Control design pattern based on safety Boolean guards for manufacturing systems: application to a palletizer

B. RIERA*, R. COUPAT*, A. PHILIPPOT*, D. ANNEBIQUE**, and F. GELLOT***

** CReSTIC (EA3804), UFR Sciences Exactes et Naturelles, Reims University (URCA), Moulin de la Housse, BP 1039, 51687 Reims - France (bernard.riera@univ-reims.fr).*
*** CReSTIC (EA3804), IUT de Troyes, 9 rue de Québec, BP 396, 10026 TROYES, Cedex, France*
****PSIGT-TE (CES), Direction de l'Ingénierie, Société Nationale des Chemins de Fer Français, 6, avenue François Mitterrand – 93574 La Plaine Saint Denis CEDEX, France*

Abstract: This paper presents an original approach for safe controller design for manufacturing systems controlled by PLC (Programmable Logic Controller). In this work, manufacturing systems are considered as Discrete Event Systems (DES) with logical Inputs (sensors) and logical Outputs (actuators). The proposed approach, which separates the functional control part from the safety control part, is easy to implement and ensures that the designed controller is safe. The methodology is based on the use of safety constraints in order to get a permissive safe controller which is validated by model-checking. This controller is then constrained by functional constraints. The approach is illustrated with a palletizer simulated process using the ITS PLC software from the Real Games Company (www.realgames.pt). The control algorithm is presented and allows resulting in a safe control using a standard control design pattern, may be simpler than a conventional approach based on a complete specification in GRAFCET (IEC 60848) that does not distinguish the functional aspect from the safety aspect. This approach presents interesting perspectives like the management of several operating modes linked to a Manufacturing Execution System (MES) or the manual modes through Human-Machine Interfaces (HMI) or Supervisory Control and Data Acquisition (SCADA) systems.

***Keywords**:* Discrete-Event Systems, 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) [Cassandra *et al.* 1999] with logical Inputs (sensors) and logical Outputs (actuators). This is an extension of the research work that the CReSTIC (Research Centre in Information and Communication Science and Technologies) has led for several years on the definition and design of guard conditions 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 are formally checked off line by using a model checker. The proposed approach, which separates the functional control part from the safety control part, enables to get a control design pattern easy to implement and ensuring that the designed controller is safe. The methodology is based on the use of safety constraints in order to get the most permissive safe controller. This controller is then constrained by functional constraints. This paper proposes several improvements of the control algorithm presented in [Riera *et al.* 2012] particularly in the management of combined safety constraints. The approach is illustrated by using one example: a virtual palletizer using the ITS PLC software from the Real Games Company (www.realgames.pt). This control synthesis approach allows to result in a safe control, may be simpler than a conventional approach based on a complete specification in GRAFCET (IEC 60848) that does not distinguish the functional aspect from the safety aspect. This approach presents interesting perspectives like the management of several operating modes linked to a Manufacturing Execution System (MES) or the manual modes through Human-Machine Interfaces (HMI) or Supervisory Control and Data Acquisition (SCADA) systems.

2. BOOLEAN SAFETY CONSTRAINTS FOR ROBUST PLC CONTROL

Since a PLC is a dedicated controller, it will only process the program over and over again. One cycle through the program is called a scan time and involves reading the inputs from the other modules, executing the logic based on these inputs and then updated the outputs accordingly. The memory in the CPU stores the program while also holding the status of the I/O and providing a means to store values. The notations used in the following of this 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 (actuators) at t.
- $o_k^* = o_k(t-1)$: logical variable corresponding to the value of k^{th} PLC Boolean output (actuators) at time $t-1$ (previous scan time).

- $i_j = i_j(t)$: logical variable corresponding to the value of j^{th} PLC Boolean input (sensors) at time t.
- $i_j^* = i_j(t-1)$ logical variable corresponding to the value of j^{th} PLC Boolean input (sensors) at time t-1.
- ".", "+", "¯" are respectively the logical operators AND, OR, and NOT.
- 0 means FALSE and 1 means TRUE.
- $\sum$ and π are respectively the logical sum (OR) and the logical product (AND) of logical variables.
- $\uparrow x$ means rising edge of Boolean variable x (in the PLC, $\uparrow x = \overline{x^*}.x$).
- $\downarrow x$ means falling edge of Boolean variable x (in the PLC, $\downarrow x = x^*.\bar{x}$).
- O: set of output variables at t
- O^*: set of output variables at t-1
- I: set of input variables at t, t-1, t-2 ...
- OBS: set of observers at t, t-1, t-2 ...
- CS: set of safety constraints.
- N_o : number of PLC Boolean outputs
- N_i: number of PLC Boolean inputs
- N_{CSs}: number of Simple Safety Constraints
- N_{CSc}: number of Combined Safety Constraints

The proposed methodology to design safe controllers is based on the use of safety constraints, which act as logical guards placed at the end of the PLC program, and forbids sending unsafe control to the plant [Marangé *et al.* 2010].

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 changes of the input vector (synchronous operation, possible simultaneous changes of state of PLC inputs). In addition, the plant is considered functioning normally without failure.

In this approach, safety constraints are expressed in the form of a logical monomial function (product of logical variables, as π) which must always be equal to 0 (FALSE) at each PLC scan time in order to guarantee the safety. It is considered in this work that the initial safe state for all the actuators (o_k) is defined to 0.

Initially, the constraints are defined in order to ensure a permissive control, and it is assumed that, with the filter, the system remains controllable. In other words, it is possible to design a controller which matches the specifications. For example, considering the previous hypothesis about the safe initial state, a filter which resets 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). Hence, the filter requires a memory function.

It may be necessary to define observers due to the lack of system observability. This is especially true when there are flows of products. Observers correspond ideally to a sequential function of PLC inputs and allow coming back to a combinatory guard.

The set of constraints CS is considered as necessary and sufficient to guarantee the safety.

In the presented approach, it is assumed that the safety constraints can always be represented as a monomial and depend on the inputs (at *t*, *t-1*, *t-2*…), outputs (at *t*, *t-1*, *t-2*…) and observers (depending ideally on only inputs at *t*, *t-1*, *t-2*…). In the initial methodology, the filter stops the process in a safe state if a safety constraint is not respected.

CSs and *CSc* can be represented respectively by equation (1) and equation (2) which are Boolean monomial functions but not necessarily minterms.

$$CSs_i(o_k) = \pi_i(o_k, \mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \quad (1)$$

$$CSc_i(o_k, o_l, \ldots) = \pi_i(o_k, o_l, \ldots, \mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \quad (2)$$

There are only 2 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 (3) or (4)):

$$CSs_i(o_k) = o_k.f_i(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \quad (3)$$

or

$$CSs_i(o_k) = \overline{o_k}.f_i(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \quad (4)$$

These Simple Safety Constraints *CSs* express the fact that if $f_i(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*)$, which is a monomial function, is TRUE, o_k must be necessarily FALSE ($o_k.f_i(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*)$) or TRUE ($\overline{o_k}.f_i(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*)$) in order to keep the constraints equal to 0.

For each output o_k, it is possible to write the sum of simple safety constraints following equation (5) where f_{s0k} and f_{s1k} are polynomial (as $\sum\pi$) functions of I (inputs at *t*, *t-1*, *t-2* ...), O^* (previous outputs) and OBS (observers at *t*, *t-1*, …). Equation (5) has always to be FALSE because all the simple safety constraints must be FALSE at each PLC scan time.

$$\sum_{i=1}^{N_{CSs}} CSs_i = \sum_{k=1}^{N_o}\left(o_k.f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) + \overline{o_k}.f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*)\right) = 0 \quad (5)$$

It is important to note that the Simple Safety Constraints have to respect the following mathematical property (equation 6):

$$f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*).f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) = 0 \quad (6)$$

Indeed, if it is not the case, that means that 2 *CSs* are in contradiction and one of both is not respected, thus the set of constraints is not coherent. One can notice that if fs_{0k} =0 or if fs_{1k} =0, the property is logically verified. In addition, if all $CSs_j(o_k)$ are only based on the rising edge and falling edge of the output o_k, the property is always TRUE (sufficient condition).

Proof: if for the output o_k, all $CSs_j(o_k)$ are only based on rising edge and falling edge, one can notice using the Shannon expansion theorem that:

$$f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) = \overline{o_k{}^*}.f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \text{ and } f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) = o_k{}^*.f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) \quad (7)$$

Consequently, because $\overline{o_k{}^*}.o_k{}^* = 0$, and the initial state safe, the property is respected.

$$f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*).f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) = \overline{o_k{}^*}.f_{s0k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*).o_k{}^*.f_{s1k}(\mathrm{I}, \mathrm{OBS}, \mathrm{O}^*) = 0 \quad (8)$$

3. SAFE CONTROL SYNTHESIS FROM LOGICAL CONSTRAINTS

The control algorithm proposed separates safety requirements from functional requirements. A set of safety constraints is considered as necessary and sufficient. In other words, if only one safety constraint is removed, the system is unsafe. All other constraints that can be added are considered as functional constraints because they don't act on safety. The synthesis algorithm consists in authorize by default everything that is not prohibited and be force by the functional requirements. In order to present the idea, let's consider a system without *CSc*.

3.1 Taking into account the CSs and Functional specification

From the equation (5), and for each output, it is possible to write equation (9) corresponding to a logical OR of all simple safety constraints.

$$\sum_{i=1}^{N_{CSs}} CSs_i = \sum_{k=1}^{N_o} \left(f_{sk}(o_k, I, OBS, O^*) \right) = 0 \quad (9)$$

$f_{sk}(\mathrm{o_k, I, OBS, O^*})$ is a logical function independent of the other outputs at *t* because only *CSs* are considered. Given what has been stated in part 2 (equation 5), it is possible to write equation (10).

$$f_{sk}(o_k, I, OBS, O^*) =$$
$$o_k . f_{s0k}(I, OBS, O^*) + \overline{o_k} . f_{s1k}(I, OBS, O^*) = 0 \quad (10)$$

f_{s0k} and f_{s1k} are polynomial (as $\sum\pi$) functions of I (inputs), OBS (observers) at times *t*, *t-1*, …, and O* (previous inputs).

It is possible to define the functional constraints (*FC*) of o_k indicating when it should be equal to 0 (g_{0k}) or when the output o_k should be equal to 1 (g_{1k}) (equation (11)) from a functional point of view. In this paper only Simple Functional Constraints *FCs* are considered.

$$g_k(o_k, I, OBS, O^*) = o_k . g_{0k} \ (or) \ g_k(o_k, I, OBS, O^*) = \overline{o_k} . g_{1k} \quad (11)$$

Generally the specifications indicate when the output must be activated and therefore g_{1k}. These g_{1k} can of course include observers obtained from GRAFCET steps (IEC 60848) or SFC (IEC 61131-3). By incorporating results by Hietter [2008] on algebraic synthesis of dependable logic controllers, it is possible to write (equation 12) the parametric solution (called o'_k) of the equation 10.

$$o'_k = \overline{f_{sok}} + f_{s1k} . p \quad (12)$$

Where *p* is a Boolean parameter. In order to guarantee the safety by integrating the *FCs*, the solution becomes equation (13):

$$o'_k = \overline{f_{sok}} . g_{1k} + f_{s1k} \ or \ \ o'_k = \overline{f_{sok}} . \overline{g_{ok}} + f_{s1k} \quad (13)$$

The control obtained is safe (if there are only *CSs*) because the safety is ensured regardless of the *FCs*. Indeed, if the *FCs* try to impose an output to 0, in contradiction with the safety, the term f_{s1k} continues to provide safety. Therefore the functional part can be designed without considering safety, what makes the job much easier for the control engineer.

In the next part of the paper, it is shown how to deal with the Combined Safety Constraints (*CSc*). We will only consider *FCs* defined by g_{1k}.

3.2 Taking into account the CSc

The problem with *CSc* seems to be more complex. Indeed, when a *CSc* is not respected, it is necessary to give the priority to one or several outputs. However, the *CSc* have always to be coherent with the *CSs*, which are most priority because they only depend on inputs and observers which are uncontrollable events. In addition, when one *CSc* is solved, it can involve problems in other *CSc*. Taking into account these points, and using equation (13), it is possible to write equation (14).

$$o_k = \overline{f_{sok}} . \left(\overline{f_{c0k}} . g_{1k} + f_{c1k} \right) + f_{s1k} \quad (14)$$

f_{c0k} and f_{c1k} force the output o_k to 0 or 1 taking into account *CSc*. It is supposed *CSc* have to be designed in order to give always the same priority to outputs. What the reader has to notice, it is that during the PLC scan time, a safe value of o_k has to be found. This means that the value of o_k has to be compliant with all *CSc* implying o_k. If f_{c0k} and f_{c1k} are badly defined, a safe value of o_k can be impossible to compute. To illustrate this problem, let's take a simple example. Suppose the 2 following *CSc* (equation 15):

$$CSc_1 = o_1 . \overline{o_2}; \ CSc_2 = o_2 . \overline{o_3} \quad (15)$$

If when CSc_1 is TRUE the priority is given to o_1 and when CSc_2 is TRUE the priority is given to o_3, if o_1=1 and o_3=0, it is impossible to find a safe value of o_2. We propose here a simple solution to detect this problem. The idea is to check that during the PLC scan time one *CSc* is not violated at least more than 2 times. That will be the case if after having tried to find a solution (N_{CSc}+1) times, you do not get a solution. Hence, this means there is a problem of definition of *CSc*. In this case, the priority has to be given to *CSs*. Let's define $\overrightarrow{f_{c0}}$ and $\overrightarrow{f_{c1}}$ as column vectors representing respectively the *k* values of f_{c0k} and f_{c1k} . $\overrightarrow{f_{c0}}$ and $\overrightarrow{f_{c1}}$ can be obtained through 2 matrices *MC0* and *MC1* that the control engineer has to define during the initial safety analysis stage defining the priority between outputs. *MC0* and *MC1* are matrices with N_{CSc} columns and N_o lines and indicate for each *CSc*, if the outputs have to be forced respectively to 0 or 1. Using the matrix logical product, one can write equations (16 and 17).

$$\overrightarrow{CSc} = \begin{pmatrix} CSc_1 \\ \dots \\ CSc_{N_{CSc}} \end{pmatrix}, \text{column vector of } CSc$$

$$\vec{O} = \begin{pmatrix} o_1 \\ \dots \\ o_k \\ \dots \\ o_{N_o} \end{pmatrix}, \text{column vector of outputs } o_k$$

$$\overrightarrow{f_{c0}} = \begin{pmatrix} f_{c01} \\ \dots \\ f_{c0N_o} \end{pmatrix} = MC0.\overrightarrow{CSc}$$

$$\overrightarrow{f_{c0}} = \begin{pmatrix} MC0_{11} & \dots & MC0_{1N_{CSc}} \\ \dots & & \dots \\ \dots & \dots & \dots \\ MC0_{N_o1} & \dots & MC0_{N_oN_{CSc}} \end{pmatrix} . \begin{pmatrix} CSc_1 \\ \dots \\ CSc_{N_{CSc}} \end{pmatrix} \quad (16)$$

$$\overrightarrow{f_{c1}} = \begin{pmatrix} f_{c11} \\ \dots \\ f_{c1N_o} \end{pmatrix} = MC1.\overrightarrow{CSc}$$

$$\overrightarrow{f_{c1}} = \begin{pmatrix} MC1_{11} & \cdots & MC1_{1N_{CSc}} \\ \cdots & \cdots & \cdots \\ MC1_{N_o1} & \cdots & MC1_{N_oN_{CSc}} \end{pmatrix} . \begin{pmatrix} CSc_1 \\ \cdots \\ CSc_{N_{CSc}} \end{pmatrix} \quad (17)$$

Figure 1 presents the algorithm which is detailed in order for the reader to be able to implement it in a PLC in ST language (IEC 611131-3).

// g_{1k} are calculated previously (functional constraints, FC) as all the observers (O) in the program. MC0 and MC1 for the CSc are known
// Each o_k, f_{s0k}, f_{s1k} are calculated at each scan PLC
// check that the CSs respect $f_{s0k}.f_{s1k}$ = FALSE
// init f_{c0k} and f_{c1k}
Flag_CSs = FALSE
For k=1 **to** N_o
Flag_CSs = Flag_CSs + $f_{s0k}.f_{s1k}$
f_{c0k} = False // INIT
f_{c1k}= False //INIT
End for
Flag = not Flag_CSs
Cpt =0 // counter for the CSc

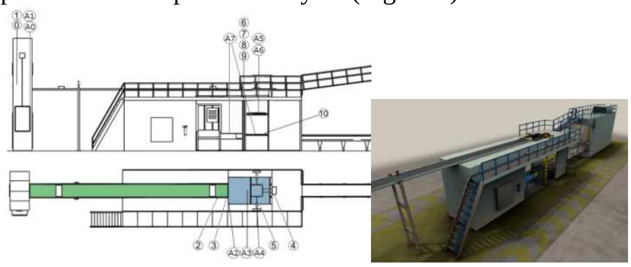
While (Flag **and** Cpt<N_{CSc}**)**
// each o_k is calculated using $o_k = \overline{f_{sok}}.(\overline{f_{c0k}}.g_{1k} + f_{c1k}) + f_{s1k}$
// ol_k is the intermediary value of o_k
For k=1 **to** N_o
$ol_k = \overline{f_{sok}}.(\overline{f_{c0k}}.g_{1k} + f_{c1k}) + f_{s1k}$
End For
// check if a CSc is violated
Flag = FALSE
For i=1 **to** N_{CSc}
Calculate CSci by using ol_k values
Flag = Flag + CSci
End For
Cpt= Cpt +1
// if CSc=TRUE, priority is given to a o_k using MC0 and MC1
For k=1 **to** N_o
f_{c0k} = FALSE
f_{c1k} = FALSE
For j=1 **to** N_{CSc}
$f_{c0k} = f_{c0k} + MC0_{kj}.CSc j$
$f_{c1k} = f_{c1k} + MC1_{kj}.CSc j$
End For
End For
End While
If flag_CSs **Then**
print "PROBLEM BAD DEFINITION CSS"
Break // STOP with problem
End If
If (cpt= =N_{CSc}) **Then**
print "PROBLEM BAD DEFINITION CSC"
// in case of bad definition of CSc, o_k are set
// to a safe value, with a priority to FALSE
For k=1 **to** N_o
$ol_k = \overline{f_{sok}}.f_{s1k}$
End For
End If
// The outputs are set with safe values
For k=1 **to** N_o
$o_k = ol_k$
End For

Fig. 1. Safe controller algorithm.

This algorithm is quite original because one can see there is a "while" structure inside the PLC program in order to manage the *CSc*. This is something which is not seen as a good practice for PLC programmer using LADDER. This algorithm is simple and the program structure is always the same whatever the system to be controlled and its specification. Even if the functional constraints are wrong, the system remains safe. In addition, if the *CSc* are incoherent, the system will be maintained in a safe state if *CSs* are well defined.

4. EXAMPLE ON A PALLETIZER

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 [Riera *et al.* 2009]. 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 "palletizer system" is used. The palletizer system is composed of a case elevator, a central body and an exit bay. The objective of this system is to palletize cases up to three layers (Figure 2).



Inputs (Sensors): i_0: Exit detector of the case elevator, i_1: Limit switch of the cursor advance movement, i_2: Detector of the conveyor belt buffer, i_3: Mat limit switch, i_4: Mat limit switch, i_5: Packing rods limit switch, i_6: Elevator low level detector, i_7: Elevator level detector - first cases layer, i_8: Elevator level detector - second cases layer, i_9: Elevator level detector - third cases layer, i_{10}: Pallet detector.

Outputs (Actuators (noted *A* in the figure 2)): o_0: Case elevator, o_1: Elevator cursor advance, o_2: Holder opening, o_3: Mat advance, o_4: Packing rods advance, o_5: Ascending movement of the pallet elevator, o_6: Descending movement of the pallet elevator, o_7: Conveyor tables.

Fig. 2. Virtual palletizing system from ITS PLC collection

The case elevator feeds an automatic conveyor belt through a cursor. The cases are accumulated at the end of the conveyor belt by a holder. At this stage the cases are ready to be loaded on the mat and transported to the packing rods. The conveyor tables drive the pallets from the pallet feeder to the elevator. The elevator, loaded with a pallet, ascends to the upper palletizer level. The cases are palletized with the returning of the mat and with the packing rods at the forward position. This palletizing cycle can be repeated one or two more times, per pallet. After the cases get palletized, the elevator goes down to the level of the exit conveyor table.

Based on a FMEA (Failures Modes and Effects Analysis) method not presented in the paper, the safety analysis has resulted in 30 *CSs* (presented separately for each actuator) and 9 *CSc*, that could be formally checked using the UPPAAL model checker [Behrmann *et al.*, 2002] applying the methodology proposed in [Riera *et al.* 2011, Marangé *et al.* 2010]. With this set of safety constraints (*CSs* and *CSc*), whatever the controller, unsafe situations are avoided (figure 3). This set of constraints ensures the controllability (there is at least one controller allowing to bring couple of cases on the pallet, level by level). It should be noted that these constraints are permissive (large control space allowed) and

require 4 observers (*Pp*, *P12*, *P2*, *cpt*).
Pp allows knowing if a pallet is loaded on the elevator, *P12* indicates if there is a case between i_1 and i_2, *P2* indicates if there are two cases waiting at the holder. Observer *cpt* represents the actual number of layers on the pallet.
This example is interesting because the separation of safety and functional aspects simplifies a lot the design of the controller. From a functional point of view the problem consists only in palletizing the cases six by six (three layers).

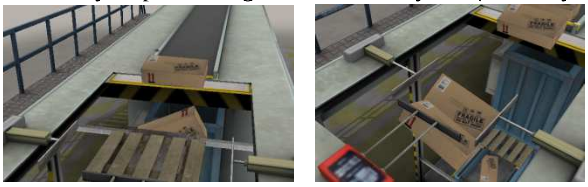


Fig. 3. Examples of unsafe situations avoided

The specification of the functional part is presented figure 4 using GRAFCET (IEC60848) which is also easy to implement in ST PLC language (IEC 61131-3). One can notice that a complete specification in GRAFCET is much more difficult to get and to read because safety and functional aspects have to be mixed. The management of the different running modes is not described in the paper. The Boolean variable *RM* indicates the normal running mode, it is to say that 3 levels of cases have to be placed on the pallets. Now it is possible to write f_{s0k}, f_{s1k}, g_{1k} for each actuator o_k from the *CSs*. Concerning g_1, they are very easy to write. For instance, o_3 from a functional point of view has to be activated only when step X41 or step X42 are active. Consequently, $g_{13} = X41 + X42$.

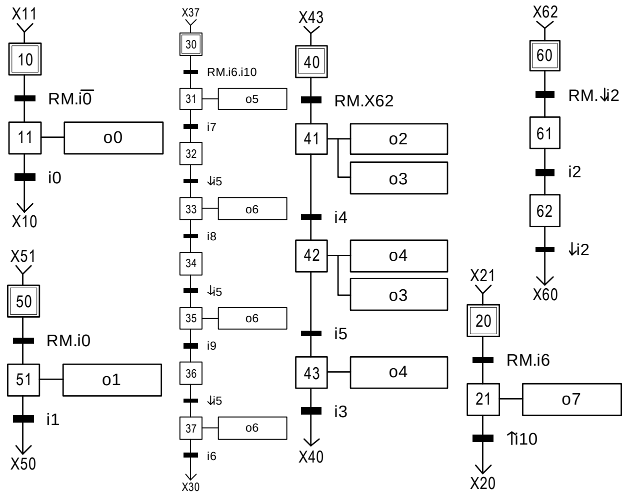


Fig. 4. Functional specification of the palletizer system

From $CSs_1 = \downarrow o_0.i_0^*.i_0$, one can write for actuator o_0:

$$o_0 \begin{cases} f_{s00} = 0 \\ f_{s10} = o_0^*.\ i_0^*.i_0 \qquad g_{10} = X11 \end{cases}$$

The actuator o_0 can only be stopped on a rising edge of i_0 in order to avoid bad ejection of the cases on the conveyor belt.

From $s_2 = o_1.i_1$; $CSs_3 = \uparrow o_1.P12$; $CSs_4 = \uparrow o_1.i_2$; $CSs_5 = \uparrow o_1.\overline{i_0}$; $CSs_6 = \downarrow o_1.\overline{i_1}$, one can write for o_1:

$$o_1 \begin{cases} f_{s01} = i_1 + \overline{o_1^*}.\ (P12 + i_2 + \overline{i_0}) \\ f_{s11} = o_1^*.\overline{i_1} \qquad g_{11} = X51 \end{cases}$$

These guards force actuator o_1 to extend to i_1 and after to retract immediately. In addition o_1 cannot be set if there isn't a box on i_0 or if there are 2 cases on the conveyor belt.

For o_2, from $CSs_7 = \downarrow o_2.\overline{i_4}$; $CSs_8 = \uparrow o_2.\overline{P2}$; $CSs_9 = \uparrow o_2.\overline{i_7}.\overline{i_8}.\overline{i_9}$; $CSs_{10} = \uparrow o_2.i_7.cpt > 0$; $CSs_{11} = \uparrow o_2.i_8.cpt > 1$; $CSs_{12} = \uparrow o_2.i_9.cpt > 2$; $CSs_{13} = \uparrow o_2.\overline{i_3}$

$$o_2 \begin{cases} f_{s02} = \overline{o_2^*}.\ \begin{pmatrix} \overline{P2} + \overline{i_7}.\overline{i_8}.\overline{i_9} + i_7.cpt > 0 \\ + i_8.cpt > 1 + i_9.cpt > 2 + \overline{i_3} \end{pmatrix} \\ f_{s12} = o_2^*.\ \overline{i_4} \qquad g_{12} = X41 \end{cases}$$

Output o_2 cannot be set if there are not 2 cases waiting at the holder and if the table is not positioned in (i_3). It cannot be activated to let the cases go on the table if the elevator is not well positioned (i_7, i_8, i_9) according to the layers on the pallet (*cpt*). Output o_2 *is* activated until i_4 is TRUE, meaning the cases leave with certitude the conveyor belt.
For o_3, from $CSs_{14} = \uparrow o_3.i_5$; $CSs_{15} = \downarrow o_3.\overline{i_4}$; $CSs_{16} = \downarrow o_3.\overline{i_5}$

$$o_3 \begin{cases} f_{s03} = \overline{o_3^*}.\ i_5 \\ f_{s13} = o_3^*.\ (\overline{i_4} + \overline{i_5}) \qquad g_{13} = X41 + X42 \end{cases}$$

Output o_3 can only be reset if the table is fully out (i_4) and when the couple of cases is hold by the packing rods (i_5). In addition, o_3 cannot only be set if the packing rods are not out.
For o_4 from $CSs_{17} = \uparrow o_4.\overline{i_4}$; $CSs_{18} = o_4.i_3$; $CSs_{19} = \downarrow o_4.\overline{i_5}.\overline{i_3}$

$$o_4 \begin{cases} f_{s04} = \overline{o_4^*}.\overline{i_4} + i_3 \\ f_{s14} = o_4^*.\ \overline{i_5}.\overline{i_3} \qquad g_{14} = X42 + X43 \end{cases}$$

Output o_4 can be activated only if i_4 is TRUE and cannot be disabled while the packing rods are not fully out (i_5) and hold the cases and while the table is not fully in (i_3).

For o_5 from $CSs_{20} = o_5.i_7$; $CSs_{21} = \uparrow o_5.\overline{Pp}$; $CSs_{22} = \downarrow o_5.\overline{i_7}$

$$o_5 \begin{cases} f_{s05} = i_7 + \overline{o_5^*}.\overline{Pp} \\ f_{s15} = o_5^*.\ \overline{i_7} \qquad g_{15} = X31 \end{cases}$$

Output o_5 cannot be activated only if the elevator is not at the top (i_7) and if a pallet is on the elevator (*Pp*). These safety guards do not authorize the elevator to stop before the top (i_7).

For o_6 from: $CSs_{23} = \uparrow o_6.\overline{i_3}$; $CSs_{24} = \uparrow o_6.i_7.cpt <> 1$; $CSs_{25} = \uparrow o_6.i_8.cpt <> 2$; $CSs_{26} = \uparrow o_6.i_9.cpt <> 3$; $CSs_{27} = \uparrow o_6.\overline{i_7}.\overline{i_8}.\overline{i_9}$; $CSs_{28} = \downarrow o_6.\overline{i_6}.\overline{i_8}.\overline{i_9}$

$$o_6 \begin{cases} f_{s06} = \overline{o_6^*}.\begin{pmatrix} \overline{i_3} + i_7.cpt <> 1 + i_8.cpt <> 2 + \\ i_9.cpt <> 3 + i_9.cpt <> 3 + \overline{i_7}.\overline{i_8}.\overline{i_9} \end{pmatrix} \\ f_{s16} = o_6^*.\ \overline{i_6}.\overline{i_8}.\overline{i_9} \qquad g_{16} = X33 + X35 + X37 \end{cases}$$

Output o_6 cannot be activated if the number of layers (*cpt*) on the pallet is not correct with regard to the current position of the elevator (i_7, i_8, i_9). Output o_6 is maintained activated until the elevator reaches the next position sensor to the bottom (i_6, i_8, i_9).

For o_7 from $CSs_{29} = \downarrow o_7 . i^*_{10} . i_{10} . i_6$; $CSs_{30} = o_7 . \overline{i_6}$

$$o_7 \begin{cases} f_{s07} = \overline{i_6} \\ f_{s17} = o_7^* .\ i^*_{10} . i_{10} . i_6 \qquad g_{17} = \mathrm{X21} \end{cases}$$

Output o_7 cannot be activated if the elevator is not at the lowest level (i_6) and cannot be disabled until the pallet is correctly positioned on the elevator ($i_6 . i_{10}$). The conveyor table can be stopped on a rising edge of i_{10} in order to be well placed. After the definition of f_{s0k} , f_{s1k} based on *CSs* and g_{1k}, let's present the *CSc*:

$CSc_1 = \uparrow o_1 . o_2$; $CSc_2 = o_5 . o_7$; $CSc_3 = o_6 . o_7$;
$CSc_4 = \uparrow o_2 . \overline{o_3}$; $CSc_5 = \uparrow o_3 . \overline{o_2}$; $CSc_6 = o_3 . o_5$;
$CSc_7 = o_3 . o_6$; $CSc_8 = o_2 . o_5$; $CSc_9 = o_2 . o_6$;

Concerning *CSc*, the priority is given as it is defined in *MC0* (equations 19 and 20). Output o_2 has priority over o_1 in CSc_1 so o_1 is forced to 0. If this case occurs it means that there are already two cases waiting at the holder and a third case is going to be placed on the conveyor belt. Output o_5 has priority over o_7 ($CSc_2=1$ implies $o_7=0$). Output o_6 has priority over o_7 ($CSc_3=1$ implies $o_7=0$). CSc_4 and CSc_5 imply that o_2 and o_3 must be set together. Outputs o_5 and o_6 have priority over o_3 ($CSc_6=1$ implies $o_3=0$, $CSc_7=1$ implies $o_3=0$). Outputs o_5 and o_6 have priority over o_2 ($CSc_6=1$ implies $o_2=0$, $CSc_7=1$ implies $o_2=0$). All these constraints give priority to the elevator.

$$\mathrm{MC0} = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1 \\ 0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0 \end{pmatrix} \qquad \mathrm{MC1} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \tag{19}$$

$$\overrightarrow{CSc} = \begin{pmatrix} CSc_1 \\ CSc_2 \\ CSc_3 \\ CSc_4 \\ CSc_5 \\ CSc_6 \\ CSc_7 \\ CSc_8 \\ CSc_9 \end{pmatrix} \qquad \vec{O} = \begin{pmatrix} o_0 \\ o_1 \\ o_2 \\ o_3 \\ o_4 \\ o_5 \\ o_6 \\ o_7 \end{pmatrix} \tag{20}$$

The control algorithm has been implemented successfully in a PLC with version ITS PLC PE using a PLC M340 from Schneider Electric.

5. CONCLUSION

This paper dealt with an original control synthesis method based on the use of safety guards (represented as a set of logical constraints which can be simple or combined). The control obtained is safe even if the functional constraints are wrong because only one control respecting the safety constraints is allowed. Contrary to SCT (supervisory control theory, [Ramadge *et al.*, 1989]) approach, the algorithm has been designed to be implemented in a PLC. The separation of the "safety" and "functional" aspects enables to get a control design pattern and suggests interesting perspectives like better process performances and flexibility, easier management of several operating modes linked to a Manufacturing Execution System (MES) or simpler management of the manual modes through Human-Machine Interfaces (HMI) or Supervisory Control and Data Acquisition (SCADA) systems. In addition, the prospects of this work also seem to be important because the obtained results could change the "traditional" way to design controllers of automated production system.

ACKNOWLEDGEMENT

This work is integrated in the frame of the CPER project MOSYP and is supported by Région Champagne-Ardenne, FRANCE.

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*.

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

Hietter Y., Roussel J.-M., Lesage J.-J. (2008). Algebraic synthesis of dependable logic controllers *17th IFAC World Congress, Seoul (Korea)*, pp. 4132-4137, July.

IEC INTERNATIONAL STANDARD 60848 (2002), Second edition 2003-01, Programmable controllers – Part 3: Programming languages GRAFCET specification language for sequential function charts Reference number CEI/IEC 60848: 2002.

IEC INTERNATIONAL STANDARD 61131-3 (2003), Second edition 2002-02, GRAFCET specification language for sequential function charts Reference number CEI/IEC 61131-3: 2003.

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.

Ramadge G., Wonham W. M. (1989). The control of discrete event systems, Proc. IEEE, *Special issue on DEDSs*, 77, pp.81-98.

Riera B., Marangé P., Gellot F., Nocent O., Magalhaes A., Vigario B. (2009). Complementary usage of real and virtual manufacturing systems for safe PLC training, *8th IFAC Symposium on Advances in Control Education (ACE'09)*. Kumamoto, Japan.

Riera B., Benlorhfar R., Annebicque D., Gellot F. , Vigario B., (2011). Robust control filter for manufacturing systems: application to PLC training, *18th World Congress of the International Federation of Automatic Control* , Milano, Italy.

Riera B., Annebicque D., Gellot F., Philippot A. and Benlorhfar R. (2012). 14th IFAC Symposium on INformation COntrol problems in Manufacturing (INCOM 2012), Bucarest, Romania, mai 2012.