Consistency Checking of Safety Constraints for Manufacturing Systems with Graph Analysis

R. Pichard, A. Philippot, Bernard Riera

Consistency Checking of Safety Constraints for Manufacturing Systems with Graph Analysis

R. PICHARD * **A. PHILIPPOT** * **B. RIERA** *

* *CReSTIC (EA3804), UFR Sciences Exactes et Naturelles, Reims University (URCA), Moulin de la Housse, 51687 Reims - France (romain.pichard@univ-reims.fr, firstname.name@univ-reims.fr).*

Abstract: This work deals with safe control of manufacturing systems controlled by Programmable Logic Controller (PLC). The used approach is based on Boolean safety constraints, which act as a safety filter, in order to guarantee the safety on-line. To ensure the safety whatever the inputs and the control program, the safety constraints must be checked formally. In this paper we proposed a formal off-line approach to check and ensure the consistency. The approach uses graph representation of the safety constraints. Thanks to graph algorithm, it is possible to detect inconsistencies and to help designers to solve them. Previous works on safety filter have proposed necessary conditions but none sufficient conditions, in this paper we proposed a necessary and sufficient condition to check and ensure the consistency. The safety filter approach and the consistency problem are presented. Then the approach is illustrated and a discussion around the application to manufacturing system is done.

Keywords: Boolean algebra, Safety filter, Discrete-Event Dynamic Systems, formal methods, consistency.

1. INTRODUCTION

The safety of a system includes the protection of human, systems, equipment, and the environment (II, 2015). Due to the automation and robotisation of manufacturing systems, the risks for human and plant increase. In order to prevent them as far as possible, automatic control engineers have to design control programs taking into account the safety. But regarding the complexity (number of variables, interaction of sub-systems, etc.), automatic control engineers do not have efficient formal approach to design safe control programs.

In this work, manufacturing systems are considered as Discrete-Event Dynamic Systems (DEDS) (Cassandras and Lafortune, 2009) with logical inputs and logical outputs. Moreover manufacturing systems are mostly controlled by Programmable Logic Controller (PLC), programmed with standardized languages (61131-3, 2003).

Ramadge and Wonham (1989) described a framework to synthesize a supervisor for DEDS: Supervisory Control Theory (SCT). SCT is based on automaton models (language theory) of plant and specification, from these models a supervisor is computed automatically. But in practice, this leads to a state explosion that restricts the applicability of SCT to small systems (Charbonnier et al., 1999) (Vilela and Pena, 2016). Finally SCT is based on the asynchronous hypothesis (one event at each time), but in a PLC several variables may change simultaneously, so the asynchronous hypothesis is not guaranteed.

To solve these problems, some approaches are based on Boolean algebra instead of automaton models and language theory. Algebraic synthesis introduces formalism, theorem and algorithm to solve a Boolean equations system (Hietter, 2009). Authors proposed a formal approach to compute PLC control program from specifications.

Riera et al. (2014) propose an approach (safety filter) in order to guarantee the safety whatever the PLC program already in PLC. This safety filter is based on a set of Boolean equations which define safety constraints. Online, for a given set of input variables (I) and a first affectations of output variables (G), the safety filter must find the nearest affectation to G which does not violate any safety constraint.

Considering the online uses of the safety filter to guarantee the safety, it is necessary to prove its ability to find a dependable solution whatever the uncontrollable variables (inputs, internal variables and first affectation). This problem is called *consistency problem* and includes two points: (i) the set of constraints admits at least one solution (ii) the online algorithm is always able to find a solution. In previous works on safety filter, necessary consistent conditions have been proposed but none sufficient condition. In this paper we proposed to use a graph representation of the safety constraints in order to check and ensure their consistency.

The paper is organized as follows, firstly the safety filter formalism is described and the online algorithm is presented. Then a methodology to construct an observer of the safety filter is proposed and illustrated. Next the consistency properties based on the observer are proposed. At last a discussion around the application to manufacturing system and a conclusion are done.

2. THE SAFETY FILTER

2.1 Principle

The classical cycle of a PLC is: update the inputs (sensors), compute the next values of actuators (PLC program) and send these values to the plant. The main idea of the safety filter is to interleave between the PLC program and the outputs update, a block which checks and modifies the outputs vectors (actuators) if needed regarding the safety (see fig. 1). In this work we consider the safety of products and machines, so the safety filter has to avoid possibilities of collision, destruction and so on. Moreover we consider that the functional part (production objectives) is fully contained in the PLC program block.

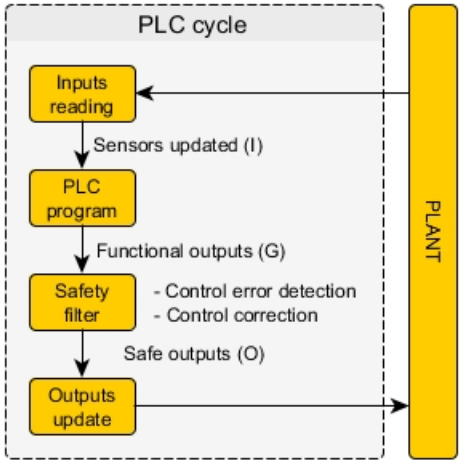

Fig. 1. PLC cycle with safety filter

The safety filter must guarantee the safety whatever the controller, so without knowledge on functional specifications. Moreover considering the online uses, the safety filter has to be deterministic. So given an affectation of the variables (sensors, functional outputs, internal variables) the decision must be always the same.

2.2 Formalism

The safety filter is defined by a set of Boolean equations called *safety constraints*. Two kinds of safety constraints exist, (i) Simple Safety Constraints and (ii) Combined Safety Constraints. These safety constraints are defined as logical functions and designed to be true when the safety is violated (Riera et al., 2014).

A *Simple Safety Constraint* (CSs) is defined as a product of uncontrollable variables and only 1 controllable variable (o_k). So there are only 2 forms of CSs. But several constraints can exist for a specific o_k. The number of CSs for o_k is depicted $Ncss_k$. Let $i \in [1, Ncss_k]$, $k \in [1, N_o]$ such that:

$$CSs_i^{o_k} = \begin{cases} h_{0i}^{o_k} \ . \ o_k \\ OR \\ h_{1i}^{o_k} \ . \ \overline{o_k} \end{cases} \quad (1)$$

With $h_{0i}^{o_k}$ and $h_{1i}^{o_k}$ two monomial (product) functions of uncontrollable variables.

A *Combined Safety Constraint* (CSc) is defined as logical product function. The number of CSc is depicted $Ncsc$. Unlike CSs, each CSc depends on several controllable variables. Let a subset of uncontrollable variables $\Gamma \subseteq Y$ and a subset of controllable variables $\Theta \subseteq O$ with $\gamma_j \in \Gamma$ and $\theta_k \in \Theta$ such that:

$$CSc_i = \prod_j \gamma_j \ . \ \prod_k \theta_k \quad (2)$$

To guarantee the safety, we proposed the following condition: the safety constraints (CSs and CSc) must be *false* ($= 0$) before updating outputs (eq. 3). So, the aim of the safety filter is to check the functional outputs values by verifying the safety condition. Else the algorithm will correct constraints by setting appropriate values to involved outputs.

Safety condition A PLC program can be considered as safe if, for the outputs vector $(o_1, \ldots, o_k, \ldots, o_{N_o})$, eq. 3 is verified before outputs update.

$$\sum_{k=1}^{N_o} \sum_{i=1}^{Ncss_k} CSs_i^{o_k} + \sum_{j=1}^{N_{CSc}} CSc_j = 0 \quad (3)$$

2.3 Resolution procedure

The resolution of the safety problem (at least 1 safety constraint equals to *true*) is subject to a procedure (see fig 2).

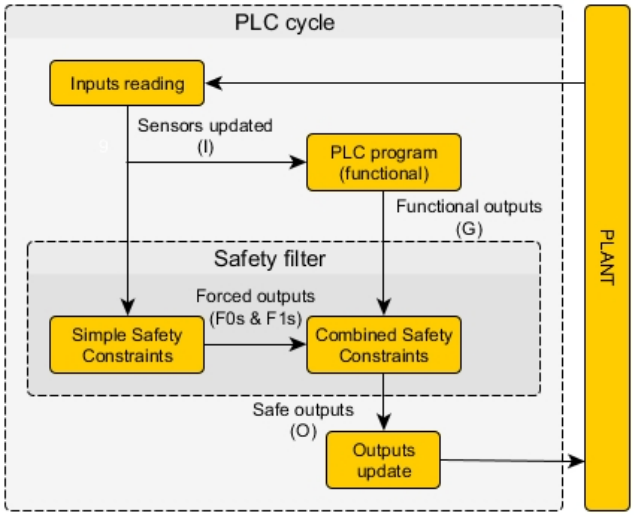

Fig. 2. Safety filter details

Firstly the Simple Safety Constraints are used to compute functions $F0s_k$ and $F1s_k$ for each output o_k (eq. 5). These functions are used as *set* and *reset* functions of each output, $F0s_k = 1$ implies o_k must be forced to 0 and $F1s_k = 1$ implies o_k must be forced to 1.

From equations 1 and 3, the logical OR of all simple safety constraints (CSs) can be written as:

$$\begin{aligned} \textstyle\sum_{i=1}^{Ncss_k} CSs_i^{o_k} &= \textstyle\sum_{i=1}^{Ncss_k} (h_{0i}^{o_k} \ . \ o_k \ + h_{1i}^{o_k} \ . \ \overline{o_k}) \\ &= (o_k . \textstyle\sum_{i=1}^{Ncss_k} (h_{0i}^{o_k}) + \overline{o_k} . \textstyle\sum_{i=1}^{Ncss_k} (h_{1i}^{o_k})) \\ \textstyle\sum_{i=1}^{Ncss_k} CSs_i^{o_k} &= (o_k \ . \ F0s_k \ + \overline{o_k} \ . \ F1s_k) \end{aligned} \quad (4)$$

With $F0s_k$ and $F1s_k$ polynomial functions (sum of products, $\sum\prod$) of only uncontrollable variables (eq. 5).

$$\forall k \in [1, N_o] : \begin{cases} F0s_k = \sum_{i=1}^{Ncss_k} h_{0i}^{o_k} \\ F1s_k = \sum_{i=1}^{Ncss_k} h_{1i}^{o_k} \end{cases} \quad (5)$$

Secondly the Combined Safety Constraints are used to choose values for unforced variables: when $F0s_k$ and $F1s_k$ are $false$. The forced variables may be seen as uncontrollable variables for Combined Safety Constraints solving. The CSc resolution is also based on functional outputs values (G): if the output o_k stays free (unforced by CSs or CSc), final value of o_k is g_k.

However the difficulty of solving CSs and CSc is not the same. The differences are illustrated in the following example.

Constraints example Let 2 uncontrollable variables $Y = \{a, b\}$ and 3 controllable variables $O = \{O_1, O_2, O_3\}$. We will consider in this paper the following example:

Table 1. Example problem

CSs_1	=	$a.O_2$	CSs_2	=	$\overline{a}.b.O_1$
CSc_1	=	$\mathbf{O_1}.\overline{O_2}$	CSc_2	=	$\mathbf{O_2}.\overline{O_3}$

Considering the safety condition (eq. 3), each constraint has to be false at the end of PLC cycle. If $a = true$, CSs_1 implies $O_2 := false$ because O_2 is controllable and not a ($F0sO_2 = a = 1$). So for a Simple Safety Constraint (CSs), the choice to solve a violated constraint is deterministic.

But for Combined Safety Constraint (CSc), a choice has to be done. For example if CSc_1 is violated the filter has to force: (i) O_1 or (ii) O_2 or (iii) both:

$$CSc_1 = true \implies \begin{array}{lll} (i) & O_1 := 0 & ;O_2 := O_2 \\ (ii) & O_1 := O_1 & ;O_2 := 1 \\ (iii) & O_1 := 0 & ;O_2 := 1 \end{array} \quad (6)$$

These choices are made by the expert of the system during the constraints definition and can translate general functional specifications. For a constraint the choice is to select which actuators have priority against others, so which actuators must be forced to solve the constraint.

2.4 Priority

Regarding the resolution procedure and the online algorithm, there are 2 kinds of priority. The first one is the *structural priority* and the second one is the *chosen priority*.

- Structural priority: if some controllable variables are forced by CSs (set/reset functions), for the CSc resolution these variables are uncontrollable. So the CSc resolution must be adaptive to take into account the actions of CSs.
- Chosen priority: if the controllable variables of a CSc are not forced by any CSs, there are several solutions to solve the constraint (see previous example). In this condition the choice made by expert must be applied to solve the constraint.

Chosen priorities are translated using **bold** actuators in table 1. For example with CSc_1 the choice is to force O_1 (i.e. (i) in equation 6).

Online, given an affectation of uncontrollable variables which violates at least one CSc, the way to solve constraints is always the same, so the safety filter is deterministic.

2.5 Safety filter algorithm

At each PLC cycle there are two inputs set: I given by input scan and G given by functional program. Both are considered as constant and uncontrollable during a cycle. The aim is to detect if a constraint is violated, and in this case, find o_k (based on g_k) to guaranty the safety. Moreover thanks to figure 2, the resolution procedure is to solve firstly the CSs and secondly the CSc.

CSs resolution Theorem 1 describes resolution and consistency condition for a single-unknown equation (Cf. theorem 11 in Roussel and Lesage (2014)).

Theorem 1. Let $\mathbb{B}$ a boolean algebra, let a, b, x $\in \mathbb{B}$. The solution of the equation $a.\overline{x} + b.x = 0$ is $x = a + \overline{b}.p$ with $p \in \mathbb{B}$ a parameter iff $a.b = 0$ (consistency condition).

Based on theorem 1, a solution of equation 4 is:

$$\forall k \in [1, N_o], o_k := \overline{F0s_k}.p + F1s_k \quad (7)$$

In our case this solution is used to initialize temporary values of outputs (o_k) based on the g_k values (functional values):

$$\forall k \in [1, N_o], temp_k := \overline{F0s_k}.g_k + F1s_k \quad (8)$$

CSc resolution For the CSc resolution the variables forced by CSs ($F0s_k = 1$ or $F1s_k = 1$) are uncontrollable. Moreover solving a violated combined constraint may violate other combined constraints, the goal is to find progressively a safe solution (no constraints violated).

The inputs of CSc resolution are: the sensors and functional values (I and G), the set and reset functions ($F0s$ and $F1s$) and a first solution ($temp$) which do not violates CSs but may violates CSc.

The CSc resolution procedure consist in: updating the CSc values based on mem_k and I, activating a flag if at least one CSc is violated and then applying the priority in order to solve the violated constraints. But by solving some constraints other may be violated, so this procedure must be applied until all constraints are solved.

At last, the final values mem_k which do not violate any constraint are sent to the plant using the o_k values.

The resulted algorithm is presented in algorithm 1.

In order to guarantee the safety whatever the inputs (sensors) and the functional control part, the safety constraints must be checked formally offline. That implies firstly to verify the consistency of the constraints with chosen priorities, then to check the sufficiency using model checking (Marangé et al., 2010) (Sufficiency is not detailed in this paper).

2.6 Safety filter consistency

In the safety filter approach, the problem (constraints and priorities) is consistent if and only if the following equation 9 is validate.

```
Data: priority, No
Input: G, I
Output: O
/* F0s_k and F1s_k calculation (eq. 5)                    */
for k = 1, ..., No do
    F0s(k) := ...;
    F1s(k) := ...;
end
/* Initialization (eq. 8)                                 */
for k = 1, ..., No do
    temp(k) := NOT F0s(k) AND G(k) OR F1s(k);
end
Flag := true;
/* Combined constraints resolution                        */
while Flag do
    CSC := cscUpdate(mem,I);
    Flag := cscIsViolated(CSC);
    if Flag then
        for k = 1, ..., No do
            temp(k) := applyPriority(F0s, F1s, priority);
        end
    end
end
/* Output update                                          */
for k = 1, ..., No do
    O(k) := temp(k);
end
```

Algorithm 1: Safety filter algorithm

Whatever the set of uncontrollable variables Y (sensors (I), functional outputs (G)) there exists a safe output vector (O):
$\forall (I,G), \exists O = (o_1, \ldots, o_{N_o})$ *such that:*

$$\sum_{k=1}^{N_o} \sum_{i=1}^{Ncss_k} CSs_i^{o_k} + \sum_{j=1}^{N_{CSc}} CSc_j = 0 \tag{9}$$

According to this consistency definition, there are 3 parts:

1- CSs set must be consistent
2- CSc set must be consistent
3- CS set with priorities must be consistent.

The first point is easily checked thanks to consistency condition of theorem 1:

$$\forall k \in [1, N_o], F0s_k.F1s_k = 0; \tag{10}$$

Second and third points are more difficult to check. We proposed in this paper a graphical representation of constraints, interaction between them and resolution procedure. Thanks to this graphical representation, the consistency checking is reduced to a cycle detection in a directed graph.

3. THE PROPOSED APPROACH

Based on the safety filter problem (constraints and priorities) we proposed to analyze the structural links between constraints without priorities in order to reduce the problem. Thanks to this reduced problem we construct an observer of the safety filter algorithm (alg. 1) in order to check the consistency (fig. 3).

3.1 Reduced problem: Structural Graph

We proposed to represent the structural interaction between constraints with an undirected graph called Structural Graph SG = (CS, E) such that:

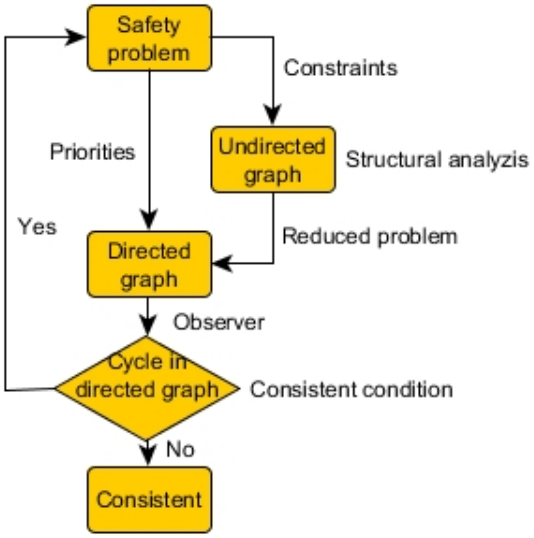

Fig. 3. Method to check the consistency

- $CS = CSs \cup CSc$ is a set of vertex: each vertex is a safety constraint;
- E is a set of edges. Each edge represents a link between 2 constraints.

An edge between 2 constraints means that if one of them is violated, the resolution may violates the other. For example, consider previous constraints ($CSs_1 = a.O_2; CSc_1 = O_1.\overline{O_2}$), $CSs_1 = 1$ implies O_2 will be forced to 0 so if $O_1 = 1$ at the same time, CSc_1 will be violated.

Definition 2. Given 2 constraints CS_i and CS_j, an edge exists between them iff:

1- Logical product of uncontrollable part of CS_i and CS_j is not false;
2- If $O_k \in CS_i$ then $\overline{O_k} \in CS_j$ OR If $\overline{O_k} \in CS_i$ then $O_k \in CS_j$

An edge can exist between 2 constraints only if they can be violated at the same time whatever the controllable variables (first point in def. 2). Moreover if there are no shared controllable variables between constraints, edge can not exist (second point in def. 2).

Considering the previous example (table 1) without priority, the corresponding structural graph is presented in figure 4.

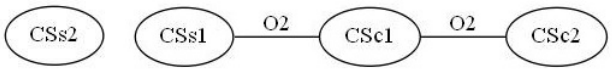

Fig. 4. Structural graph

There is no edge between CSs_2 and CSs_1 because $(a).(\overline{a}.b) = 0$ (first point in def 2). Moreover there is no edge between CSs_2 and CSc_1 due to second point in def 2. At last there are no shared variables between CSs_2 and CSc_2 so no edge between them.

CSs_2 is isolated (no edges with other constraints) so regarding the consistency problem, this constraint should be ignored. Indeed the resolution of CSs_2 will never violates other constraints, and other constraints will never violate CSs_2.

Considering the structural graph in figure 4 we can extract the reduced problem by removing the isolated constraint CSs_2 (table 2).

Table 2. Reduced problem

CSs_1	=	$a.O_2$			
CSc_1	=	$\mathbf{O_1}.\overline{O_2}$	CSc_2	=	$\mathbf{O_2}.\overline{O_3}$

Based on this reduced problem the observer can be constructed.

3.2 Observer construction

The goal is to construct a graph which represents all possible developments of the variables during a PLC cycle. Each vertex of the graph is defined by an affectation of variables and labeled by constraints which are violated or set and reset functions (eq. 5). Each arc represents the choice made to solve the constraints. The Safety Filter Observer (SFO) is defined as follows:

Definition 3. The Safety Filter Observer (SFO) is a directed graph SFO = (V, A) such that:

- V is a set of vertices: each vertex is a full affectation of the safety filter variables;
- A is a set of arcs which represents the choice made by the safety filter algorithm.

Given a reduced problem (section 3.1), the Safety Filter Observer is built as follows:

1- Compute each safe vertex;
2- Compute each unsafe vertex;
3- Link vertices according to the priorities;
4- Delete safe vertices which are not linked.

Computation of safe vertices is possible with a SATisfiability problem solver (Du et al., 1997). Given a set of clauses (Boolean equations) a SAT solver is able to provide all possible affectations of variables which satisfy all the clauses.

Each affectation which is not listed by SAT solver is obviously unsafe. For each of them we have to analyze the affectation in order to know which constraints are violated. We labeled each unsafe vertex with the corresponding violated constraints.

For each unsafe vertex and considering the violated constraints we apply the priorities. As shown previously the resolution is deterministic for an affectation, so there exists only one outgoing arcs from each vertex. If for an affectation, two constraints are in contradiction (one force to $false$ and an other to $true$) the arc is changed to a loop on the same unsafe vertex.

Finally the safe vertices without any entering or outgoing arcs are removed in order to reduce the size of final graph.

Considering the previous example and the reduced problem presented in table 2 (with CSc priorities), the resulting Safety Filter Observer is presented in figure 5.

3.3 Consistency condition

Each vertex of the observer represents an affectation of variables at the beginning of the safety filter algorithm (algo. 1). So each vertex may be initial and given an initial vertex, the final value of actuators will be always the same because the resolution procedure (through priorities) is deterministic.

Based on the definition and construction of the Safety Filter Observer (SFO), the consistency condition (section 2.6) can be translated to a reachable analysis (theorem 4).

Theorem 4. Problem consistent ⇔ For all unsafe vertices, a safe vertex is reachable.

Checking each path of the graph is time-consuming. But regarding the construction of the SFO, there is only 1 outgoing arc from each vertex. So the consistency condition can be resumed to the non-existence of a cycle in the SFO (theorem 5). Indeed if a cycle exists, it means that there exists at least 1 unsafe vertex which can not be changed to a safe vertex (example provided below).

Theorem 5. Problem consistent ⇔ A cycle does not exist in the SFO.

The consistency condition can be illustrated on the previous example by changing priority for CSc_1: O_1 prioritary instead of O_2. The constraints have not changed so the structural analysis must not be computed. The SFO corresponding to this new priority is presented in figure 6.

A cycle exists between vectors 0110 and 0100. So if the inputs of algorithm (sensors I and functional outputs G) equal one of these vectors, the algorithm will not be able

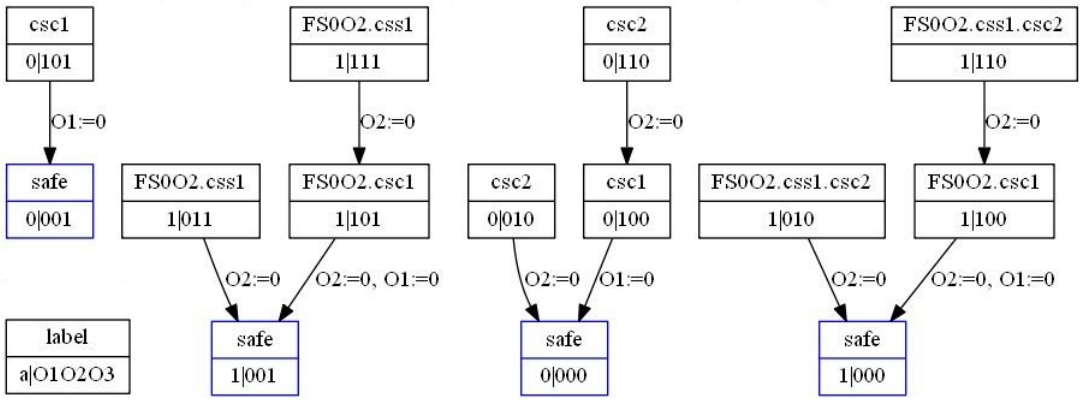

Fig. 5. Safety Filter Observer for example 2

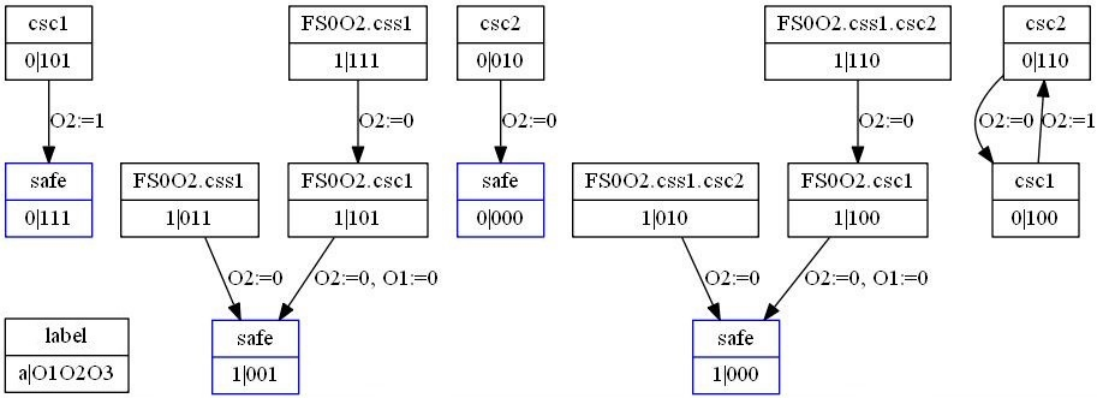

Fig. 6. SFO with inconsistent priorities for example 2

to find a safe affectation. In this condition the problem is inconsistent and the constraints and/or the priorities have to be redefined.

When the set of constraints and priorities are consistent, the safety filter can be synthesized and implemented in the PLC in order to make safe any existing program.

4. WHAT ABOUT MANUFACTURING SYSTEM ?

Our approach is based on an exploration of all the possible states of variables, so there exists a risk of state explosion. But in practice on manufacturing systems, the constraints are weakly connected so the problem is highly reduced.

Riera et al. (2014) detail the constraints and priorities for a sorting boxes system. The system is instrumented using 11 sensors and 7 actuators. The safety analysis has resulted in 17 CSs and 5 CSc for a total of 21 variables (sensors, actuators and internal variables).

The structural analysis presented before reduced the problem to only 7 constraints (4 CSs and 3 CSc) and 10 variables. And the consistency checking on the reduced problem is computed in few seconds.

Moreover in many times manufacturing systems are structurally modular. This modularity can be seen in the structural graph if there are subgraphs. Indeed, unconnected subgraphs allowed to analyze separately each subgraph to conclude about the global consistency.

Based on our experiences and tests on different kinds of manufacturing systems, the state explosion problem has never appear.

5. CONCLUSION

In this paper we have proposed for the first time a graphical representation of the safety constraints. Two levels of representation have been developed, the first one is a structural approach by undirected graph in order to underline the links between constraints and to reduce the problem. The second one is an observer of the safety filter using directed graph in order to follow the developments of the algorithm. Thanks to this observer, we have proposed a graphical necessary and sufficient condition which ensures the consistency of the constraints and priorities off-line. The graphical representation of the safety constraints offers new ways of research for the safety filter approach: model-based diagnosis and compatibility checking with the plant's model.

REFERENCES

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

Cassandras, C.G. and Lafortune, S. (2009). *Introduction to Discrete Event Systems*. Springer Science & Business Media.

Charbonnier, F., Alla, H., and David, R. (1999). Discrete-event dynamic systems. *Control Systems Technology, IEEE Transactions on*, 7(2), 175–187.

Du, D., Gu, J., and Pardalos, P.M. (1997). *Satisfiability Problem: Theory and Applications : DIMACS Workshop, March 11-13, 1996*. American Mathematical Soc.

Hietter, Y. (2009). *Synthèse algébrique de lois de commande pour les systèmes à évènements discrets logiques*. Ph.D. thesis, École normale supérieure de Cachan-ENS Cachan.

II, C.A.E. (2015). *Hazard Analysis Techniques for System Safety*. John Wiley & Sons. Google-Books-ID: piLyCQAAQBAJ.

Marangé, P., Benlorhfar, R., Gellot, F., and Riera, B. (2010). Prevention of human control errors by robust filter for manufacturing system. In *Analysis, Design, and Evaluation of Human-Machine Systems*, volume 11, 175–180.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98. doi:10.1109/5.21072.

Riera, B., Coupat, R., Annebicque, D., Philippot, A., and Gellot, F. (2014). Control synthesis based on safety boolean guards for manufacturing systems: application to a sorting system. In *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*. Nancy, France.

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.

Vilela, J.N. and Pena, P.N. (2016). Supervisor abstraction to deal with planning problems in manufacturing systems. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 117–122.