Algebraic synthesis of logical controllers despite inconsistencies in specifications

Jean-Marc Roussel, Jean-Jacques Lesage

Algebraic synthesis of logical controllers despite inconsistencies in specifications

Jean-Marc Roussel * **Jean-Jacques Lesage** *

* *LURPA / ENS Cachan, 61 Avenue du Pt Wilson, F-94235 Cachan Cedex France, (e-mail: firstname.lastname@lurpa.ens-cachan.fr).*

Abstract: This paper deals with the problem of consistency of the requirements which are the starting point of controller synthesis methods for Discrete Event Systems (DES). In previous papers, an algebraic synthesis method for logical systems has been proposed. This method includes a theorem allowing the proof of consistency of the set of specifications. In this paper, we show that if inconsistencies are detected, the conditions of these inconsistencies can be given to the designer. It is therefore possible for the designer to propose priority rules between the involved requirements allowing the correction of incoherent specifications. Both the mathematical and the methodological aspects of this work are presented. In an illustrative purpose, the case study of a hydraulic press is developed.

Keywords: Controller synthesis, Algebraic approaches, Boolean algebra, Dependable system

1. INTRODUCTION

Programmable Logic Controllers (PLCs) are frequently used for the control of safety-critical systems. Since failure of such systems may have disastrous effects, the use of formal methods for the design of control algorithms is often mandatory (IEC 61508 (1998)). Automatic synthesis methods are among the best solutions to handle this problem and are one of the most active subjects of research in the field of Discrete Event Systems (DES) since the end of 80's (cf. numerous sessions organized in this field in the past WODES editions [1]). Most part of recent works in this area are still based onto the Supervisory Control Theory (SCT) (Ramadge and Wonham (1989)) and are aiming to the synthesis of a *Supervisor*, and not directly to the *Controller* of an automated system. Furthermore, the use of state models (Finite Automata, Petri Nets...) and their composition for the construction of the models of the plant and of the specifications generates a complexity which remains problematic for the synthesis of a supervisor for complex systems (Gohari and Wonham (2000)). It is therefore interesting to explore other ways for performing synthesis, such as algebraic approaches. In previous works, we proposed a method specifically developed to get the control laws that can be directly implemented into the controller (Hietter (2009)). We have chosen to synthesize these control laws under the form of recurrent Boolean equations because of the wide possibilities they offer for the formalization of safety requirements and for implementation.

Nevertheless, whatever is the synthesis method used, one of the weak links of the automatic generation of the control laws is the step of formal transcription (within state models or algebraic expressions) by the designer of the informal requirements and safety properties the controller has to satisfy. In the case of SCT, some authors have proposed more or less generic approaches for the construction of the models of the plant (Hanisch et al. (1998)) or of the specifications (Roussel and Giua (2005)). But in any case, the hypothesis that requirements can be inconsistent has never been taken into account, except in (Zowghi and Offen (1997)) where authors propose a theorization of requirements evolution. Unfortunately in the framework of industrial collaborations we have been able to verify that it is often the case. In this paper we show how, in consideration of specific hypotheses, it is possible to install a correction loop for helping the designer to formalize these requirements and so, improving the synthesis method robustness to the lack of precision of the specifications.

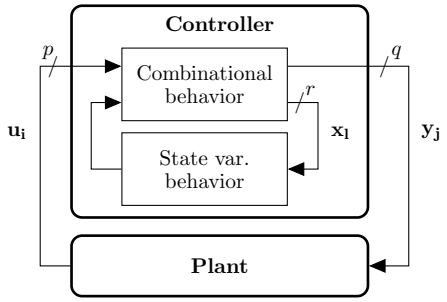
This paper is organized as follows. Some basics of algebraic synthesis are given in Section 2, and Section 3 recalls the main steps of our method. Section 4 presents the mathematical framework of our approach and new results that allow us to accept inconsistencies in specifications. The strategy we developed for making the synthesis more robust to the lack of consistency of the specifications is described in section 5 thanks to a case study.

2. PROBLEM STATEMENT

Fig. 1 proposes a generic representation of a DES whose the controller has p Boolean inputs (u_i), q Boolean outputs (y_j) and r Boolean state variables (x_l). Plant and Controller are connected through a closed loop exchanging inputs and outputs signals. The state variables, needed for expressing sequential behaviors of the controller, are represented by internal variables.

The algebraic modeling of the control laws of the controller necessitates the definition of $(q + r)$ switching functions of $(p + r)$ variables. Even if this representation is very compact (the r Boolean state variables allow the representation of 2^r different states), the construction by hands of

[1] available at http://www.diee.unica.it/giua/WODES/



$$\begin{cases} y_j[k] = F_j(u_1[k], \cdots, u_p[k], x_1[k-1], \cdots, x_r[k-1]) \\ x_l[k] = F_{q+l}(u_1[k], \cdots, u_p[k], x_1[k-1], \cdots, x_r[k-1]) \end{cases}$$

Fig. 1. A sequential DES

these switching functions is a very tedious and error-prone task (Huffman (1954)): the controller of Fig. 1, admits 2^p inputs combinations, can send 2^q outputs combinations and can express $(2^{2^{(p+r)}})^{(q+r)}$ sequential behaviors. That is the reason why algebraic modeling approaches have been replaced by methods based on state models since the middle of 50's (Mealy (1955), Moore (1956)). Nevertheless, thanks to recent mathematical results obtained onto Boolean algebras (Rudeanu (2001), Brown (2003)), the automatic algebraic synthesis of switching functions is now possible.

In Pnueli and Rosner (1989) an interesting approach for the systematic construction of a reactive program from its formal specification is proposed. In this work, the program synthesis is considered as a theorem proving activity. A program with input x and output y, specified by the formula $\varphi(x, y)$, is constructed as a by-product of proving the theorem $(\forall x)(\exists y)\varphi(x, y)$. The specification $\varphi(x, y)$ characterizes the expected relation between the input x and the output y computed by the program. This approach is based on the observation that the formula $(\forall x)(\exists y)\varphi(x, y)$ is equivalent to the second order formula $(\exists f)(\forall x)\varphi(x, f(x))$, stating the existence of a function f, such that $\varphi(x, f(x))$ holds for every x.

This approach provides a conceptual framework for the rigorous derivation of a program from its formal specification. It has also been used to synthesize specifications under the form of finite automata from their Linear Temporal Logic (LTL) description (Filiot et al. (2011)).

The core of our approach is based on this strategy: we aim at deducing the $(q + r)$ switching functions of $(p + r)$ variables which define the behavior of the controller from a formula $\varphi(u_i[k], x_l[k-1], y_j[k], x_l[k])$ that holds for every k, every $u_i[k]$ and every $x_l[k-1]$.

To cope with combinatorial explosion, switching functions will be handled through a symbolic representation (and not their truth-tables which contains $2^{(p+r)}$ Boolean values). Each input u_i (res. output y_j) of the controller will be represented by a switching function U_i (res. Y_j). To take into account the recursive aspect of state variables, each state variable x_l will be represented by two switching functions: X_l (for time $[k]$) and ${}_pX_l$ (for time $[k-1]$).

According to this representation, the synthesis of control laws of a logical system from its specification can now be transformed into the search of the solution of the mathematical problem:

$$(\forall U_i)(\forall\, {}_pX_l)(\exists Y_j)(\exists X_l)\varphi(U_i, {}_pX_l, Y_j, X_l)$$

where $(U_i, {}_pX_l, Y_j, X_l)$ are $(p+q+2r)$ switching functions of $(p + r)$ variables.

3. OVERVIEW OF OUR METHOD

The input data of the proposed method (Fig 2) are unformal functional and safety requirements given by the designer. In practice, these requirements are most often given in a textual form and/or by using technical Taylor-made languages (Gantt Diagrams, Function Blocks Diagrams, Grafcet...).

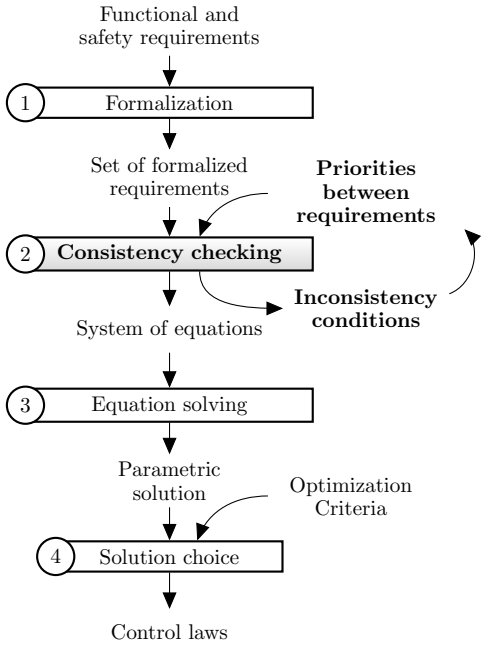


Fig. 2. The algebraic synthesis method step by step

All the steps of our synthesis method are implemented into a prototype software tool developed in Python [2] . The first step is the formalization of requirements within an algebraic description. Requirements expressed with a state model can directly be translated into recurrent Boolean equations, thanks to the algorithm proposed by Machado et al. (2006). Generic formalization of some requirements (change of operation modes, emergency management...) are also proposed for helping the designer to formalize "standard" requirements given in textual form (examples are given in Section 5.2). In case where the know-how of the designer enables him to build a priori the global form of the solution (or of a part of the whole solution) it is also possible to give fragments of solution as requirements (Hietter et al. (2008c)).

The second step consists in checking the consistency of the set of requirements by symbolic calculation. In a previous paper (Hietter et al. (2008a)), a theorem allowing the proof of consistency of the set of requirements has been proposed. In this paper, we show that if inconsistencies are detected, the conditions of these inconsistencies can be given

[2] Case studies are available online: http://www.lurpa.ens-cachan.fr/isa/asc/case_studies.html

to the designer. It is therefore possible for the designer to propose priority between the involved requirements allowing the correction of inconsistent specifications. We will develop more especially the mathematical results on which this step of the method is based in Section 4.3; it will be applied to a case study in Section 5.3.

The core of the method is the third step, which consists in the synthesis of the control laws. This step is performed by solving the system of equations which represents the set of consistent requirements. The mathematical results we have obtained (Theorem 2 recalled in Section 4.2), allow to find a symbolic expression of the set of solutions under the form of a parametric expression (Hietter et al. (2008b)).

In the fourth step of the method, a particular solution has to be chosen among the set of solutions. For that, a specific value of each parameter of the general solution has to be fixed. In some cases, optimal solutions, according to given criteria, can be automatically found (Leroux (2011)). For ergonomic reasons, the synthesized control laws can finally be displayed under the form of a finite automaton (Guignard (2011)).

After the mathematical background of the method has been recalled, we are going to show how, in consideration of specific hypotheses, the second step of the method can be improved by a correction loop helping the designer to formalize the requirements and so improving the robustness of our synthesis method to the lack of precision of the specifications.

4. MATHEMATICAL BASICS

Section 4.1 recalls classical notations and results onto the Boolean algebra of n-variable switching functions. Section 4.2 recalls the principal results on which our synthesis method is based. Section 4.3 presents the new results we use for helping the designer to formalize requirements.

4.1 The Boolean algebra of n-variable switching functions

To avoid confusion between Boolean variables and Boolean functions of Boolean variables, each Boolean variable b_i is denoted as ${}_b b_i$. The set of the two Boolean values ${}_b 0$ and ${}_b 1$ is denoted as: $B = \{{}_b 0, {}_b 1\}$. The classical two-element Boolean Algebra is $(\{{}_b 0, {}_b 1\}, \vee, \wedge, \neg, {}_b 0, {}_b 1)$.

An n-variable *switching function* is a mapping [3]

$$\begin{array}{rc} f: & B^n \to B \quad \text{where } B = \{{}_b 0, {}_b 1\} \\ & ({}_b b_1, \ldots, {}_b b_n) \mapsto f({}_b b_1, \ldots, {}_b b_n) \end{array}$$

Let $F_n(B)$ be the set of the 2^{2^n} n-variable switching functions. $F_n(B)$ contains $(n+2)$ specific n-variable switching functions: the 2 *constant functions* (0, 1), and the n *projection-functions* ($\mathrm{f}^i_{\mathrm{Pj}}$). These functions are defined by:

$$\begin{array}{rcrc} 0: & B^n \to B & 1: & B^n \to B \\ & ({}_b b_1, \ldots, {}_b b_n) \mapsto {}_b 0 & & ({}_b b_1, \ldots, {}_b b_n) \mapsto {}_b 1 \end{array}$$

$$\begin{array}{rc} \mathrm{f}^i_{\mathrm{Pj}}: & B^n \to B \\ & ({}_b b_1, \ldots, {}_b b_n) \mapsto {}_b b_i \end{array}$$

[3] From Section 3.11 of Brown (2003).

$F_n(B)$ can be equipped with three closed operations in order to obtain the Boolean Algebra $(F_n(B), +, \cdot, \bar{}, 0, 1)$:

$$\begin{array}{rcrc} \text{Op. } +: & F_n(B)^2 \to F_n(B) & \text{Op. } \cdot: & F_n(B)^2 \to F_n(B) \\ & (f,g) \mapsto f+g & & (f,g) \mapsto f \cdot g \end{array}$$

$$\begin{array}{rc} \text{Op. } \bar{}: & F_n(B) \to F_n(B) \\ & f \mapsto \overline{f} \end{array}$$

where $\forall ({}_b b_1, \ldots, {}_b b_n) \in B^n$,

$$\begin{aligned} (f+g)({}_b b_1, \ldots, {}_b b_n) &= f({}_b b_1, \ldots, {}_b b_n) \vee g({}_b b_1, \ldots, {}_b b_n) \\ (f \cdot g)({}_b b_1, \ldots, {}_b b_n) &= f({}_b b_1, \ldots, {}_b b_n) \wedge g({}_b b_1, \ldots, {}_b b_n) \\ \overline{f}({}_b b_1, \ldots, {}_b b_n) &= \neg f({}_b b_1, \ldots, {}_b b_n) \end{aligned}$$

A *Boolean formula* onto $F_n(B)$ is a composition $\mathcal{F}(\alpha_1, \ldots, \alpha_n)$ of $(\alpha_1, \ldots, \alpha_n)$ by operations $+$, $\cdot$ and $\bar{}$. It represents one and only one element of $F_n(B)$ and can be expressed as:

$$\mathcal{F}(\alpha_1, \ldots, \alpha_n) = \mathcal{F}_0(\alpha_2, \ldots, \alpha_n) \cdot \overline{\alpha_1} + \mathcal{F}_1(\alpha_2, \ldots, \alpha_n) \cdot \alpha_1$$

where $\mathcal{F}_0(\alpha_2, \ldots, \alpha_n)$ and $\mathcal{F}_1(\alpha_2, \ldots, \alpha_n)$ are Boolean formulae of $\alpha_2, \ldots, \alpha_n$. These two formulae can be directly obtained from $\mathcal{F}(\alpha_1, \ldots, \alpha_n)$ as follows:

$$\begin{cases} \mathcal{F}_0(\alpha_2, \ldots, \alpha_n) = \mathcal{F}(\alpha_1, \ldots, \alpha_n)|_{\alpha_1 \leftarrow 0} = \mathcal{F}(0, \alpha_2, \ldots, \alpha_n) \\ \mathcal{F}_1(\alpha_2, \ldots, \alpha_n) = \mathcal{F}(\alpha_1, \ldots, \alpha_n)|_{\alpha_1 \leftarrow 1} = \mathcal{F}(1, \alpha_2, \ldots, \alpha_n) \end{cases}$$

It is also possible to define a partial order relation, called *Inclusion-Relation*, between elements of $F_n(B)$.

Definition 1. (Inclusion-Relation). [4]
$\forall (x, y) \in F_n(B)^2$, define $x \leq y$ if and only if $x \cdot y = x$.

Relation Inclusion is a partial order: it is reflexive ($x \leq x$), antisymmetric (if $x \leq y$ and $y \leq x$, then $x = y$) and transitive (if $x \leq y$ and $y \leq z$, then $x \leq z$).
Since $x \cdot y = x \Leftrightarrow x \cdot \overline{y} = 0$, we also have $x \leq y \Leftrightarrow x \cdot \overline{y} = 0$.

4.2 Solutions of Boolean equations over $F_n(B)$

Consider the Boolean algebra of n-variable switching functions $(F_n(B), +, \cdot, \bar{}, 0, 1)$.

- Let $(\mathrm{f}^1_{\mathrm{Pj}}, \cdots, \mathrm{f}^n_{\mathrm{Pj}})$ be the *projection-functions* of $F_n(B)$.
- Let $(x_1, \cdots, x_k)$ be k elements of $F_n(B)$ considered as *unknowns*.

For notational convenience, we note 'X_k' the vector $(x_1, \cdots, x_k)$ of the k unknowns and 'Pj' the vector $(\mathrm{f}^1_{\mathrm{Pj}}, \cdots, \mathrm{f}^n_{\mathrm{Pj}})$ of the n projection-functions of $F_n(B)$.

Any set of simultaneously-asserted relations of n-variable switching functions can be reduced to a single equivalent relation as: $\mathcal{F}(\mathrm{X}_k, \mathrm{Pj}) = 0$. For that, it is sufficient:

- to rewrite each equality according to:
  $\mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) = \mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj}) \quad \Leftrightarrow$
  $\mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) \cdot \overline{\mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj})} + \overline{\mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj})} \cdot \mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj}) = 0$
- to rewrite each inclusion according to:
  $\mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) \leq \mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj}) \Leftrightarrow \mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) \cdot \overline{\mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj})} = 0$
- to group rewritten equalities as follows:
  $\begin{cases} \mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) = 0 \\ \mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj}) = 0 \end{cases} \Leftrightarrow \mathcal{F}_1(\mathrm{X}_k, \mathrm{Pj}) + \mathcal{F}_2(\mathrm{X}_k, \mathrm{Pj}) = 0$

[4] Definition 15.6 of Grimaldi (2004).

In order to express $\mathrm{Eq}(\mathrm{X}_k, \mathrm{Pj}) = 0$ in a canonic form, the following notation has to be introduced: for $x \in F_n(B)$ and $a \in \{0,1\}$, x^a is defined by

$$x^0 = \overline{x} \qquad x^1 = x$$

This notation can be used for vectors as follows:
for $\mathrm{X}_k = (x_1, \cdots, x_k) \in F_n(B)^k$ and $\mathrm{A}_k = (a_1, \cdots, a_k) \in \{0,1\}^k$, $\mathrm{X}_k^{A_k}$ is defined by

$$\mathrm{X}_k^{A_k} = \prod_{i=1}^{i=k} x_i^{a_i} = x_1^{a_1} \cdot \ldots \cdot x_k^{a_k}$$

Theorem 1. Canonic form of a Boolean equation
Any Boolean equation $\mathrm{Eq}(\mathrm{X}_k, \mathrm{Pj}) = 0$ can be expressed in the canonic form

$$\sum_{\mathrm{A}_k \in \{0,1\}^k} \mathrm{Eq}(\mathrm{A}_k, \mathrm{Pj}) \cdot \mathrm{X}_k^{A_k} = 0$$

where $\mathrm{Eq}(\mathrm{A}_k$,Pj) (with $\mathrm{A}_k \in \{0,1\}^k$) are the 2^k discriminants [5] of $\mathrm{Eq}(\mathrm{X}_k, \mathrm{Pj})$ according to X_k.

This canonic form is obtained by expressing $\mathrm{Eq}(\mathrm{X}_k, \mathrm{Pj})$ according to the k unknowns. For example, we have:

$$\begin{aligned}\mathrm{Eq}(x_1, x_2, \mathrm{Pj}) = {} & \mathrm{Eq}(0,0,\mathrm{Pj}) \cdot \overline{x_1} \cdot \overline{x_2} + \mathrm{Eq}(0,1,\mathrm{Pj}) \cdot \overline{x_1} \cdot x_2 \\ & + \mathrm{Eq}(1,0,\mathrm{Pj}) \cdot x_1 \cdot \overline{x_2} + \mathrm{Eq}(1,1,\mathrm{Pj}) \cdot x_1 \cdot x_2\end{aligned}$$

Theorem 2. Solution of a k-unknown equation
The Boolean equation over $F_n(B)$

$$\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0 \tag{1}$$

is consistent (*i.e.* has at least one solution) if and only if the following condition is satisfied:

$$\prod_{\mathrm{A}_k \in \{0,1\}^k} \mathrm{Eq}_0(\mathrm{A}_k, \mathrm{Pj}) = 0 \tag{2}$$

When (2) is satisfied, Equation (1) admits one or more k-tuple solutions $(S(x_1), \cdots, S(x_k))$ where each $S(x_i)$ is defined by

$$\begin{aligned}S(x_i) = {} & \prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(0, \mathrm{A}_{k-i}, \mathrm{Pj}) \\ & + p_i \cdot \overline{\prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(1, \mathrm{A}_{k-i}, \mathrm{Pj})}\end{aligned} \tag{3}$$

where

- $\mathrm{Eq}_i(x_{i+1}, \cdots, x_k, \mathrm{Pj}) = \mathrm{Eq}_{i-1}(x_i, \cdots, x_k, \mathrm{Pj})|_{x_i \leftarrow S(x_i)}$
- p_i is an *arbitrary parameter*, *i.e*, a freely-chosen element of $F_n(B)$.

The general form of this theorem can be found in Rudeanu (2001) or Brown (2003). However, in these works, the solution is not expressed in a parametric form, but only with intervals. The proof of this parametric formulation, which is mandatory in our approach, can be found in Hietter (2009).

4.3 Two new theorems for coping with inconsistencies of specifications

In practice, it is very difficult for a designer to specify the whole requirements of a complex system without inconsistencies. It is the reason why requirements given by the designer are often declared as inconsistent according to Theorem 2. Since the inconsistency condition is a Boolean formula, it is possible to use it for the detection of the origin of inconsistencies. Two cases have to be considered:

- Several requirements cannot be simultaneously respected. In this case, a hierarchy between requirements can be proposed in order to find a solution. The requirements which have the lower priority have to be corrected for becoming consistent with the requirements which have the higher priority. This strategy is based on Theorem 4.
- The detected inconsistency refers to combinations of the projection-functions for which the designer knows that there is no solution. To avoid blocking the synthesis process, it is necessary to introduce new assumptions and to use Theorem 3.

Theorem 3. Solution of a Boolean equation according to an assumption among the projection-functions.
The following problem

$$\left[\begin{array}{l}\text{Equation to solve:} \\ \quad \mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0 \\ \text{Assumptions:} \\ \quad \mathcal{A}(\mathrm{Pj}) = 0\end{array}\right. \tag{4}$$

admits the same solutions as the following equation:

$$\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) \leq \mathcal{A}(\mathrm{Pj}) \tag{5}$$

Proof. According to $\mathcal{A}(\mathrm{Pj}) = 0$, $\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0$ can be rewritten as:

$$\begin{cases}\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0 \\ \mathcal{A}(\mathrm{Pj}) = 0\end{cases} \Leftrightarrow \mathcal{A}(\mathrm{Pj}) + \mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0$$

$$\Leftrightarrow \mathcal{A}(\mathrm{Pj}) + \overline{\mathcal{A}(\mathrm{Pj})} \cdot \mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0$$

$$\Leftrightarrow \begin{cases}\overline{\mathcal{A}(\mathrm{Pj})} \cdot \mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0 \\ \mathcal{A}(\mathrm{Pj}) = 0\end{cases} \Leftrightarrow \begin{cases}\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) \leq \mathcal{A}(\mathrm{Pj}) \\ \mathcal{A}(\mathrm{Pj}) = 0\end{cases}$$

Equation $\overline{\mathcal{A}(\mathrm{Pj})} \cdot \mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0$ is consistent if and only if the following condition is true (Theorem 2):

$$\overline{\mathcal{A}(\mathrm{Pj})} \cdot \prod_{\mathrm{A}_k \in \{0,1\}^k} \mathrm{Eq}_0(\mathrm{A}_k, \mathrm{Pj}) = 0 \tag{6}$$

By construction, this new condition is the subset of the initial condition ($\prod_{\mathrm{A}_k \in \{0,1\}^k} \mathrm{Eq}_0(\mathrm{A}_k, \mathrm{Pj}) = 0$) for which the proposed assumption is satisfied. All the others terms have been removed.

If (6) is satisfied, (5) admits one or more k-tuple solutions where each component $S(x_i)$ is defined by

$$\begin{aligned}S(x_i) = {} & \overline{\mathcal{A}(\mathrm{Pj})} \cdot \left(\prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(0, \mathrm{A}_{k-i}, \mathrm{Pj}) \right. \\ & \left. + p_i \cdot \overline{\prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(1, \mathrm{A}_{k-i}, \mathrm{Pj})} \right) \\ & + \mathcal{A}(\mathrm{Pj}) \cdot p_i\end{aligned}$$

As $\mathcal{A}(\mathrm{Pj}) = 0$, $S(x_i)$ can also be expressed as:

$$\begin{aligned}S(x_i) = {} & \prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(0, \mathrm{A}_{k-i}, \mathrm{Pj}) \\ & + p_i \cdot \overline{\prod_{\mathrm{A}_{k-i} \in \{0,1\}^{k-i}} \mathrm{Eq}_{i-1}(1, \mathrm{A}_{k-i}, \mathrm{Pj})}\end{aligned}$$

When $\mathcal{A}(\mathrm{Pj}) = 0$ is satisfied, the solutions of (5) are also solution of Equation $\mathrm{Eq}_0(\mathrm{X}_k, \mathrm{Pj}) = 0$.

[5] The term of 'discriminant' comes from Brown (2003).

□

Theorem 4. Solution of a Boolean equation system according to a priority rule between requirements.
The following problem

$$\left[\begin{array}{l}\text{Equations system to solve:}\\ \begin{cases}\text{HR} & \mathcal{F}_{\mathcal{H}}(\mathrm{X}_k,\mathrm{Pj})=0\\ \text{LR} & \mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj})=0\\ \text{OR} & \mathcal{F}_{\mathcal{O}}(\mathrm{X}_k,\mathrm{Pj})=0\end{cases}\\ \text{Priority rule between requirements:}\\ \quad \text{HR} \gg \text{LR}\end{array}\right. \tag{7}$$

where:

- $\mathcal{F}_{\mathcal{H}}(\mathrm{X}_k,\mathrm{Pj}) = 0$ is the formal expression of the requirements which have the higher priority (HR).
- $\mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj}) = 0$ is the formal expression of the requirements which have the lower priority (LR).
- $\mathcal{F}_{\mathcal{O}}(\mathrm{X}_k,\mathrm{Pj}) = 0$ is the formal expression of the others requirements (OR).
- HR $\gg$ LR is the priority rule between inconsistent requirements.

admits the same solutions as the system of equations:

$$\begin{cases}\mathcal{F}_{\mathcal{H}}(\mathrm{X}_k,\mathrm{Pj})=0\\ \mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj})\leq\mathcal{I}(\mathrm{Pj})\\ \mathcal{F}_{\mathcal{O}}(\mathrm{X}_k,\mathrm{Pj})=0\end{cases} \tag{8}$$

where $\mathcal{I}(\mathrm{Pj})$ is the inconsistency condition between requirements 'HR' and 'LR':

$$\mathcal{I}(\mathrm{Pj}) = \prod_{\mathrm{A}_k\in\{0,1\}^k} \left(\mathcal{F}_{\mathcal{H}}(\mathrm{A}_k,\mathrm{Pj}) + \mathcal{F}_{\mathcal{L}}(\mathrm{A}_k,\mathrm{Pj})\right)$$

Proof. Thanks to Theorem 2, the inconsistency condition $\mathcal{I}(\mathrm{Pj})$ between requirements 'HR' and 'LR' can be found by solving equation: $\mathcal{F}_{\mathcal{H}}(\mathrm{X}_k,\mathrm{Pj}) + \mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj}) = 0$. We have:

$$\mathcal{I}(\mathrm{Pj}) = \prod_{\mathrm{A}_k\in\{0,1\}^k} \left(\mathcal{F}_{\mathcal{H}}(\mathrm{A}_k,\mathrm{Pj}) + \mathcal{F}_{\mathcal{L}}(\mathrm{A}_k,\mathrm{Pj})\right)$$

To remove the inconsistency between Requirements 'HR' and 'LR' according to the priority rule 'HR $\gg$ LR', it is necessary to restrict the range of requirement 'LR' to the part for which there is no inconsistency, i.e. $\mathcal{I}(\mathrm{Pj}) = 0$. That is the case, when $\mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj}) = 0$ is replaced by $\mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj}) \leq \mathcal{I}(\mathrm{Pj})$.

Thanks to Theorem 2, (9) admits always one or more k-tuple solutions and it is impossible to find a less restrictive condition over Requirement 'LR'.

$$\begin{cases}\mathcal{F}_{\mathcal{H}}(\mathrm{X}_k,\mathrm{Pj})=0\\ \mathcal{F}_{\mathcal{L}}(\mathrm{X}_k,\mathrm{Pj})\leq\mathcal{I}(\mathrm{Pj})\end{cases} \tag{9}$$

□

5. SYNTHESIS OF CONTROLLERS DESPITE INCOHERENT REQUIREMENTS

5.1 Control system specifications

Let us consider a hydraulic press with a vertical ram (Fig 3). A safety-light curtain is used to safeguard operators during the movements of the ram. A control panel allows to select the mode of operation: Manual or Automatic mode.

- In Manual mode, all the operations are carried out by pressing the corresponding push-buttons. As soon as a push-button is released, the ram movement stops.
- In Automatic mode, the cycle starts by pressing the 'Start' push-button: the ram is going down and comes back to the up position after the press operation has been done.

Fig. 3. A hydraulic press and its human-machine interfaces

Inputs and outputs of the controller (Fig. 4a) Each movement of the ram is controlled thanks to a Boolean output ('goUp' and 'goDown'). The controller is informed of the position of the ram thanks to inputs 'up' and 'down'. The safety light curtain is connected to input 'lc' (lc $=_b 1$ when the operator is not in the detection zone of the light curtain). The control panel of the press is composed of an emergency stop button (input: 'esb'), a three position center-off switch for the operating mode selection ('sbA', off: no mode selected, 'sbM') and four push-buttons (inputs: 'vpb', 'spb', 'uppb' and 'dopb').

Control laws to synthetize It is not possible to identify automatically how many and which state variables must be used. The designer has to fix the state variables by expertise.

For this case study, we propose to use 5 state variables: one for each output; one for each mode of operation (Automatic, Manual) [6] and one for characterizing a state where the press is in a failure mode (Fail). According to this choice, 5 15-variable switching functions (Auto, Manual, Fail, GoUp and GoDown) have to be synthesized (Fig. 4b). The 15 projection-functions of $F_{15}(B)$ are therefore:

- The 10 switching functions (Up, Down, Lc, Esb, SbA, SbM, Vpb, Spb, Uppb, and Dopb) which characterize the behavior of the inputs of the controller. They are defined as follows:

$$\begin{aligned}\text{Up}: \quad & B^{15}\to B\\ & (\text{up}[k],\ldots,\text{goDown}[k-1])\mapsto \text{up}[k]\end{aligned}$$

- The 5 switching functions ($_p$Auto, $_p$Manual, $_p$Fail, $_p$GoUp and $_p$GoDown) which characterize the previous behavior of the state variables of the controller. They are defined as follows:

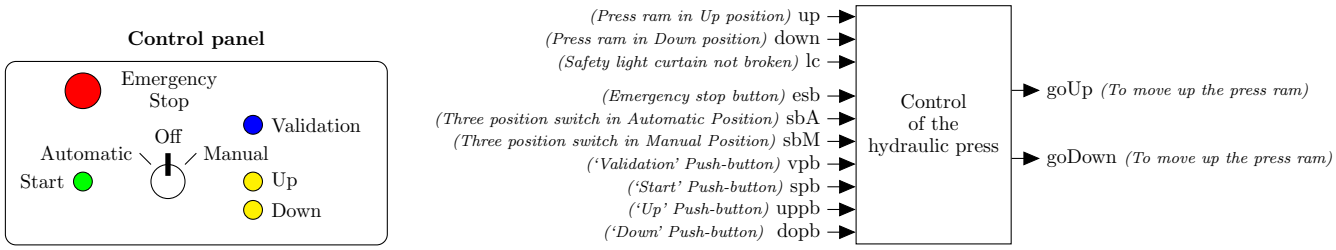
$$\begin{aligned}{}_p\text{Auto}: \quad & B^{15}\to B\\ & (\text{up}[k],\ldots,\text{goDown}[k-1])\mapsto \text{auto}[k-1]\end{aligned}$$

5.2 Algebraic formalization of requirements

The complete formalization of the behavior of the hydraulic press is given Fig. 4c. In order to illustrate the power of expression of relations Equality and Inclusion,

[6] These mode variables must not be confused with inputs 'sbA' or 'sbM' which are the demand of operator for reaching these modes.

a) Inputs and outputs of the controller



b) General form of the expected control laws

$$\begin{cases} \text{auto}[k] = \text{Auto}(\text{up}[k], \ldots, \text{dopb}[k], \text{auto}[k-1], \text{manual}[k-1], \text{fail}[k-1], \text{goUp}[k-1], \text{goDown}[k-1]) \\ \text{manual}[k] = \text{Manual}(\text{up}[k], \ldots, \text{dopb}[k], \text{auto}[k-1], \text{manual}[k-1], \text{fail}[k-1], \text{goUp}[k-1], \text{goDown}[k-1]) \\ \text{fail}[k] = \text{Fail}(\text{up}[k], \ldots, \text{dopb}[k], \text{auto}[k-1], \text{manual}[k-1], \text{fail}[k-1], \text{goUp}[k-1], \text{goDown}[k-1]) \\ \text{goUp}[k] = \text{GoUp}(\text{up}[k], \ldots, \text{dopb}[k], \text{auto}[k-1], \text{manual}[k-1], \text{fail}[k-1], \text{goUp}[k-1], \text{goDown}[k-1]) \\ \text{goDown}[k] = \text{GoDown}(\text{up}[k], \ldots, \text{dopb}[k], \text{auto}[k-1], \text{manual}[k-1], \text{fail}[k-1], \text{goUp}[k-1], \text{goDown}[k-1]) \end{cases}$$

c) Complete formal specification

Requirements:

$$\begin{cases} \textbf{R1} & \text{Auto} \cdot \text{Manual} + \text{Auto} \cdot \text{Fail} + \text{Manual} \cdot \text{Fail} = 0 \\ \textbf{R2} & \text{Esb} \leq \text{Fail} \\ \textbf{R3} & \text{Up} \cdot \text{Down} \leq \text{Fail} \\ \textbf{R4} & \overline{\text{Fail}} \cdot {}_p\text{Fail} \leq \text{Lc} \\ \textbf{R5} & \text{Auto} = \text{SbA} \\ \textbf{R6} & \text{Auto} \cdot \overline{{}_p\text{Auto}} \leq \text{Up} \cdot \text{Lc} \\ \textbf{R7} & \text{Auto} \cdot \overline{{}_p\text{Auto}} \leq \text{Vpb} \\ \textbf{R8} & \text{Auto} \cdot \overline{\text{Lc}} \leq \text{Up} \\ \textbf{R9} & \overline{\text{Auto}} \cdot {}_p\text{Auto} \cdot \overline{(\text{Lc} + \text{Up})} \leq \text{Fail} \\ \textbf{R10} & \text{Manual} = \text{SbM} \\ \textbf{R11} & \text{Manual} \cdot \overline{{}_p\text{Manual}} \leq \text{Lc} \\ \textbf{R12} & \text{Manual} \cdot \overline{{}_p\text{Manual}} \leq \text{Vpb} \end{cases} \quad \begin{cases} \textbf{R20} & \text{GoUp} \cdot \text{GoDown} = 0 \\ \textbf{R21} & \text{Up} \leq \overline{\text{GoUp}} \\ \textbf{R22} & \text{Down} \leq \overline{\text{GoDown}} \\ \textbf{R23} & (\text{GoUp} + \text{GoDown}) \leq \text{Lc} \\ \textbf{R24} & (\text{GoUp} + \text{GoDown}) \leq (\text{Auto} + \text{Manual}) \\ \textbf{R25} & \text{Manual} \leq ((\text{GoUp} \cdot \text{Uppb}) + (\overline{\text{GoUp}} \cdot \overline{\text{Uppb}})) \\ \textbf{R26} & \text{Manual} \leq ((\text{GoDown} \cdot \text{Dopb}) + (\overline{\text{GoDown}} \cdot \overline{\text{Dopb}})) \\ \textbf{R27} & \text{Manual} \cdot (\text{Uppb} \cdot \text{Dopb}) \leq \overline{(\text{GoUp} + \text{GoDown})} \\ \textbf{R28} & \text{Auto} \cdot (\overline{\text{GoUp}} \cdot {}_p\text{GoUp}) \leq \text{Up} \\ \textbf{R29} & \text{Auto} \cdot (\overline{\text{GoDown}} \cdot {}_p\text{GoDown}) \leq \text{Down} \\ \textbf{R30} & \text{Auto} \leq (\text{Up} + \text{GoUp} + \text{GoDown}) \\ \textbf{R31} & \text{Auto} \leq ((\text{GoDown} \cdot \overline{{}_p\text{GoDown}}) \cdot (\text{Up} \cdot \text{Spb}) \\ & \quad + \overline{(\text{GoDown} \cdot \overline{{}_p\text{GoDown}})} \cdot \overline{(\overline{{}_p\text{GoDown}} \cdot \text{Up} \cdot \text{Spb})}) \end{cases}$$

Priority rules:

$$\begin{cases} \{\text{R1,R2,R3,R4}\} \gg \text{R5} & \textit{(* Fail mode has priority on Automatic mode. *)} \\ \{\text{R6,R7,R8}\} \gg \text{R5} & \textit{(* Starting rules have priority on the switch position rule. *)} \\ \{\text{R1,R2,R3,R4,R8}\} \gg \text{R10} & \textit{(* Fail mode has priority on Manual mode. *)} \\ \{\text{R11,R12}\} \gg \text{R10} & \textit{(* Starting rules have priority on the switch position rule. *)} \\ \{\text{R21,R23,R27}\} \gg \text{R25} & \textit{(* Safety requirements have priority on functional requirements. *)} \\ \{\text{R21,R23,R27}\} \gg \text{R26} & \textit{(* Safety requirements have priority on functional requirements. *)} \\ \text{R23} \gg \text{R28} & \textit{(* Safety requirements have priority on functional requirements. *)} \\ \text{R23} \gg \text{R29} & \textit{(* Safety requirements have priority on functional requirements. *)} \\ \text{R23} \gg \text{R30} & \textit{(* Safety requirements have priority on functional requirements. *)} \\ \{\text{R22,R23}\} \gg \text{R31} & \textit{(* Safety requirements have priority on functional requirements. *)} \end{cases}$$

Assumptions:

$$\begin{cases} \textbf{A1} & \text{SbA} \cdot \text{SbM} = 0 \quad \textit{(* The 2 positions of the switch button are exclusive. *)} \\ \textbf{A2} & {}_p\text{Auto} \cdot {}_p\text{Manual} + {}_p\text{Auto} \cdot {}_p\text{Fail} + {}_p\text{Manual} \cdot {}_p\text{Fail} = 0 \quad \textit{(* Consequence of R1. *)} \\ \textbf{A3} & {}_p\text{GoUp} \cdot {}_p\text{GoDown} = 0 \quad \textit{(* Consequence of R20. *)} \end{cases}$$

Optimization criteria:

Minimisation of: Fail

d) Solution obtained by symbolic calculation

$$\begin{cases} \text{Fail} & = \text{Esb} + \text{Up} \cdot \text{Down} + \overline{\text{Lc}} \cdot {}_p\text{Fail} + \overline{\text{Up}} \cdot \overline{\text{Lc}} \cdot {}_p\text{Auto} \\ \text{Auto} & = \overline{\text{Esb}} \cdot \text{SbA} \cdot (\text{Up} \cdot \overline{\text{Down}} \cdot \text{Lc} \cdot \text{Vpb} + {}_p\text{Auto} \cdot (\text{Up} \cdot \overline{\text{Down}} + \overline{\text{Up}} \cdot \text{Lc})) \\ \text{Manual} & = \overline{\text{Esb}} \cdot \text{SbM} \cdot \overline{(\text{Up} \cdot \text{Down})} \cdot (\text{Lc} \cdot \text{Vpb} + {}_p\text{Manual}) \\ \text{GoUp} & = \overline{\text{Esb}} \cdot \overline{\text{Up}} \cdot \text{Lc} \cdot (\text{SbA} \cdot {}_p\text{Auto} \cdot (\text{Down} + \overline{{}_p\text{GoDown}}) + \text{SbM} \cdot \text{Uppb} \cdot \overline{\text{Dopb}} \cdot (\text{Vpb} + {}_p\text{Manual})) \\ \text{GoDown} & = \overline{\text{Esb}} \cdot \overline{\text{Down}} \cdot \text{Lc} \cdot (\text{SbA} \cdot (\text{Spb} \cdot \text{Up} \cdot (\text{Vpb} + {}_p\text{Auto}) + {}_p\text{GoDown} \cdot ({}_p\text{Auto} + (\text{Vpb} \cdot \text{Up}))) \\ & \quad + \text{SbM} \cdot \text{Dopb} \cdot \overline{\text{Uppb}} \cdot (\text{Vpb} + {}_p\text{Manual})) \end{cases}$$

e) Control laws of the hydraulic press

$$\begin{cases} \text{fail}[k] & = \text{esb}[k] \vee \text{up}[k] \wedge \text{down}[k] \vee \neg\text{lc}[k] \wedge \text{fail}[k-1] \vee \neg\text{up}[k] \wedge \neg\text{lc}[k] \wedge \text{auto}[k-1] \\ \text{auto}[k] & = \neg\text{esb}[k] \wedge \text{sbA}[k] \wedge (\text{up}[k] \wedge \neg\text{down}[k] \wedge \text{lc}[k] \wedge \text{vpb}[k] \vee \text{auto}[k-1] \wedge (\text{up}[k] \wedge \neg\text{down}[k] \vee \neg\text{up}[k] \wedge \text{lc}[k])) \\ \text{manual}[k] & = \neg\text{esb}[k] \wedge \text{sbM}[k] \wedge \neg(\text{up}[k] \wedge \text{down}[k]) \wedge (\text{lc}[k] \wedge \text{vpb}[k] \vee \text{manual}[k-1]) \\ \text{goUp}[k] & = \neg\text{esb}[k] \wedge \neg\text{up}[k] \wedge \text{lc}[k] \wedge (\text{sbA}[k] \wedge \text{auto}[k-1] \wedge (\text{down}[k] \vee \neg\text{goDown}[k-1]) \vee \text{sbM}[k] \wedge \text{uppb}[k] \wedge \neg\text{dopb}[k] \wedge (\text{vpb}[k] \vee \text{manual}[k-1])) \\ \text{goDown}[k] & = \neg\text{esb}[k] \wedge \neg\text{down}[k] \wedge \text{lc}[k] \wedge (\text{sbA}[k] \wedge (\text{spb}[k] \wedge \text{up}[k] \wedge (\text{vpb}[k] \vee \text{auto}[k-1]) \vee \text{goDown}[k-1] \wedge (\text{auto}[k-1] \vee (\text{vpb}[k] \wedge \text{up}[k]))) \\ & \quad \vee \text{sbM}[k] \wedge \text{dopb}[k] \wedge \neg\text{uppb}[k] \wedge (\text{vpb}[k] \vee \text{manual}[k-1])) \end{cases}$$

Fig. 4. Details of the case study

all the requirements used for the synthesis of the operation modes are expressed in textual form hereafter:

- **R1** The three modes (Automatic, Manual and Fail) are exclusive.
- **R2** While the Emergency stop button 'esb' is pressed, the press is in Failure mode.
- **R3** If the observed position of the ram is both 'up' and 'down', the press is in Failure mode.
- **R4** For leaving the Failure mode, the operator must not be in the detection zone of the light curtain.
- **R5** The press is in Automatic mode if and only if the three position center-off switch is turned on 'sbA' position.
- **R6** For reaching the Automatic mode, the press ram must be in 'up' position and the operator must not be in the detection zone of the light curtain.
- **R7** For reaching the Automatic mode, the 'vpb' push-button must be pressed.
- **R8** During the Automatic mode, the operator can be in the detection zone of the light curtain without to be in danger only if the press ram is in 'up' position.
- **R9** During the Automatic mode, if the operator is detected by the light curtain while the press ram is not in 'up' position, one has to switch into the Failure mode.
- **R10** The press is in Manual mode if and only if the three position center-off switch is turned on 'sbM' position.
- **R11** For reaching the Manual mode, the press ram must be in 'up' position and the operator must not be in the detection zone of the light curtain.
- **R12** For reaching the Manual mode, the 'vpb' push-button must be pressed.

The relation "Inclusion" is the more used in our approach since it allows the expression of both *sufficient* (as R2) and *necessary* conditions (as R4).

5.3 Synthesis process

In traditional design methods, the design procedure of a logic controller is not a linear process, but an iterative one converging to an acceptable solution. At the beginning of the design, requirements are neither complete nor without errors. Most often, new requirements are added during the search of solutions, others are corrected. This complementary information is given by the designer after analysis of the partial solutions he found or when inconsistencies have been detected. If we do not make the hypothesis that the specifications are complete and consistent, designing a controller with a synthesis technique will also be an iterative process in which the designer plays an important role.

Synthesis of the operation modes For this case study, we have started with the synthesis of operation modes. The first four studied requirements have been: R1, R2, R5 and R10. For this subset of requirements, the result given by our software tool was the following inconsistency condition:

$$\mathcal{I}_0 = \mathrm{SbA} \cdot \mathrm{SbM} + \mathrm{Esb} \cdot \mathrm{SbA} + \mathrm{Esb} \cdot \mathrm{SbM}$$

Since requirements are declared as inconsistent, we have to give complementary information to precise our specification. By analyzing each term of this formula, it is possible to detect the origin of the inconsistency:

- SbA·SbM: What happens if the Automatic mode and the Manual mode are simultaneously selected? We consider that it is not possible (due to the technology of the three position center-off switch) and we have added Assumption A1.
- Esb · SbA: What happens if the Emergency stop button is activated during the Automatic mode? We consider that the Fail mode has priority on Automatic mode (for security reasons) and we have added the priority rule: {R1,R2} $\gg$ R5.
- Esb · SbM: What happens if the Emergency stop button is activated during the Manual mode? We consider that the Fail mode has priority on Manual mode (for security reasons) and we have added the priority rule: {R1,R2} $\gg$ R10.

With this complementary information, Problem (10) admits as parametric solution (11):

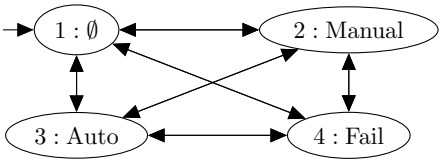
$$\left[\begin{array}{l}
\text{Requirements:} \\
\quad \begin{cases} \text{R1} & \mathrm{Auto} \cdot \mathrm{Manual} + \mathrm{Auto} \cdot \mathrm{Fail} \\ & \qquad + \mathrm{Manual} \cdot \mathrm{Fail} = 0 \\ \text{R2} & \mathrm{Esb} \leq \mathrm{Fail} \\ \text{R5} & \mathrm{Auto} = \mathrm{SbA} \\ \text{R10} & \mathrm{Manual} = \mathrm{SbM} \end{cases} \\
\text{Priority rules:} \\
\quad \begin{cases} \{\text{R1,R2}\} \gg \text{R5} \\ \{\text{R1,R2}\} \gg \text{R10} \end{cases} \\
\text{Assumptions:} \\
\quad \{\text{A1} \quad \mathrm{SbA} \cdot \mathrm{SbM} = 0
\end{array}\right. \tag{10}$$

$$\begin{cases} \mathrm{Fail} = \mathrm{Esb} + p_1 \cdot \overline{\mathrm{SbA}} \cdot \overline{\mathrm{SbA}} & p_1 \in F_{15}(B) \\ \mathrm{Auto} = \overline{\mathrm{Esb}} \cdot \mathrm{SbA} \\ \mathrm{Manual} = \overline{\mathrm{Esb}} \cdot \mathrm{SbM} \end{cases} \tag{11}$$

By gradually adding all the requirements and selecting the solution which minimizes the Fail mode (the parameter p_1 is fixed to 0) the solution we obtain is proposed on Fig. 4d (the first three equations). In an illustrative purpose, a state model representation of the synthesized operation modes management, automatically built thanks to Guignard (2011), is given in (Fig. 5). The transition conditions are non-trivial Boolean expressions of inputs that take into account the whole set of specifications given in Section 5.2.

Synthesis of the reactive control laws To obtain the complete control law of the hydraulic press, the solution previously obtained for the operation modes has to be completed thanks to requirements R20 to R31. All the priorities rules and assumptions used for this case study are given on Fig. 4c. The solution we obtain is proposed on Fig. 4d.

The control laws presented Fig. 4e have been obtained by translating the expression of the five unknowns according to the projection-functions into relations between recurrent Boolean equations. These control laws can be automatically translated in the syntax of the Ladder Diagram language (IEC 61131-3 (2003)) before being implemented



$$
\begin{cases}
E_{1-2} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbM} \wedge \text{vpb} \wedge (\neg\text{up} \vee \neg\text{down}) \\
E_{1-3} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbA} \wedge \text{vpb} \wedge \text{up} \wedge \neg\text{down} \\
E_{1-4} = \text{esb} \vee \text{up} \wedge \text{down} \\
E_{2-1} = \neg\text{esb} \wedge \neg\text{sbM} \wedge (\neg\text{up} \vee \neg\text{down} \wedge \neg(\text{lc} \wedge \text{sbA} \wedge \text{vpb})) \\
E_{2-3} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbA} \wedge \text{vpb} \wedge \text{up} \wedge \neg\text{down} \\
E_{2-4} = \text{esb} \vee \text{up} \wedge \text{down} \\
E_{3-1} = \neg\text{esb} \wedge \neg\text{sbA} \wedge (\neg\text{lc} \wedge \text{up} \wedge \neg\text{down} \vee \\
\qquad (\neg\text{sbM} \vee \neg\text{vpb}) \wedge (\text{up} \wedge \neg\text{down} \vee \text{lc} \wedge \neg\text{up})) \\
E_{3-2} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbM} \wedge \text{vpb} \wedge (\neg\text{up} \vee \neg\text{down}) \\
E_{3-4} = \text{esb} \vee \text{up} \wedge \text{down} \vee \neg\text{up} \wedge \neg\text{lc} \\
E_{4-1} = \neg\text{esb} \wedge \text{lc} \wedge (\neg\text{up} \vee \neg\text{down}) \wedge \\
\qquad (\neg\text{sbA} \wedge \neg\text{sbM} \vee \neg\text{vpb} \vee \text{sbA} \wedge \neg\text{up}) \\
E_{4-2} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbM} \wedge \text{vpb} \wedge (\neg\text{up} \vee \neg\text{down}) \\
E_{4-3} = \neg\text{esb} \wedge \text{lc} \wedge \text{sbA} \wedge \text{vpb} \wedge \text{up} \wedge \neg\text{down}
\end{cases}
$$

Fig. 5. State model of modes of operation

into a PLC. The complete code is composed of only 10 rungs.

6. CONCLUSION

Many research works in the field of DES aim at formalizing steps of the systems life cycle. Since 20 years, significant progresses have been obtained for the synthesis, verification, performance evaluation, diagnosis ... of DESs. Nevertheless, one of the common difficulties of these works is that there comes a time when it is necessary to pass from the informal expression of the knowledge of a system to its formalization. Few works have paid attention to this important task which is very error prone. In this paper, we proposed an iterative process that allows coping with inconsistencies of the requirements during the synthesis of the controller. The framework in which we proposed this approach is an algebraic synthesis method. Since the problem is located to the frontier between formal and informal, intervention of the designer is necessary. Nevertheless, we have shown that this intervention can be guided by the results the formal method provides.

REFERENCES

Brown, F.M. (2003). *Boolean Reasoning: The Logic of Boolean Equations*. Dover Publications.

Filiot, E., Jin, N., and Raskin, J.F. (2011). Antichains and compositional algorithms for LTL synthesis. *Form. Methods Syst. Des.*, 39(3), 261–296.

Gohari, P. and Wonham, W.M. (2000). On the complexity of supervisory control design in the RW framework. *IEEE Trans. on Systems, Man, and Cybernetics - part B*, 30(5), 643–652.

Grimaldi, R.P. (2004). *Discrete and Combinatorial Mathematics: An Applied Introduction*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, Fifth edition.

Guignard, A. (2011). *Symbolic generation of the automaton representing an algebraic description of a logic system*. Master's thesis, ENS Cachan.

Hanisch, H.M., Lüder, A., and Tbieme, J. (1998). A modular plant modeling technique and related controller synthesis problems. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, 686–691.

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

Hietter, Y., Roussel, J.M., and Lesage, J.J. (2008a). Algebraic synthesis of transition conditions of a state model. In *Proceedings of 9th International Workshop On Discrete Event Systems (WODES'08)*, 187–192. Göteborg, Sweden.

Hietter, Y., Roussel, J.M., and Lesage, J.J. (2008b). Algebraic synthesis of dependable logic controllers. In *Proceedings of 17th IFAC World Congress, 2008*, 4132–4137. Seoul, South Korea.

Hietter, Y., Roussel, J.M., and Lesage, J.J. (2008c). Calcul des conditions de transition d'un Réseau de Petri par synthèse algébrique. In *Proceedings of CIFA'08*, CDRom paper 83. Bucharest, Romanian. 6 pages.

Huffman, D.A. (1954). The synthesis of sequential switching circuits. *J. of the Franklin Institute*, 257(3-4), 161–190 and 275–303.

IEC 61131-3 (2003). *IEC 61131-3 Standard: Programmable controllers - Part 3: Programming languages*. International Electrotechnical Commission, Geneva, Switzerland, 2 edition.

IEC 61508 (1998). *IEC 61508 Standard: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*. International Electrotechnical Commission, Geneva, Switzerland.

Leroux, H. (2011). *Algebraic synthesis of logical controllers with optimization criteria*. Master's thesis, ENS Cachan.

Machado, J., Denis, B., Lesage, J.J., Faure, J.M., and Fereira, J. (2006). Logic controllers dependability verification using a plant model. In *Proceedings of the 3rd IFAC Workshop on Discrete-Event System Design, DESDes'06*, pp. 37– 42. Rydzyna, Poland.

Mealy, G.H. (1955). A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5), 1045–1079.

Moore, E.F. (1956). Gedanken Experiments on Sequential Machines. In *Automata Studies*, 129–153. Princeton U.

Pnueli, A. and Rosner, R. (1989). On the synthesis of a reactive module. In *Proceedings of the 16th ACM symposium on Principles of programming languages*, POPL'89, 179–190. ACM, New York, NY, USA.

Ramadge, P.J.G. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE Transactions on Automatic Control*, 77(1), 81–98.

Roussel, J.M. and Giua, A. (2005). Designing dependable logic controllers using the supervisory control theory. In *Proceedings of the 16th IFAC World Congress, 2005*. Praha, Czech Republic. CDRom paper 4427, 6 pages.

Rudeanu, S. (2001). *Lattice Functions and Equations (Discrete Mathematics and Theoretical Computer Science)*. Springer.

Zowghi, D. and Offen, R. (1997). A logical framework for modeling and reasoning about the evolution of requirements. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, 247–257.