Algebraic Synthesis of Safety Logical Filter on Manufacturing Systems

T Ranger, Philippot Alexandre, Bernard Riera

Algebraic Synthesis of Safety Logical Filter on Manufacturing Systems

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: This work focuses on the management of safety constraints for the control of cyber-physical manufacturing systems. A methodology for constructing a set of constraints to ensure the safety of an existing control law is proposed. The resolution of this constraint set is performed using algebraic synthesis. This tool facilitates the implementation of a logic filter in a way that complies with IEC 61131-3.

Keywords: Discrete Event Systems, Programmable Logic Controllers, Safety, Formal Methods, Algebraic Approaches

1. INTRODUCTION

In this article, we will focus on the control of industrial manufacturing systems. These systems being controlled by Programmable Logic Controllers (PLCs) they can be modeled by discrete event systems (DES) (Cassandras et al., 2008) with logical Inputs (sensors), logical Outputs (actuators) and internal variables (observers). The development of Industry 4.0 requires new design paradigms for the control of cyber-physical manufacturing systems. One of them being the ability to ensure operational safety. In order to guarantee this reliability we decided to use formal approaches. The most common one discussed in the literature is Supervisory Control Theory (SCT) (Ramadge and Wonham, 1989). However, the use of PLCs imposes a synchronous operation which is opposed to the asynchronous character of finite state automata used in SCT. Moreover, the SCT addresses the problem of supervisor synthesis where it is necessary to obtain controllers for implementation in a PLC. The difference between the 2 is major, indeed where a supervisor will limit the evolution of the system by remaining as permissive as possible a controller will force to follow a single path. This is why we decided to use an approach based on the use of safety constraints placed after the controller in order to act as a logical safety filter (Pichard et al., 2018b) to compensate for possible safety flaws in the PLC program.

This kind of filter can be applied to any control program, for which it is necessary to be exhaustive when writing the safety constraints because no assumption can be made on the control law. They can also be defined in conjunction with the control law, this approach simplifies the definition of these two elements (Zaytoon and Riera, 2017). The previous work used a SAT solver minimizing the Hamming distance to solve the security constraints, which involved implementing an online solving algorithm. Another method of solving the constraints is therefore preferred. The use of algebraic synthesis (Hietter et al., 2008) was chosen. This approach allows the solution of systems of Boolean equations. The solution obtained is presented in the form of a logical relation, which allows to get rid of the online solution algorithm.

The first part of this article presents the notion of logic filter and the use of algebraic synthesis as a solution tool. In the second part a methodology for the implementation of a logic filter using algebraic synthesis is presented. Finally the use of this method is illustrated in the last part.

2. STATE OF THE ART

All the equations in this paper are based on Boolean algebra $(\mathcal{B}, +, ., ^-, 0, 1)$ (Definition 15.5 of Grimaldi, 2004) where "+", ".", "-" are respectively the logical operators OR, AND and NOT.

2.1 Logical filter

The principle of operation of a logic filter is to insert a validation block at the end of the control program. The purpose of this block is to detect possible violations of safety constraints among the whole output vector. The detection of one of these violations can lead to the blocking of the system in a stable state or to the modification of the output vector in order to ensure the respect of the safety constraints.

Our approach of logical filters is based on the one developed at CReSTIC (Marangé et al., 2010), (Pichard et al., 2018b). A logical filter takes place after the program execution as shown in figure 1. Two kinds of modifications can occur to the command law. In the case of a blocking filter, a constraint violation will lead the system to a predetermined and stable state. In the case of a corrective filter,

a constraint violation will lead to an output adjustment to fit with the filter constraint set.

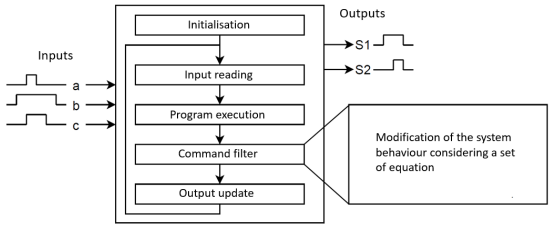

Fig. 1. Principle of implementation of the logic filter in a PLC

The contribution of Pichard (2018a) to logical filter brings a formalism to the definition of logical constraints which enable him formal verification for logical filter consistency. Work on solving systems of Boolean equations (Roussel and Lesage, 2014) is used to check the consistency of the set of constraints. In Pichard's work he distinguishes two types of constraints, simple constraints and combined constraints. These constraints are made of unknown (actuator) and known variables (input PLC variables). Simple constraints have only one unknown variable, while combined constraints have several.

As a simple example, let us consider i_1 a known variable and O_1 and O_2 2 unknown variables. For instance equation 1 is a simple constraint and equation 2 is a combined constraint.

$$CSs_1 \ = \ i_1.O_1 \tag{1}$$

$$CSc_1 \ = \ O_1.O_2 \tag{2}$$

A constraint is considered satisfied if it is equal to 0. The objective being to define a control logic filter, the violation of one or more constraints must lead to a modification of the output vector. While solving simple constraints is straightforward, solving combined constraints requires additional information. Indeed, equation 2 can be solved in 3 different ways:

(1) $O_1 = 1$ and $O_2 = 0$
(2) $O_1 = 0$ and $O_2 = 1$
(3) $O_1 = 0$ and $O_2 = 0$

The choice of the variable to be set to 1 in priority will then be indicated between brackets when defining the constraint. If both are to be set to 0, no further indication is given. The constraint Cc_1 is then defined as follows if the priority is given to O_1 :

$$Cc_1 \ = \ O_1.O_2 \ \ [O_1] \tag{3}$$

This problem has been identified as a boolean satisfiability problem which is abbreviated as SAT (Vizel et al., 2015). A SAT solver has therefore been implemented to solve this problem (Pichard et al., 2018a). This solver will provide an output vector conforming to the safety constraints at each PLC cycle. This vector is chosen using the Hamming distance. This distance measures the deviation from the output vector proposed by the control law. The vector provided at the end of the resolution algorithm is therefore the one minimizing this Hamming distance.

However the implementation of this solver is also the weak point of the method. Indeed, the implementation of an online solving algorithm has a major drawback. The convergence time of the algorithm being variable, it can lead to uncontrolled cycle times. In particular if the cycle time of the automaton reaches the characteristic time of evolution of the system. This is why we decided to turn to another tool to synthesize a control logic filter.

2.2 Algebraic synthesis

The use of algebraic synthesis as a solving tool has been retained for the implementation of control logic filters. Algebraic synthesis is a method for solving systems of Boolean equations.

Initially algebraic synthesis was developed at LURPA by Hietter (2008). The original idea was to use the results of Brown (1990) to generate the control law by solving a Boolean equations system. Hietter's work has focused on formalizing mathematical tools for solving systems of Boolean equations. These tools allow to obtain a parametric solution to a problem formed by a set of Boolean equations. The existence of a solution is guaranteed by the verification of a consistency condition.

The latest theoretical results on the domain come from Hélène Leroux (2012). This paper brings optimisation criteria. These criteria allow to orient the choice of a solution. There are 2 types of criterion, the maximisation and the minimisation. These criteria are defined by a logical expression that will be maximised or minimised. The set formed by the constraint system and the optimization criteria is the problem to solve. This resolution takes place in 4 steps:

(1) Computation of parametric solutions by solving the constraint system without applying criteria
(2) Computation of an optimization criterion by replacing the unknown variables by the parametric solutions computed in step 1
(3) Addition of the expression of the criterion in the set of equations and resolution of this new system to obtain a new parametric solution
(4) Repeat steps 2 and 3 to apply a new criterion

The criteria being applied one after the other, their order of definition has an impact on the solution.

The use of these criteria to guide the choice of a solution is presented bellow.

For instance, consider O_1 and O_2 2 unknown variables and i_1 a known variable. The following constraint is defined:

$$O_1.O_2 \ = \ 0 \tag{4}$$

At the end of the first step, as defined in Theorem 11 (Roussel and Lesage, 2014) the following parametric expressions is obtained:

$$\begin{aligned} O_1 &= p_1 \\ O_2 &= \overline{p_1}.p_2 \end{aligned} \tag{5}$$

With p_1 and p_2 arbitrary parameters.

Then the objective is to get rid of the parameters by applying the following preferences:

- when i_1 is true, we want O_1 to be true
- O_1 must be true as less as possible
- O_2 must be true as much as possible

Therefore the following criteria are defined:

$$Max(i_1.O_1) \tag{6}$$

$$Min(O_1) \tag{7}$$

$$Max(O_2) \tag{8}$$

The resolution of the first criterion (equation 6) is presented bellow:

$$Max(i_1.O_1) = i1 \tag{9}$$

The new equation to solve is:

$$\begin{aligned} & i_1.O_1 = Max(i_1.O_1) \\ \Leftrightarrow\ & i_1.p_1 = i_1 \\ \Leftrightarrow\ & i_1.p_1.\overline{i_1} + \overline{i_1.p_1}.i_1 = 0 \\ \Leftrightarrow\ & i_1.\overline{p_1} = 0 \end{aligned} \tag{10}$$

This new equation is solved considering p_1 the unknown variable. The following parametric solution is obtained by injecting the value of p_1 obtained in the first solution:

$$\begin{aligned} O_1 &= i_1 + p'_1 \\ O_2 &= p_2.\overline{p'_1}.\overline{i_1} \end{aligned} \tag{11}$$

With p'_1 an arbitrary parameter. The application of the second criterion (equation 7) provides the following parametric solution:

$$\begin{aligned} O_1 &= i_1 \\ O_2 &= p_2.\overline{i_1} \end{aligned} \tag{12}$$

Finally the application of the last criterion (equation 8) gives:

$$\begin{aligned} O_1 &= i_1 \\ O_2 &= \overline{i_1} \end{aligned} \tag{13}$$

Therefore, a unique solution is obtained after applying 3 optimization criteria. However we can observe that the last two criteria removed 1 parameter while the first one only provided a new parametric solution. As we want to reach a unique solution in order to implement it in a PLC we need to know under which conditions a criterion effectively removes a parameter. We have therefore defined a sufficiency condition to make a parameter disappear from a parametric solution.

Theorem 1. (Parameter reduction condition theorem). For an optimization criterion to remove the parameter associated with an unknown variable x, it is sufficient that it is of the form:

$$Min(ax + b\overline{x}) \tag{14}$$

or

$$Max(ax + b\overline{x}) \tag{15}$$

where $a, b \in \mathcal{B}$ *such as* $a = \overline{b}$

Proof. To prove the theorem we will show that the sufficiency condition implies the removal of the parameter for both forms of criteria.

We will start with the maximization criterion. Let us consider the following equation with one unknown $f(x) = 0$. According to Theorem 10 of (Roussel and Lesage, 2014) this equation can be put in the following canonical form:

$$f(0).\overline{x} + f(1).x = 0 \tag{16}$$

as presented in (Leroux and Roussel, 2012) the result of the solution of this equation to which we apply the criterion:

$$Max(ax + b\overline{x}) \tag{17}$$

is:

$$\begin{aligned} x = f(0) + \overline{a}.b.\overline{f(1)} \\ +p.(\overline{a}.\overline{f(1)} + b.\overline{f(1)}) \end{aligned} \tag{18}$$

in this case if $a = \overline{b}$:

$$\begin{aligned} & x = f(0) + \overline{a}.\overline{a}.\overline{f(1)} \\ & + p.(\overline{a}.\overline{f(1)} + \overline{a}.\overline{f(1)}) \\ & = f(0) + \overline{a}.\overline{f(1)} + p.\overline{a}.\overline{f(1)} \\ & = f(0) + \overline{a}.\overline{f(1)} \end{aligned} \tag{19}$$

For the minimization criterion the solution is :

$$\begin{aligned} x = f(0) + a.\overline{b}.\overline{f(1)} \\ +p.(a.\overline{f(1)} + \overline{b}.\overline{f(1)}) \end{aligned} \tag{20}$$

And in the same way we obtain:

$$x = f(0) + a.\overline{f(1)} \tag{21}$$

In both cases the result of the application of the criterion gives a solution without parameters.

In order to obtain a unique solution for a system of equations with n unknowns, it is sufficient to apply 1 criterion respecting the reduction condition to each of these variables. It is important to note that the criteria being applied one after the other the criteria not respecting this condition must be defined first in order to have an impact on the solution.

3. SETTING UP A SAFETY FILTER

Our goal is to apply a safety filter using algebraic synthesis for any control law. In existing works on logic filters the set of constraints was always considered as well constructed. In this section we will propose a method to obtain this set by performing a system analysis.

3.1 System analysis

The first step for the implementation of a safety filter goes through a system analysis. This analysis is used to identify potential sources of risks for the system. The first step is to identify all the risk areas in the system. Then for each of this area every situation that can lead to a safety issue must be defined. Finally a set of constraints is drafted to prevent the occurrence of these situations.

In order to identify these situations the following questions must be asked for each area identified:

- Is there any risk of collision if some actuators are operated simultaneously?
- Is there any risk of cluttering the system if some actuators are not activated together?
- Can the actuators be controlled in a way that damages them?

3.2 Problem formalization

Now that the safety constraints are written we will show how to define problem that will be solved by algebraic synthesis.

The definition of the constraints is different from the one used by Pichard (2018a). Indeed, even if they were defined as logical equations, additional indications were provided for the combined constraints. Moreover, the definition of an algebraic synthesis problem requires more than the constraints redaction. It is also necessary to ensure that a unique solution is obtained. For simple constraints, they are defined in the same way as in Pichard's work. The difference appears for the combined constraints, in particular for the choice of the solution. Additional variables are also used to define the problem. These are known variables representing the request of the execution of an actuator by the control law. If the actuator variables are of the form O_i these execution requests are of the form R_O_i.

The choice of the solution to be prioritized in the case of combined constraints is made using the optimization criteria. Let us go back to the combined constraint example presented earlier:

$$Cc_1 \ = \ O_1.O_2 \ \ [O_1] \tag{22}$$

The formalization of this constraint in the framework of the algebraic synthesis is carried out in two steps.

Writing a Boolean constraint equation:

$$O_1.O_2 \ = \ 0 \tag{23}$$

The definition of an optimization criterion:

$$Min(R\_O_1.R\_O_2.\overline{O_1}) \tag{24}$$

This criterion is interpreted as follows, in the case where O_1 and O_2 are requested simultaneously $\overline{O_1}$ must be minimize. This has the effect of favouring the solution $O_1 \ = \ 1$ and $O_2 \ = \ 0$.

In the same way that R. Pichard used a solver minimizing the Hamming distance, we want the value of the actuator variables to be as close as possible to the demands of the control law. We therefore define similarity variables between the actuator execution requests and the final values of these actuators. Optimization criteria are then defined to maximize these similarity variables.

For our example the similarity variables will be expressed as follows:

$$\begin{aligned} Sim\_O_1 \ &= \ R\_O_1.O_1 + \overline{R\_O_1}.\overline{O_1} \\ Sim\_O_2 \ &= \ R\_O_2.O_2 + \overline{R\_O_2}.\overline{O_2} \end{aligned} \tag{25}$$

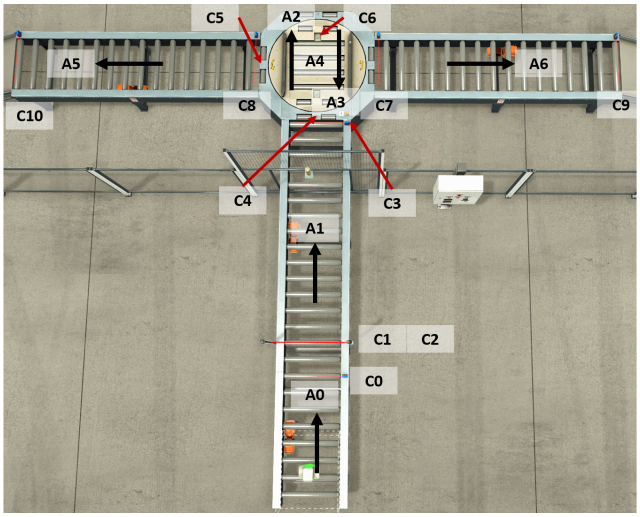

Fig. 2. Box sorting system

Table 1. System Inputs

Inputs	Description
C_0	Box between the two supply conveyors
C_1	Box at the case detection
C_2	Box detected
C_3	Box at the end of the second supply conveyor
C_4	Turntable aligned with supply conveyors
C_5	Turntable aligned with exit conveyors
C_6	Box at the end of the turntable
C_7	Box at the beginning of the right exit conveyor
C_8	Box at the beginning of the left exit conveyor
C_9	Box at the end of the right exit conveyor
C_{10}	Box at the end of the left exit conveyor

And the criteria will be defined as follows:

$$\begin{aligned} &Max(Sim\_O_1) \\ &Max(Sim\_O_2) \end{aligned} \tag{26}$$

For these criteria if we take $a = R\_O_1$ and $b = \overline{R\_O_1}$ the condition $a = \overline{b}$ is well verified. These criteria ensure the uniqueness of the solution.

4. APPLICATION OF THE METHOD TO AN ACADEMIC BENCHMARK

In order to illustrate the application of control logic filters by algebraic synthesis, we use the example presented in Pichard et al. (2016).

4.1 Notations

The following notations is used:

- X^{-1}: Variable X at the previous cycle of the PLC ;
- $\uparrow X \ = \ \overline{X^{-1}}.X$: Rising edge of variable X ;
- $\downarrow X \ = \ X^{-1}.\overline{X}$: Falling edge of variable X.

4.2 Benchmark presentation

This system is represented in figure 2 and his inputs and outputs are detailed in tables 1 and 2.

In addition to these inputs we use 2 observers named $P01$ and $P36$ which indicates the presence of a box respectively between the sensors C_0/C_1 and C_3/C_6

Table 2. System Outputs

Outputs	Description
A_0	First supply conveyor
A_1	Second supply conveyor
A_2	Turntable conveyor direction 1
A_3	Turntable conveyor direction 2
A_4	Rotation of the turntable
A_5	Left exit conveyor
A_6	Right exit conveyor

4.3 Filter definition and implementation

In the box sorting system we can identify 5 risk areas :

(1) Connection between the two supply conveyors;
(2) Connection between the second supply conveyor and the turntable;
(3) Connection between the turntable and the right exit conveyor;
(4) Connection between the turntable and the left exit conveyor;
(5) The turntable.

The situations to be avoided are now defined for each of these risk areas as well as the safety constraints to avoid them. The list of these situations and their associated constraints are presented in the table 3.

We can notice that a single constraint can be used to avoid several situations. Now this set of constraints needs to be formalized into an algebraic synthesis problem. The constraints 27 and the optimization criteria 28 and 29 have to be defined.

$$\begin{cases} P_{01}.A_0 = 0 \\ C_0.A_0.\overline{A_1} = 0 \\ C_3.\overline{C_4}.A1 = 0 \\ C_3.(C_6 + P_{36}).A1 = 0 \\ C_3.C_4.A_1.\overline{A_2} = 0 \\ \uparrow A_4.(A_2 + A_3) = 0 \\ C_4.\overline{\uparrow C_6}.\downarrow A_2 = 0 \\ \downarrow A_4.(A_2 + A_3) = 0 \\ C5.\overline{\downarrow C_8}.\downarrow A_2 = 0 \\ C5.\overline{\downarrow C_7}.\downarrow A_3 = 0 \\ \overline{C_5}.C_6.A_3 = 0 \\ \overline{C_5}.C_6.A_2 = 0 \\ A_2.A_3 = 0 \end{cases} \quad (27)$$

Table 3. Situations to avoid and associated safety constraints

Situation to avoid	Safety constraints
Pallets should not touch each other (1)	$P_{01}.A_0$
We cannot load a part on the A_1 if it is stopped (1)	$C_0.A_0.\overline{A_1}$ $[\overline{A_1}]$
A part cannot be loaded on the turntable if it is not aligned with the supply conveyors (2)	$C_3.\overline{C_4}.A1$
It is not possible to load a part on the turntable if one is already on it (2)	$C_3.(C_6 + P_{36}).A1$
It is not possible to load a part on the turntable if the conveyor A2 is stopped (2)	$C_3.C_4.A_1.\overline{A_2}$ $[\overline{A_2}]$
The turntable cannot be turned towards the exit if its conveyors are used (2)	$\uparrow A_4.(A_2 + A_3)$ $[A_2 + A_3]$
A loading cannot be stopped until it is completed (2)	$C_4.\overline{\uparrow C_6}.\downarrow A_2$
An unloading by A_3 cannot be stopped until it is completed (3)	$C5.\overline{\downarrow C_7}.\downarrow A_3$
The turntable cannot be turned towards the supply if its conveyors are used (3)	$\downarrow A_4.(A_2 + A_3)$ $[A_2 + A_3]$
The turntable cannot be turned towards the supply if its conveyors are used (4)	
An unloading by A_2 cannot be stopped until it is completed (4)	$C5.\overline{\downarrow C_8}.\downarrow A_2$
The turntable cannot be unloaded by A_3 if it is not aligned with the exit conveyors (5)	$\overline{C_5}.C_6.A_3$
The turntable cannot be unloaded by A_2 if it is not aligned with the exit conveyors (5)	$\overline{C_5}.C_6.A_2$
The turntable conveyors cannot be operated in both directions simultaneously (5)	$A_2.A_3$

$$\begin{cases} Min(R\_A_0.\overline{R\_A_1}.C_0.A_1) \\ Min(C_3.C_4.R\_A_1.\overline{R\_A_2}.A_2) \\ Min(\overline{A_4^{-1}}.R\_A_4.(R\_A_2 + R\_A_3).(\overline{A_2} + \overline{A_3})) \\ Min(A_4^{-1}.\overline{R\_A_4}.(R\_A_2 + R\_A_3).(\overline{A_2} + \overline{A_3})) \\ Min(R\_A_2.R\_A_3.(A_2 + A_3)) \end{cases} \quad (28)$$

$$\begin{cases} Max(Sim\_A0) \\ Max(Sim\_A1) \\ Max(Sim\_A2) \\ Max(Sim\_A3) \\ Max(Sim\_A4) \\ Max(Sim\_A5) \\ Max(Sim\_A6) \end{cases} \tag{29}$$

The solution of this problem provides the solutions presented in 30. As the filter only consists of logical relations, it can be directly implemented in ladder or ST code in a PLC in order to be IEC 61131-3 compliant.

$$\begin{cases} A_0 &= R\_A_0.\overline{P_{01}}.(\overline{C_0} + A_1) \\ A_1 &= R\_A_1.(\overline{C_3} + C4.\overline{C_6}.\overline{P_{36}}.(A_2^{-1} + R\_A_2.\overline{R\_A_3})) \\ A_2 &= A_2^{-1}.(C_4.\overline{C_6} + C_5.\overline{\downarrow C_8}) + R\_A_2.\overline{R\_A_3} \\ & \quad .(\overline{C_6}.(C_4 + \overline{C_5}) + C_5.(A_2^{-1} + \overline{A_3^{-1}} + \downarrow C_7)) \\ A_3 &= A_3^{-1}.C_5.\overline{\downarrow C_7} + R\_A_3.\overline{R\_A_2}.(C_5.(\overline{A_2^{-1}} + \downarrow C_8) \\ & \quad + \overline{C_6}.(\overline{A_2^{-1}} + \overline{C_4}.\overline{C_5}) + C_4.\downarrow C_6) \\ A_4 &= A_4^{-1}.(A_2 + A_3) + R\_A_4.(A_4^{-1} + C_4.C_6 + \overline{A_2}.\overline{A_3}) \\ A_5 &= R\_A_5 \\ A_6 &= R\_A_6 \end{cases} \tag{30}$$

Compared to the filter proposed by Pichard (2018a) we obtain a similar functional behavior using the control law proposed in his paper. Moreover the Hamming distance has been computed online and in both cases the maximum observed distance is 3. The filter obtained using algebraic synthesis thus has a behavior similar to that using an SAT solver without the need to implement an online solver.

5. CONCLUSION

In this paper we have presented a method for defining a corrective logic filter to ensure compliance with security constraints. This filter is implemented using algebraic synthesis as a tool to solve the set of security constraints. This allows for an easily implementable solution that does not require the use of an online resolution algorithm. Since this filter only takes into account the security aspects, it does not guarantee that the expected functional behavior is respected. Indeed, in the case of a poorly designed control law, the application of a safety filter can lead to a system block. In this case the filter can be used to identify the defects of the control law to correct them. We also plan to develop a filter to ensure the liveliness of the system in our future work.

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.

Cassandras, C.G., Lafortune, S., et al. (2008). *Introduction to discrete event systems*, volume 2. Springer.

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

Hietter, Y., Roussel, J.M., and Lesage, J.J. (2008). Algebraic synthesis of dependable logic controllers. *IFAC Proceedings Volumes*, 41(2), 4132–4137. 17th IFAC World Congress.

IEC 61131-3 (2013). Programmable controlers-part 3: Programming languages. *(3rd ed.)*.

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.

Marangé, P., Benlorhfar, R., Gellot, F., and Riera, B. (2010). Prevention of human control errors by robust filter for manufacturing system. *IFAC Proceedings Volumes*, 43(13), 175–180.

Pichard, R., Ben Rabah, N., Carre-Menetrier, V., and Riera, B. (2016). Csp solver for safe plc controller: Application to manufacturing systems. *IFAC-PapersOnLine*, 49(12), 402–407. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.

Pichard, R., Philippot, A., and Riera, B. (2018a). Safe PLC Controller Implementation IEC 61131-3 Compliant based on a Simple SAT Solver: Application to Manufacturing Systems. In *15th International Conference on Informatics in Control, Automation and Robotics*. SCITEPRESS - Science and Technology Publications, Porto, France.

Pichard, R., Philippot, A., Saddem, R., and Riera, B. (2018b). Safety of manufacturing systems controllers by logical constraints with safety filter. *IEEE Transactions on Control Systems Technology*, 27(4), 1659–1667.

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

Vizel, Y., Weissenbacher, G., and Malik, S. (2015). Boolean satisfiability solvers and their applications in model checking. *Proceedings of the IEEE*, 103(11), 2021–2035.

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