AntC\_T_i.CI\_T_i \\ AUT\_T_i &\leq REQ\_T_i.R_1 \end{aligned} \tag{13}$$

$$SIM\_T_i = AUT\_T_i.REQ\_T_i + \overline{AUT\_T_i}.\overline{REQ\_T_i} \tag{14}$$

Now we must prevent the resource from being allocated more than once. To do this we prohibit all tasks requiring access to the resource to have their launch authorizations true simultaneously. In our case, we prevent AUT_T_6 and AUT_T_7 to be true at the same PLC cycle using the equation 15.

$$AUT\_T_6 \ . \ AUT\_T_7 \ = \ 0 \tag{15}$$

Finally, we must be able to manage the priority of access to resources. To do this we exploit one of the properties of the optimization criteria presented in the 2.1 section. The order in which these criteria are defined depends on the priority given to the tasks. The task with the highest priority will have its criterion applied first. Solving this system of equations allows to obtain the logical expressions of the authorizations as well as the generated tokens. These expressions can be directly written in ST code which is one of the standard languages for PLCs (figure 4).

For the resulting code to be functional, it is also necessary to indicate which tokens will be available at the initial time. For this example the tokens $GT3\_1$, $GT4\_2$, $GT6\_3$, $GT7\_4$ and $GT8\_5$ are available at launch.

```
GT1_3  :=  End_T1 AND NOT Start_T3 OR End_T1
 AND NOT GT1_3p OR NOT Start_T3 AND GT1_3p;

AUT_T1  :=  IC_T1 AND End_T3 AND NOT Start_T1
 OR IC_T1 AND End_T3 AND NOT GT3_1p OR IC_T1
 AND NOT Start_T1 AND GT3_1p;
```

Fig. 4. Task 1 equations in ST code

4.3 Implementation

The implementation of our method is assisted by several automation tools. In the figure 5 we distinguish the steps carried out by the designer (upper frame) from those automated (lower frame). Two of these tools have been developed by us and the last one comes from LURPA. The LURPA tool is a Boolean equation system solver. It takes as input a file containing the variables (known and unknown), the constraint equations and the optimization criteria and provides the solution of the system in ST code. The first tool we have developed allows to generate the input file of this solver from the task synchronization table. The second tool allows to obtain the control law from the data of the Factory I/O scene, the tasks defined in the form of GRAFCET and the solutions of the Boolean equation system. Because of the method of separation by elementary tasks, GRAFCETs are linear and independent which makes them easy to define. The command is generated in a UNITY file.

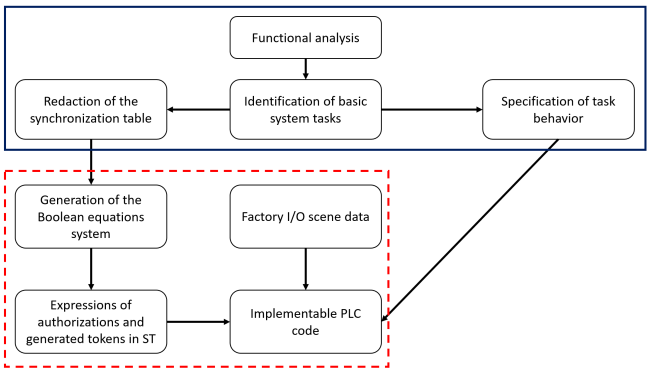

Fig. 5. Steps to obtain PLC code by task synchronisation

5. CONCLUSION

In this paper we have presented a method to obtain PLC program through task-based structural analysis using a formal tool that is algebraic synthesis to ensure task synchronization. The evolution of the tasks being described by Grafcet it allows to overcome the problem met by Roussel concerning the description of sequential behavior. Moreover this approach allows to automate the whole task synchronization process once their chaining rules have been defined. In the future, we intend to explore in more details the choice of task granularity. In addition, we wish to integrate this approach into a global method that also deals with safety requirements. Finally, we would like to focus on the problems of verification and validation. This will concern the PLC program but also the synchronization table whose validity is critical for this method.

ACKNOWLEDGEMENTS

This paper is carried out in the context of the HUMANISM ANR-17-CE10-0009 research program, funded by the ANR "Agence Nationale de la Recherche". This project involves three laboratories in Automatics Sciences and in Human Factors, CReSTIC (Reims), Lab-STICC (Lorient), and LAMIH (Valenciennes).

The authors gratefully acknowledge these institutions.

REFERENCES

Brown, F.M. (1990). *Boolean Reasoning.* Kluwer Academic Publishers.

Cantarelli, M. and Roussel, J.M. (2008). Reactive control system design using the Supervisory Control Theory: evaluation of possibilities and limits. In *9th International Workshop On Discrete Event Systems (WODES'08)*, pp. 200–205. Göteborg, Sweden.

de Queiroz, M.H. and Cury, J.E.R. (2000). Modular supervisory control of large scale discrete event systems. In *Discrete Event Systems: Analysis and Control (R. Boel, G. Stremersch(Eds.)). Kluwer Academic Publishers.*

Fabian, M. and Hellgren, A. (1998). Plc-based implementation of supervisory control for discrete event systems. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171).*

Grimaldi, R.P. (2004). *Discrete and Combinatorial Mathematics: An Applied Introduction.* Addison-Wesley Longman Publishing Co.

Hellgren, A., Lennartson, B., and Fabian, M. (2002). Modelling and plc-based implementation of modular supervisory control. In *Sixth International Workshop on Discrete Event Systems, 2002. Proceedings.*, 371–376. doi:10.1109/WODES.2002.1167713.

Leroux, H. and Roussel, J.M. (2012). Algebraic synthesis of logical controllers with optimization criteria. In *6th International Workshop on Verification and Evaluation of Computer and Communication Systems VECOS 2012*, pp. 103–114. Paris, France. 12 pages.

Ramadge, P. and Wonham, W. (1989). Control of discrete event systems. *Proceedings of the IEEE, vol.77, no.1, pp.643-652.*

Roussel, J.M. and Lesage, J.J. (2012). Algebraic synthesis of logical controllers despite inconsistencies in specifications. In *11th International Workshop on Discrete Event Systems, WODES 2012*, pp. 307–314. Guadalajara, Mexico. 8 pages.

Roussel, J.M. and Lesage, J.J. (2014). Design of Logic Controllers Thanks to Symbolic Computation of Simultaneously Asserted Boolean Equations. *Mathematical Problems in Engineering*, 2014, Article ID 726246. 15 pages.

Zaytoon, J. and Riera, B. (2017). Synthesis and implementation of logic controllers – a review. *Annual Reviews in Control*, 43, 152 – 168.