CSP solver for Safe PLC Controller: Application to manufacturing systems

R. Pichard, N. Ben Rabah, V. Carre-Menetrier, B. Riera

CSP solver for Safe PLC Controller: Application to manufacturing systems

Conference Paper · June 2016

DOI: 10.1016/j.ifacol.2016.07.638

4 authors:

Romain Pichard
Université de Reims Champagne-Ardenne
13 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Nourhène ben Rabah
Ecole Nationale des Sciences de l'Informatique
6 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Véronique Carré-Ménétrier
Université de Reims Champagne-Ardenne
56 PUBLICATIONS **214** CITATIONS
SEE PROFILE

Bernard Riera
Université de Reims Champagne-Ardenne
109 PUBLICATIONS **376** CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-12 (2016) 402–407

CSP solver for Safe PLC Controller: Application to manufacturing systems

R. PICHARD*, N. BEN RABAH*, V. CARRE-MENETRIER* and B. RIERA*

* *CReSTIC (EA3804), UFR Sciences Exactes et Naturelles, Reims University (URCA), Moulin de la Housse, BP 1039, 51687 Reims - France (bernard.riera@univ-reims.fr).*

Abstract: This paper presents an original approach of safe control synthesis for manufacturing systems controlled by Programmable Logic Controller (PLC) based on the use of a CSP (constraint satisfaction problem) solver. In this work, manufacturing systems are considered as Discrete Event Systems (DES) with logical Inputs (sensors) and logical Outputs (actuators). The proposed approach separates the functional control part from the safety control part. The methodology is based on the use of safety constraints in order to get from a CSP solver all the safe outputs vectors at each PLC scan time. The safe outputs vector is selected by choosing the one which minimizes the Hamming distance with the functional outputs vector. The approach is illustrated with a sorting boxes simulated process using the ITS PLC software from the Real Games Company (www.realgames.pt).

Keywords: Discrete-Event Systems, CSP, Control, Safety, Programmable Logic Controllers, Manufacturing Systems.

1. INTRODUCTION

This paper presents an original approach of control synthesis for manufacturing systems controlled by PLC (Programmable Logic Controller). In this work, manufacturing systems are considered as Discrete Event Systems (DES) (Cassandras *et al.,* 1999) with logical Inputs (sensors) and logical Outputs (actuators). This is an extension of the research work that the CReSTIC (Research Center in Information and Communication Science and Technologies) has led for several years on the definition and design of guard conditions (also called constraints) placed at the end of the PLC program which act as a logic filter in order to be robust to control errors. These safety constraints can be formally checked off line by using a model checker (Marangé *et al.,* 2010). This idea has been extended to propose a safe control design pattern based on safety logical constraints. This approach, which separates the functional control part from the safety control part, is easy to implement and involve a new way to design the controller. The methodology is based on the use of safety constraints in order to get the most permissive safe controller allowed by the safety constraints set. This controller is then constrained by functional constraints while respecting the safety constraints (Riera *et al.,* 2014, 2015). In this paper, we propose a new approach for the control synthesis algorithm. The idea is to use at each PLC scan time a CSP (Constraint Satisfaction Problem) solver to get the set of all outputs vectors respecting the set of safety constraints and to select the one which is the closest in the sense of Hamming distance of the functional outputs vector. Hence, the controller continues to work with safe outputs values. This approach to PLC programming makes safety a priority and allows for a controller to create a safe environment where functional and safety aspects are clearly separated. Compared to the algorithm already proposed, the approach using a CSP solver does not require to define priorities between outputs when a combined safety constraint is violated. The first part of the paper is dedicated to the concept of robust logic filter to control errors. In the second part, the definition and mathematical formalism used for the safety guards are detailed. The third part presents the control algorithm using a CSP solver. At least, the approach is illustrated by using one example: a virtual sorting system using the ITS PLC software from the Real Games Company (www.realgames.pt). Today, PLC does not include CSP solver. To test the idea, a soft PLC written in IronPython and using logilab-constraint, an open source constraint solver (https://www.logilab.org/project/logilab-constraint) written in pure Python controller, has been designed. This example shows the interest in terms of simplicity and efficiency of this original control synthesis method. It seems to be the first time that a CSP solver is used in real time as a part of a PLC program to get a safe controller.

2. LOGICAL GUARDS FOR SAFE PLC PROGRAM

Since a PLC is a dedicated controller it will only process this one program over and over again. One cycle through the program is called a scan time and involves reading the inputs (i) from the other modules (input scan), executing the logic based on these inputs (logic scan) and then updated the outputs (o) accordingly (output scan). The memory in the CPU stores the program while also holding the status of the I/O and providing a means to store values. A controller at each PLC scan time has to compute the outputs values (controllable variables) based on inputs (uncontrollable variables) and internal memories. The use of a memory map

10.1016/j.ifacol.2016.07.638

enables to guarantee that all the calculations are performed with inputs values which are not modified during a PLC scan time. Outputs update is performed with the last outputs calculation in the PLC program. These three basic stages of operations (input scan, logic solve and output scan are repeated at each scan time (cf. figure 1).

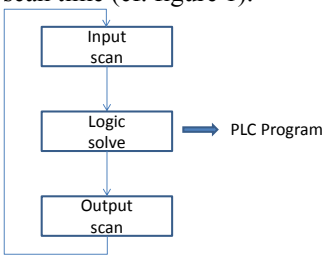


Fig. 1. PLC operation sequence

The idea proposed by (Marangé *et al.,* 2010) is to place a logic filter between the logic solve and the output scan. The goal of this filter is to detect and compensate control errors (cf. figure 2).

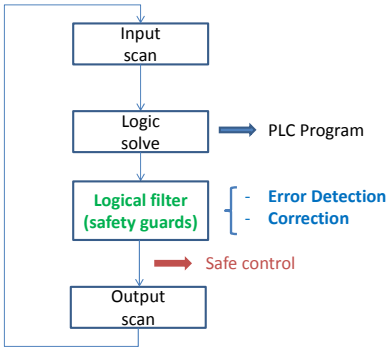


Fig. 2. Principle of the logic filter

Three use cases can be thought of doing with the logic filter: safe blocking, supervisor and controller. In the first case, when a safety constraint is violated, the controller is frozen in a safe state which is supposed known. The supervisory approach consists in correcting the control errors without blocking the controller. This enables for instance to safe existing PLC program without changing the code. The controller approach is similar to the supervisor approach. The main difference is that in the design of the controller, it is taken into account by the designer that the safety part is managed by the safety constraints. Hence, there is a separation between functional and safety aspects of the controller. In addition, even if the functional part is badly defined, the system remains safe (Riera *et al.*, 2015). Contrary to the supervisor approach, the fact to violate a safety constraint can be seen as normal (cf. figure 3).

This approach modifies the way to design a PLC program but presents several advantages (tasks synchronization, management of running modes, connection to a Manufacturing Execution System …). Control design based on logical constraints involve 2 main difficulties:

1) Constraints definition and validation which are not going. We suppose in this paper that the designer has got a correct set of safety constraints.

2) The proposal of a control algorithm which defines, when one or several constraints are violated, a safe outputs vector compliant with all the safety constraints.

We have already proposed an algorithm to compute at each PLC scan time a safe outputs vector (Riera *et al.*, 2015). The idea in this paper is to propose a new approach based on CSP to perform the detection and the correction stages. The main advantages of this new approach come from the fact that it is simpler than the previous algorithm because it is not necessary to define priority between outputs when a combined safety constraint is violated. The idea is to use at each PLC scan time a CSP solver to get the set of all outputs vectors respecting the set of safety constraints and to select the one which is the closest in the sense of Hamming distance of the functional outputs vector. Indeed, the functional part of the PLC program aims at reaching the production goals. The idea is to select the safe outputs vector which is the most similar to the functional outputs vector.

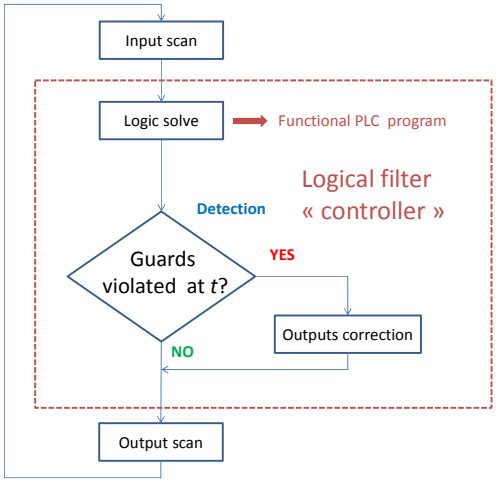


Fig. 3. Logic filter as a controller

2. BOOLEAN SAFETY CONSTRAINTS FORMALISM

The notations used in the following paper are:

- *t*: current scan time (from PLC point of view), *t-1* previous PLC scan time.
- $o_k = o_k(t)$: logical variable corresponding to the value of k^{th} PLC boolean output (actuator) at *t*. Outputs at *t* are considered as the one and only variables that can be controlled (write variables) at each PLC scan time. All other PLC variables (inputs, previous outputs …) are uncontrollable (read only variables).
- $o_k^* = o_k(t-1)$: logical variable corresponding to the value of k^{th} PLC boolean output (actuator) at time *t-1* (previous PLC scan time).
- "$.$", "$+$", "$\oplus$" "$\overline{\ }$" are respectively the logical operators AND, OR, XOR and NOT.
- 0 means *False* and 1 means *True*.
- $\uparrow x$ and $\downarrow x$ means respectively rising edge and falling edge of boolean variable x (in the PLC, $\uparrow x = \overline{x^*}.x, \downarrow x = x^*.\bar{x}$).
- $\sum$ and $\prod$ are respectively the logical sum (OR) and the logical product (AND) of logical variables.
- $\sum\prod$ is a logical polynomial (sum of products expression also called SIGMA-PI).
- O: set of output variables at *t*.
- Y: set of uncontrollable variables at *t, t-1, t-2* ...

- N_o: PLC boolean outputs number.
- N_{CSs}: Simple Safety Constraints number.
- N_{CSc}: Combined Safety Constraints number.

The proposed methodology to design safe controllers is based on the use of logical safety constraints, which act as logical guards placed at the end of the PLC program, and forbid sending unsafe control to the plant (Marangé *et al.,* 2010). The set of safety constraints acts as a control filter.

Constraints (or guards) are always modeled with the point of view of the control part (PLC), and it is assumed that the PLC scan time is sufficient to detect any change of the input vector (synchronous operation, possible simultaneous changes of state of PLC inputs). In addition, the plant is considered functioning normally without failure.

It is considered in this work that the initial safe state for all the actuators (o_k) is defined to be 0. The constraints have to be defined in order to keep the system controllable. This means that, even with the set of safety constraints, it is possible to design a controller which matches the specifications. For example, considering the previous hypothesis about the safe initial state, a set of safety constraints which resets at each scan time all outputs is safe but does not ensure the controllability. Some guards involve a single output at time *t* (called simple safety constraints *CSs*), other constraints involve several outputs at time *t* (combined safety constraints *CSc*). Constraints require the knowledge of I/O at the current time *t* and possibly previous times (presence of edge (*t-1*) for instance). Safety constraints are not always depending only on PLC inputs at *t*. It may be necessary to define supplementary uncontrollable variables called observers. Observers are memories enabling to get a combinatory constraint.

The set of safety constraints is considered as necessary and sufficient to guarantee the safety. In this approach, it is assumed that the safety constraints can always be represented as a monomial and depend on the inputs (at time *t, t-1, t-2*…), outputs (at time *t, t-1, t-2*…) and observers (depending ideally on only inputs (at time *t, t-1, t-2*…). In the initial methodology (Marangé *et al.* 2010), the control filter is validated offline by model checking (Behrmann *et al.,* 2002) and stops the process in a safe state if a safety constraint (*CSs* and *CSc*) is violated.

In this paper, *CSs* and *CSc* are represented (equations (1) and (2)) as logical monomial functions ($\prod$, products of variables but not necessarily minterms) which have always to be *False* at the end of each scan time, before updating the outputs, in order to guarantee the safety. It is important to note that each *CSs* depends only on one controllable event (output: o_k) and that each *CSc* depends on several controllable events (outputs: o_k, o_l, …).

$$\forall m \in [1, \mathrm{N_{CSs}}],\ \exists!\, k \in [1, \mathrm{N_o}] /\ \ CSs_m = \prod(o_k, \mathrm{Y}) \quad (1)$$

$$\forall n \in [1, \mathrm{N_{CSc}}],\ \exists!\, (k, l, \dots) \in [1, \mathrm{N_o}]\ avec\ k \neq l \neq \cdots /\ \ CSc_n = \prod(o_k, o_l, \dots, \mathrm{Y}) \quad (2)$$

To guarantee the safety, *CSs* and *CSc* must be *False* (=0) in the PLC program before updating outputs, the logical sum of safety constraints computed with all o_k has to be *False* (equation 3). It is the detection function of the logic filter. A PLC program can be considered as safe if for the outputs vector $(o_1, \dots, o_k, \dots, o_{No})$, equation (3) is verified before output scan.

$$\sum_{i=1}^{N_{CSs}} CSs_i + \sum_{\mathrm{j}=1}^{N_{CSc}} CSc_\mathrm{j} = 0 \quad (3)$$

There are only 2 exclusive forms of simple safety constraints (*CSs*) because they are expressed as a monomial function, and they only involve a single output at time *t* (equation (4):

$$\forall m \in [1, \mathrm{N_{CSs}}], \quad \exists!\, k \in [1, \mathrm{N_O}] /\ \ CSs_m = o_k \cdot h_{0m}(\mathrm{Y}) + \overline{o_k} \cdot h_{1m}(\mathrm{Y}) \ \text{with}\ h_{0m}(\mathrm{Y}) \oplus h_{1m}(\mathrm{Y}) = 1 \quad (4)$$

These simple safety constraints (*CSs*) express the fact that if $h_{0m}(\mathrm{Y})$, which is a monomial (product) function of only uncontrollable variables at *t*, is *True*, o_k must be necessarily *False* in order to keep the constraints equal to 0. If $h_{1m}(\mathrm{Y})$ is *True*, o_k must be necessarily *True*. In addition, it is not possible to have $h_{0m}(\mathrm{Y})$ and $h_{1m}(\mathrm{Y})$ true simultaneously. For each output, it is possible to write equation (5) corresponding to a logical OR of all simple safety constraints.

$$\sum_{i=1}^{N_{CSs}} CSs_i = \sum_{k=1}^{N_o} \left(f_{sk}(o_k, Y) \right) \quad (5)$$

$f_{sk}(o_k, \mathrm{Y})$ is a logical $\sum\prod$ function independent of the other outputs at *t* because only *CSs* are considered. $f_{sk}(o_k, \mathrm{Y})$ can be developed in equation (6) where f_{s0k} and f_{s1k} are polynomial functions (sum of products, $\sum\prod$) of uncontrollable (read only) variables. Equation (6) has always to be *False* because all simple safety constraints must be *False* at the end of each PLC scan time. To simplify equations, a logical function can be represented by a logical variable having the same name.

$$f_{sk}(o_k, Y) = o_k \cdot f_{s0k}(Y) + \overline{o_k} \cdot f_{s1k}(Y)$$
$$f_{sk} = f_{sk}(o_k, Y) = o_k \cdot f_{s0k} + \overline{o_k} \cdot f_{s1k} \quad (6)$$

From equations (5) and (6), it is possible to write equation (7).

$$\sum_{\mathrm{i}=1}^{\mathrm{N_{CSs}}} \mathrm{CSs_i} = \sum_{\mathrm{k}=1}^{\mathrm{N_o}} \left(\mathrm{o_k} \cdot \mathrm{f_{s0k}}(\mathrm{Y}) + \overline{\mathrm{o_k}} \cdot \mathrm{f_{s1k}}(\mathrm{Y}) \right)$$
$$\sum_{\mathrm{i}=1}^{\mathrm{N_{CSs}}} \mathrm{CSs_i} = \sum_{\mathrm{k}=1}^{\mathrm{N_o}} (\mathrm{o_k} \cdot \mathrm{f_{s0k}} + \overline{\mathrm{o_k}} \cdot \mathrm{f_{s1k}}) = \sum_{k=1}^{N_o} f_{sk} \quad (7)$$

The outputs vector can be considered as safe at the end of the PLC scan time, if equation (8) is checked.

$$\sum_{k=1}^{N_o} f_{sk} + \sum_{\mathrm{j}=1}^{N_{CSc}} CSc_\mathrm{j} = 0 \quad (8)$$

One can notice that we have got a set of safety constraints and a formalism which is compliant with a constraints satisfaction problem (CSP) to find a safe outputs vector.

3. SAFE PLC CONTROLER BASED ON CSP SOLVER

CSP are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations (Hooker, 2000) (Krzysztof, 2003) (Tsang, 1993). CSP represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods. CSP are the subject of intense research in both artificial intelligence and operations research, since the regularity in their formulation provides a common basis to analyze and solve problems of many seemingly unrelated families. CSP often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods to be solved in a reasonable time. In this paper, a part of the control problem is seen as a CSP. Formally, in this work, a constraint satisfaction problem is defined as a triple:

$O = \{o_1, \dots, o_n\}$ is the set of outputs variables,

$D = \{True, False\}$ is a set of the respective domains of values,

$C = \{CSs_1, \dots, CSs_{N_{CSs}}, CSc_1, \dots, CSc_{N_{CSc}}\} = \{C_1, \dots, C_{N_{CSs}+N_{CSc}}\}$ is the set of simple safety constraints and combined safety constraints.

It could be possible to define only 1 constraint as the sum of all constraints but when using CSP solver it is important to try to avoid constraints with a lot of variables. It is better to have several binary constraints than one big N-ary constraint with several "and" conditions. In order to minimize the size of the set of constraints, we are going to use the following reduced set of constraints based on equation (8).

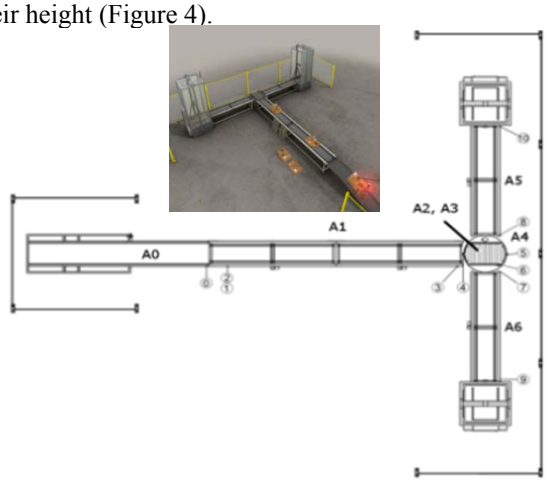
$$Cr = \{f_{s1}, \dots, f_{sN_O}, CSc_1, \dots, CSc_{N_{CSc}}\} = \{Cr_1, \dots, Cr_{N_O+N_{CSc}}\} \quad (9)$$

Constraints with less variables are placed first in the list, because they will get evaluated before by the CSP solver. Hence, it will take less time to process, and will hopefully reduce the domains of the variables playing a role in the *CSc* constraints. Each variable o_i can take on the values in the nonempty domain $\{True, False\}$. Every constraint Cr_k is in turn a pair $\langle t_j, R_j \rangle$ where $t_j \subset X$ is a subset of k variables and R_j is an k-ary relation on the corresponding subset of domains $\{True, False\}$. An evaluation of the variables o is a function from a subset of variables to a particular set of values in the corresponding subset of domains. An evaluation v satisfies $\langle t_j, R_j \rangle$ if the values assigned to the variables t_j satisfies the relation R_j. An evaluation is consistent if it does not violate any of the constraints. An evaluation is complete if it includes all variables. An evaluation is a solution if it is consistent and complete; such an evaluation is said to solve the constraint satisfaction problem.

The idea is to use a CSP solver at each PLC scan time to get all the safe output vectors based on the safety constraints. From these, we select the first one which is the closest from the functional output vectors. For that, the Hamming distance is used. In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In another way, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. This approach seems quite simple and interesting. If the Hamming distance is null, this means that the functional outputs vector is safe and can be updated. If the Hamming distance is different from 0, this means that the functional outputs vector is not safe. Selecting the safe outputs vector which has got the smallest Hamming distance from the functional one enables to select the closest outputs vector in order to achieve the production (i.e. functional) goals. Of course, that will work if the functional part of the controller is correctly designed. However, whatever the functional part (even if it is badly designed), the system will remain safe.

4. EXAMPLE OF A SORTING SYSTEM

The control algorithm will be illustrated by the mean of a virtual system from the ITS PLC collection, proposed by the Portuguese company Real Games. ITS PLC collection is a set of simulation software dedicated to automation training. Demos and technical descriptions of the five virtual industrial systems are available and freely downloadable at web address www.realgames.pt. As part of the work presented in this paper, the "sorting system" is used. Today, PLC does not include CSP solver. So in order to test this original control approach, we have used a specific release of ITS PLC where one can script ITS PLC in IronPython, a language that offers the Python programming style and all the power of the .NET framework (http://ironpython.net). One can use scripting to interface with the inputs and outputs of each system and to design your own soft PLC. Hence it becomes easy to design advanced controllers in IronPyhon which works exactly as a PLC. In addition, there are several CSP libraries in Python enabling to realize a proof of concept of our approach. The objective of the "sorting system" is to transport boxes from entry conveyor to exit conveyor by sorting them according to their height (Figure 4).



Inputs (Sensors):
C0: Feeder belt exit detector, C1: Lower case detector, C2: Higher case detector, C3 Exit detector of the entry conveyor, C4-C5: Detectors of the turntable position, C6: Turntable pallet detector, C7: Entry detector of the left exit conveyor, C9: Exit detector of the left exit conveyor, C8: Entry detector of the right exit conveyor, C10: Exit detector of the right exit conveyor

Outputs (Actuators): A0: Feeder belt, A1: Entry conveyor, A2: Turntable rollers (loading), A3: Turntable rollers, A4: Turntable, A5: Left exit conveyor, A6: Right exit conveyor

Fig. 4. Virtual sorting system from ITS PLC collection

The system is instrumented using 11 sensors to determine the size of the boxes (small or large) and the entry or exit of a box in different conveyors (feeding, intermediate, evacuation) or turntable. The seven outputs of the PLC can activate the various conveyors and the turntable. The specification used is as follows. After pressing the "start" button, the boxes are sent successively one to the left elevator and one to the right elevator. After pressing the "stop" button, boxes in transit are evacuated. The safety analysis has resulted in 17 *CSs* and 5 *CSc* (table 1), formally checked using the UPPAAL model checker (Behrmann *et al.,* 2002) and the methodology proposed in (Marangé *et al.,* 2010). With this set of safety constraints (*CSs* and *CSc*), whatever the controller, the collisions between boxes and falling down of boxes, are avoided (figure 5). More information about constraints for the sorting system can be found in (Riera *et al*., 2014).

$CSs1 = 2P.A0$
$CSs2 = C3.\overline{C4}.A1$
$CSs3 = C3.C4.C6.A1$
$CSs4 = C3.P36.A1$
$CSs5 = \overline{C5}.A3$
$CSs6 = C4.C6.A2$
$CSs7 = \overline{C4}.\overline{C5}.A2$
$CSs8 = C5.C6.\overline{A4}$
$CSs9 = C5.C8.\overline{A4}$
$CSs10 = C5.C7.\overline{A4}$
$CSs11 = C5.P67.\overline{A4}$
$CSs12 = C5.C7.A2$
$CSs13 = C4.C9.A4$
$CSs14 = C4.\overline{C6}.A4$
$CSs15 = C4.P79.A4$
$CSs16 = C4.P810.A4$
$CSs17 = C4.C10.A4$

$CSc1 = C0.A0.\overline{A1}$
$CSc2 = C3.C4.A1.\overline{A2}$
$CSc3 = C5.C8.A2.\overline{A5}$
$CSc4 = C5.C7.A3.\overline{A6}$
$CSc5 = A2.A3$

Table. 1. Set of simple safety constraints (CSs) and combined safety constraints (CSc) for the sorting system

This set of constraints ensure the controllability (there is at least one controller allowing to bring boxes to the left elevator and the right elevator), and the safety regardless of the control. It should be noted that these constraints are permissive (large control space allowed) but require five observers (*2P*, *P36*, *P67, P79 and P810*). This example is interesting because the separation of safety and functional aspects simplifies a lot the control design. Indeed, from a functional point of view, the problem consists in mainly deciding if the box goes to the right or to the left and when to switch on actuators.

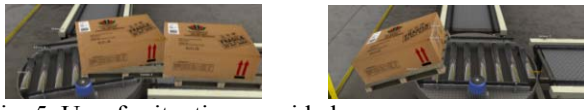
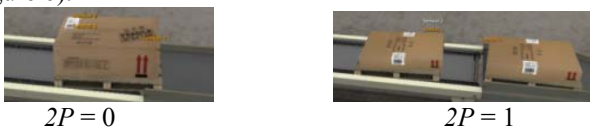


Fig. 5. Unsafe situations avoided

P36, *P67, P79, P810* are observers which allow knowing that a box is respectively present between the sensors *C3* and *C6*, *C6* and *C7, C7* and *C9*, and *C8* and *C10* (sensors excluded). For example, *P36* is set to 1 on the falling edge of the sensor *C3* and reset to 0 on a rising edge of the sensor *C6*. In this system, the distance between the sensors *C0* and *C1* is smaller than the size of a box. The observer *2P* indicates when *C0=C1=1* if 2 boxes are present and not only one (figure 6).



2P = 0 *2P* = 1

Fig. 6. Observers *2P*

Concerning *CSc*, following the path of boxes, *A2* has priority over *A1*, and *A1* has priority over *A0* (*CSc1=1* implies *A0=0*, *CSc2=1* implies *A1=0*). *A5* and *A6* have priority over *A2* and *A3* (*CSc3=1* implies *A2=0*, *CSc4=1* implies *A3=0*). At least, when *A2* = *A3* = 1, there is no priority, *A2* and *A3* are reset to *0* (*CSc5=1* implies *A2=A3=0*).

The specification of the functional part is presented figure 7 using GRAFCET (IEC60848) which can be implemented in one of the PLC languages (IEC 61131-3). The variable *cpt_conv1* is a counter which indicates the number of boxes on the entry conveyor (controlled by *A1*). *PC* is an observer whose value is complemented on a falling edge of the sensors *C7* or *C8*, and allows directing the boxes to the left elevator or the right elevator. One can notice that a complete specification in GRAFCET is much more difficult to design because safety and functional aspects have to be mixed. One can also note, that theoretically the motion of the turntable must be maintained in steps 14 and 15. This will be automatically managed by the safety guards.

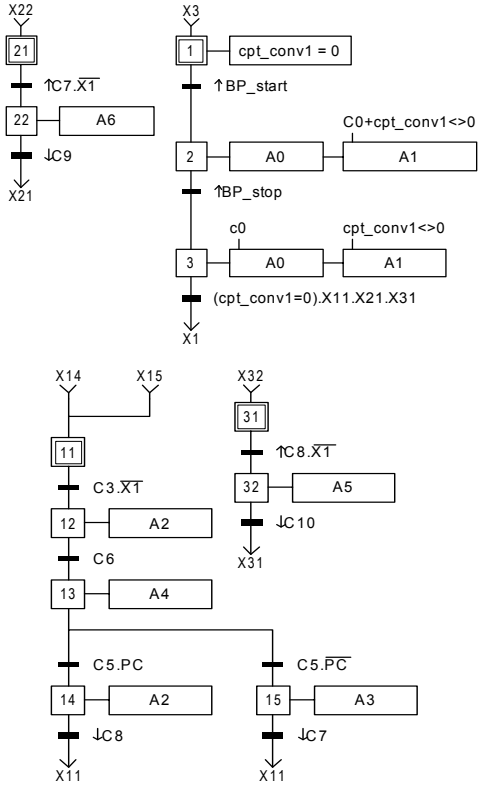


Fig. 7. Functional specification (GRAFCET) of the sorting system

The functional outputs vector (*fov*) can be defined as

follows:

$$fov = (g10, g11, g12, g13, g14, g15, g16) \quad (10)$$

With:

For A0: $g10$ = X2 or (X3 and $C0$)
For A1: $g11$ = (X2 and ($C0$ or cpt_conv1<>0)) or (X3 and cpt_conv1<>0)
For A2: $g12$ = X12 or X14
For A3: $g13$ = X15 For A4: $g14$ = X13
For A5: $g15$ = X32 For A6: $g16$ = X22 (11)

Xi is a step variable. The active or inactive state of the step may be represented by the logical values "1" or "0" respectively of a boolean variable Xi, in which i shall be replaced by the label of the relevant step. The CSP in the case of the sorting system is defined as follows:

$$O = \{o_1, \dots, o_7\} = \{A_0, \dots, A_6\}$$
$$D = \{True, False\}$$
$$Cr = \{f_{s1}, \dots, f_{s7}, CSc_1, \dots, CSc_5\} = \{Cr_1, \dots, Cr_{12}\} \quad (12)$$

With:

$$f_{sk} = f_{sk}(o_k, Y) = o_k \cdot f_{s0k} + \overline{o_k} \cdot f_{s1k} \quad (13)$$

In the case of the sorting system, one can get:

$$Cr1 = 2P.A0 \qquad Cr2 = C3.\overline{C4} + C3.C4.C6 + C3.P36.A1$$
$$Cr3 = C4.C6 + \overline{C4}.\overline{C5} + C5.C7.A2 \qquad Cr4 = C5.A3$$
$$Cr5 = (CC4.\overline{C6} + C4.P79 + C4.P810 + C4.C9 + C4.C1).A4 + (C5.C6 + C5.C8 + C5.C7 + C5.P67).\overline{A4}$$
$$Cr6 = False \qquad Cr7 = False$$
$$Cr8 = C0.A0.\overline{A1} \qquad Cr9 = C3.C4.A1.\overline{A2}$$
$$Cr10 = C5.C8.A2.\overline{A5} \qquad Cr11 = C5.C7.A3.\overline{A6} \qquad Cr12 = A2.A3 \quad (14)$$

We have used the package "logilab-constraint", an open source constraint solver written in pure Python with constraint propagation algorithms. The proposed control algorithm calculates at each scan all the safe outputs vectors and selects the one with the minimum Hamming distance compared to the functional output vector (*fov*). The control algorithm based on CSP has been implemented successfully. We did not have any problem of time calculation and a scan time of 16ms was respected for the soft PLC.

5. CONCLUSION

This paper has proposed a safe control synthesis method based on the use of safety guards (represented as a set of logical constraints which can be simple or combined) and a CSP solver. This approach to PLC programming makes safety a priority and allows for a controller to create a safe environment where functional and safety aspects are clearly separated. The idea is to use at each PLC scan time a CSP solver to get the set of all outputs vectors respecting the set of safety constraints and to select the one which is the most similar in the sense of Hamming distance of the functional outputs vector. A virtual sorting system and a soft PLC in IronPython have been used to preliminary test the idea and to get a proof of concept. The results obtained have been very encouraging. The controller code is simple and efficient. However, even if the controller is safe, it is not deterministic and it has to be proved that the minimum Hamming distance compared to the functional output vector is suitable in the sense of the specification of the functional control. This work in progress seems to be the first time that a PLC controller is based on the use in real time of a CSP solver. In this kind of applications, it could be more appropriated to use a SAT solver (Vizel *et al*., 2015). However, it is important to test the concept for different manufacturing examples and real PLC. For that, it is necessary to develop a CSP solver in ST (Structured Text) compliant with the IEC 61131-3. This one does not seem to be too complicated because the structure of the safety constraints are known and simple (monomial). However, even if the idea of using CSP solver presents several advantages, the proposed control methodology is very different from the "traditional" way to design controllers of automated production system.

REFERENCES

Behrmann G., Bengtsson J., David A., Larsen K.G., Pettersson P., Yi W. (2002). Uppaal implementation secrets. *7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems*.

Cassandras C. G., Lafortune S. (1999). *Introduction to discrete event systems*. Boston, MA: Kluwer Academic Publishers.

Hooker J (2000). *Logic-Based Methods for Optimization - Combining Optimization and Constraint Satisfaction.* Wiley-Interscience series in discrete mathematics and optimization. John Wiley and Sons, 2000.

IEC INTERNATIONAL STANDARD 61131-3 (2003). *Programmable controllers – Part 3: Programming languages.* Reference number CEI/IEC 61131-3: 2003.

IEC INTERNATIONAL STANDARD 60848 (2002). *GRAFCET specification language for sequential function charts*. Reference number CEI/IEC 60848: 2002.

Krzysztof A. (2003). *Principles of Constraint Programming.* Cambridge University Press, ISBN: 0521825830, New York, NY, USA.

Marangé P., Benlorhfar R., Gellot F., Riera B. (2010). Prevention of human control errors by robust filter for manufacturing system, *11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France.

Riera B., Coupat R., Annebicque D., Philippot A., Gellot F. (2014). Control synthesis based on safety boolean guards for manufacturing systems: application to a sorting system, *10th International Conference on Modeling, Optimization & SIMulation (MOSIM'2014)*, Nancy, France, November 2014.

Riera B., Philippot A., Coupat R., Gellot F., Annebicque D. (2015). A non-intrusive method to make safe existing PLC Program, *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS'15)*, Paris, France, September 2015.

Tsang E.P.K. (1993). *Foundations of Constraints Satisfaction*, Academic Press Limited, UK, 1993.

Vizel, Y.; Weissenbacher, G.; Malik, S. (2015). *Boolean Satisfiability Solvers and Their Applications in Model Checking*. Proceedings of the IEEE 103 (11). doi:10.1109/JPROC.2015.2455034.