HAL
open science

Algebraic Synthesis of Transition Conditions of a State Model

Yann Hietter, Jean-Marc Roussel, Jean-Jacques Lesage

To cite this version:

Yann Hietter, Jean-Marc Roussel, Jean-Jacques Lesage. Algebraic Synthesis of Transition Conditions of a State Model. 9th International Workshop On Discrete Event Systems (WODES'08), May 2008, Göteborg, Sweden. pp. 187-192. hal-00353634

HAL Id: hal-00353634

https://hal.science/hal-00353634v1

Submitted on 15 Jan 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Algebraic Synthesis of Transition Conditions of a State Model

Yann Hietter, Jean-Marc Roussel and Jean-Jacques Lesage IEEE member

***Abstract*— The synthesis method presented in this paper has been developed to automatically design logic controllers. In this paper, we show how to use this approach in the specific case where a designer must derive a particular controller from a generic model. The instantiation of the model is completely achieved by an algebraic synthesis. To illustrate the approach, the example of a water supply system is used and the generic model of the controller is given under the form of a Sequential Function Chart (SFC).**

I. Introduction

Logic controllers are used in a very large number of systems such as embedded systems, transport systems or power plants, ... that often are critical systems. This explains why the dependability of logic controllers is a major concern for control engineers. To improve the dependability of logic controllers, many formal methods have been proposed over the last twenty years (formal languages, automatic synthesis, verification by model-checking, automatic generation of tests, diagnosis, ...). In terms of dependability, these approaches are not competing but they are complementary.

The method presented in this paper belongs to the class of synthesis approaches, i.e. that aim at deducing the control laws from the specifications and from the dependability expected properties, without the involvement of the designer (or at least by limiting his involvement to a strict minimum). The most important progress in the field of synthesis of Discrete Event Systems (DES) has been carried out thanks to the Supervisory Control Theory (SCT) defined by Ramadge and Wonham [1]. Since the publication of this innovative theoretical paradigm, many researchers have extended the first results, both in theoretical and methodological ways. Their countless publications allowed to enlarge the power of the SCT for designing an optimal supervisor, but also to point out the limits of this approach for practical design and the implementation of controllers [2]. The synthesis method we present in this paper is radically different from the SCT for two main reasons. First it is limited to a sub-class of DES – the logical systems – but it aims to design control laws (not a supervisor); secondly it is purely algebraic.

In a previous paper [3], we used our approach to synthesize the whole control law of a sequential system. We showed that different formalisms (such as finite states machines, logical expressions, timing diagrams, ...) could be used jointly to express functional requirements, priority or safety rules, ... The translation of these fragments of specifications into algebraic equations makes up an equations system whose resolution gives the space of solutions (if it exists). The choice of a control law in this space can be done by using an optimization criterion. In [3] we showed how to choose the most dependable control law.

In this paper, we are going to deal with the specific – but very frequent – case where the know-how of the designer enables him to a priori build the global form of the solution (or of a part of the whole solution). In this case the designer often expresses the control law by using a generic model. The specific solution of a given control problem is then obtained by instanciating the generic model. We are now going to show how our synthesis method can be used to find and optimize these instances of generic models.

This paper is organized as follows. Section 2 gives the broad lines of our method. In Section 3 we describe a small example of logical controller to be designed. Section 4 presents the formal framework used and the mathematical results on which this method is based. Our algebraic synthesis method is then used in Section 5 by treating the example given in Section 3.

II. Synthesis method

A global survey of our synthesis method is given in figure 1. First of all, it is important to describe the kinds of input data that are needed for the controller synthesis. In fact, any knowledge of the system whose controller is to be designed can be expressed by the designer. These fragments of specifications can be functional requirements, dangerous situations that must be forbidden, partial control laws specifications, ... Different formalisms can be used to express these fragments of specifications: state models (automata, Petri nets, state charts, ...), timing diagrams, logic gates diagrams, ... In a more general way, any formalism that can be translated into Boolean equations can be used.

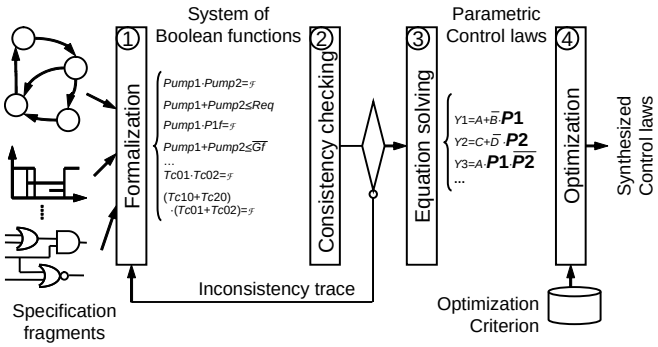

Fig. 1. A global survey of our synthesis method

Y. Hietter, J.-M. Roussel and J.-J. Lesage are with LURPA, ENS Cachan, UniverSud, 61 Av. du President Wilson, 94235 Cachan, France {yann.hietter, jean-marc.roussel, jean-jacques.lesage}@lurpa.ens-cachan.fr

The first step of the method is the formalization of the heterogeneous fragments of specifications into relations between Boolean functions. This step is largely automated (see for example in Section 5 the translation of a SFC in Boolean functions). At the end of this step, the set of specifications take the form of a system of Boolean equations between Boolean functions. The second step, totally automated, is the checking of the consistency of this system of equations. If an inconsistency is detected, the concerned specifications must be modified. The third step is the resolution of the system of equations. It will be detailed in Section 4 of this paper. The result of this step is a set of parametric solutions, i.e. the set of control laws that satisfy all the fragments of specification. The last step of the method aims to choose among the set of solutions the best control law for a given problem. At present this step is not implemented. The method which can be used to find the optimal control is being discussed in Section 5.

III. Description of the case study

In this paper, we treat the specific case where the know-how of the designer enables him to build the global form of the solution. We are illustrating the treatment of this class of problems with the example of the control of a water supply (figure 2). The system is composed of two pumps that supply water to customers from a tank. In the nominal mode only one pump is required to supply water. The logical controller that controls this system has 4 inputs and 2 outputs:

- **req**: request of water by customers,
- **p1f**: detected failure of pump 1,
- **p2f**: detected failure of pump 2,
- **gf**: detected global failure. The water supply must be stopped.

pump1 and **pump2** are the two output signals that respectively control the operation of each pump.

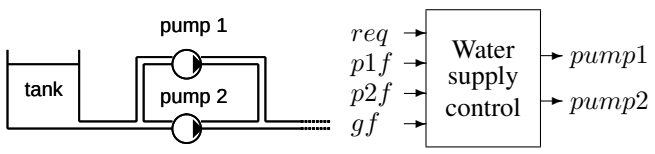

Fig. 2. (a) Water supply plant (b) Controller inputs and outputs

An experimented specialist of automation recognizes in these informal and incomplete requirements a problem of redundancy control. He knows different parametrical sequential structures that are solutions of this classical problem (one of them is given in figure 3 under the form of a SFC). The real problem to solve is in fact fixing the parameters of this solution, i.e. the conditions for the activation of one and only one pump ($Tc01$ and $Tc02$), the conditions of switching from one pump to the other ($Tc12$ and $Tc21$) and the conditions for stopping the water supply ($Tc10$ and $Tc20$).

A priori, $Tcij$ are Boolean functions of inputs and state variables of the SFC.

$$Tcij = f_{ij}(req, p1f, p2f, gf, S0, S1, S2)$$

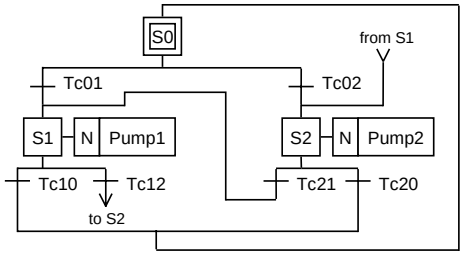

Fig. 3. A partial SFC, expert parametric solution to the water supply

$2^{2^7} \simeq 3 * 10^{38}$ different Boolean functions $Tcij$ can be defined with 7 Boolean variables. The method described in Section 2 allows us to calculate (under parametric form) the subset of these $Tcij$ that respect all the requirements. The choice for a particular solution is done by fixing the parameters (which also are Boolean functions). The basic elements we have to handle in our synthesis method are therefore Boolean functions (and not Boolean variables) like $Tcij$. The mathematical framework, which we developed to handle these BFs and relations between the BFs, are defined hereafter.

IV. Mathematical framework

A. Boolean functions

From a mathematical point of view, a Boolean Function (BF) is a function (G_i) from $\mathbb{B}^m \rightarrow \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ is a Boolean domain. For each m-tuple of Boolean variables $(b_1, ..., b_m)$, $2^{(2^m)}$ different BFs can be defined.

Let $(b_1, ..., b_m)$ be a m-tuple of Boolean variables. Let Γ_m be the set of the $2^{(2^m)}$ possible BFs. Γ_m contains two BFs which are defined as follows:

$$\begin{array}{rrcl} \mathcal{F}: & \mathbb{B}^m & \rightarrow & \mathbb{B} \\ & (b_1, ..., b_m) & \mapsto & 0 \end{array} \quad \Bigg| \quad \begin{array}{rrcl} \mathcal{T}: & \mathbb{B}^m & \rightarrow & \mathbb{B} \\ & (b_1, ..., b_m) & \mapsto & 1 \end{array}$$

$\mathcal{F}$ is the "always false" function and $\mathcal{T}$ is the "always true" function.

Γ_m also contains m specific BFs whose image is a Boolean variable of $(b_1, ..., b_m)$. These functions are defined as follows:

$$\begin{array}{rrcl} B_i: & \mathbb{B}^m & \rightarrow & \mathbb{B} \\ & (b_1, ..., b_m) & \mapsto & b_i \end{array}$$

Nota: In this paper, to avoid confusion between Boolean variables and Boolean functions, Boolean variables are noted with lowercase letters (b_i) whereas Boolean functions are noted with uppercase letters (B_i).

As presented in [4], Γ_m with the two binary operations OR (noted "+"), AND (noted "·"), the unary operation NOT (noted "¯") and the two identity elements $\mathcal{T}$ and $\mathcal{F}$, is a Boolean algebra. These three operations are defined as follows:

$\forall G_1, G_2 \in \Gamma_m,$
$(G_1 + G_2)(b_1, ..., b_m) = (G_1(b_1, ..., b_m)) \vee (G_2(b_1, ..., b_m))$
$(G_1 \cdot G_2)(b_1, ..., b_m) = (G_1(b_1, ..., b_m)) \wedge (G_2(b_1, ..., b_m))$
$\overline{G}(b_1, ..., b_m) = \neg(G(b_1, ..., b_m))$

where $\vee$, $\wedge$, $\neg$ are the basic operations on Boolean variables.

Nota: To avoid unreadable formulae, the reference to m-tuple $(b_1, ..., b_m)$ in the notation of BFs is removed from now on.

B. Composition of BFs

As (Γ_m, $+$, $\cdot$, $^{-}$, $\mathcal{F}$, $\mathcal{T}$) is a Boolean algebra, each composition of elements of Γ_m by operations OR, AND and NOT is an element of Γ_m. Two distinct compositions of elements of Γ_m are said "equal" if they have the same image in Γ_m.

Let (G_1, ..., G_k) be k elements of Γ_m. For each composition C of (G_1, ..., G_k), we have proved (but not the place to present it) that it exist two compositions C_1 and C_2 of (G_2, ..., G_k) such as:

$$C(G_1, \ldots, G_k) = C_1(G_2, \ldots, G_k) \cdot \overline{G_1} + C_2(G_2, \ldots, G_k) \cdot G_1$$

Compositions C_1 and C_2 can directly be obtained from C by substitution:

$$\begin{cases} C_1(G_2, \ldots, G_k) = C(\mathcal{F}, G_2, \ldots, G_k) \\ C_2(G_2, \ldots, G_k) = C(\mathcal{T}, G_2, \ldots, G_k) \end{cases}$$

This property is the extension of the decomposition proposed by Shannon [5] for Boolean variables to Boolean functions. In our approach, this property is used to express each composition of BFs with a generic expression.

C. Relations between BFs

The relations between BFs are the cornerstone of our approach as they permit to express fragments of specifications. As (Γ_m, $+$, $\cdot$, $^{-}$, $\mathcal{F}$, $\mathcal{T}$) is a Boolean algebra, there is a supplementary relation [4] to a equality relation based on the partial order between elements of Γ_m. This relation is defined as follows:

*Definition 1.*Inclusion Relation

$$\forall G_1, G_2 \in \Gamma_m, G_1 \leq G_2 \text{ if and only if } G_1 \cdot G_2 = G_1$$

In other terms BFs G_1 and G_2 satisfy relation $G_1 \leq G_2$ (read "G_1 is included into G_2") if and only if the set of values of m-tuple $(b_1, ..., b_m)$ whose image by G_1 is 1 is included into the set of values of m-tuple $(b_1, ..., b_m)$, whose image by G_2 is 1.

In our approach, this relation is often used to express requirements. For example, relation $C \leq D$ can be the formalization of the three following requirements given in natural language:

- When C is true, D is also true.
- It is sufficient to have C to have D.
- It is necessary to have D to have C.

Moreover, relation "Equality" is used to formalize requirements such as:

- C and D are never simultaneously true: $C \cdot D = \mathcal{F}$
- One of BFs C and D is always true: $C + D = \mathcal{T}$

It is important to note that relations expressed by using "Inclusion" and "Equality" can always be rewritten due to the following equivalences:

$$G_1 \leq G_2 \Leftrightarrow G_1 \cdot G_2 = G_1 \tag{1}$$

$$G_1 \leq G_2 \Leftrightarrow G_1 \cdot \overline{G_2} = \mathcal{F} \tag{2}$$

$$G_1 = G_2 \Leftrightarrow G_1 \cdot \overline{G_2} + \overline{G_1} \cdot G_2 = \mathcal{F} \tag{3}$$

$$\begin{cases} G_1 = \mathcal{F} \\ G_2 = \mathcal{F} \end{cases} \Leftrightarrow \quad G_1 + G_2 = \mathcal{F} \tag{4}$$

- Equivalence (2) allows to rewrite each inclusion as an equality whose second member is $\mathcal{F}$.
- Equivalence (3) allows to rewrite each equality as an equality whose second member is $\mathcal{F}$.
- Equivalence (4) allows to rewrite a set of equalities as only one equality whose second member is $\mathcal{F}$.

Finally, all sets of requirements between known BFs B_i and unknown BFs Y_j given under the form of an equality or an inclusion, can always be expressed as follows:

$$R(B_1, \ldots, B_m, Y_1, \ldots, Y_n) = \mathcal{F} \tag{5}$$

where $R(B_1, \ldots, B_m, Y_1, \ldots, Y_n)$ (as Requirements) is a composition of BFs B_i and Y_j by operations OR, AND and NOT. To simplify the notations, the indication of functions B_i for R is omitted from now on.

D. Solving a system of BFs equations

The results presented in this section have been demonstrated for any Boolean algebra, and in particular for the algebra of BFs. These demonstrations are the generalization of previous results obtained for Boolean equations between Boolean variables:

- The condition to have a unique solution for an equation between Boolean variables is given in [6].
- The case of systems of equations between Boolean variables is studied in [7].
- The general form of the set of solutions for equations with several Boolean unknowns is presented in [8].

The generalization of results obtained for Boolean equations of Boolean variables to all Boolean algebra are needed to substitute all proposed proofs by exhaustion by a direct proof. In the case of equation between Boolean variables, proofs by exhaustion are often used as this algebra is composed of only two elements. This strategy could not be used in our case due to the size of Γ_m.

We are now going to present how to solve equations between Boolean functions with only one unknown.

1) Case of one unknown: $R(Y) = \mathcal{F}$: By applying the generalization of Shannon's decomposition to $R(Y)$, we have:

$$R(Y) = R(\mathcal{F}) \cdot \overline{Y} + R(\mathcal{T}) \cdot Y = \mathcal{F} \tag{6}$$

Theorem 1. Let $R(Y) = \mathcal{F}$ be an equation with a single unknown Y expressed on the Boolean algebra of BFs $(\Gamma_m, +, \cdot, ^{-}, \mathcal{F}, \mathcal{T})$.
$R(Y) = \mathcal{F}$ has solutions if and only if:

$$R(\mathcal{F}) \cdot R(\mathcal{T}) = \mathcal{F}$$

In this case, the set of solutions of this equation can be expressed as follows:

$$Y = R(\mathcal{F}) + \overline{R(\mathcal{T})} \cdot P$$

Where P is a constant term of Γ_m whose set of values describes the whole space of solutions.

Indeed, if the condition $R(\mathcal{F}) \cdot R(\mathcal{T}) = \mathcal{F}$ is respected, then $Y = R(\mathcal{F})$ and $Y = \overline{R(\mathcal{T})}$ are two particular solutions.

Furthermore, $R(Y) = \mathcal{F}$ has a single solution iff:

$$R(\mathcal{T}) = \overline{R(\mathcal{F})}$$

Proof: The demonstration of this result is given in three steps:

- First, it is demonstrated that condition $R(\mathcal{F}) \cdot R(\mathcal{T}) = \mathcal{F}$ is necessary and sufficient for the existence of solutions of (6).
- Secondly, it is verified that proposed solution $Y = R(\mathcal{F}) + \overline{R(\mathcal{T})} \cdot P$ is a solution of (6).
- Finally, it is demonstrated that all the solutions can be expressed in the proposed form.

To facilitate the reading of this proof, we will note $A = R(\mathcal{F})$ and $B = R(\mathcal{T})$. With these notations, the studied equation is:

$$A \cdot \overline{Y} + B \cdot Y = \mathcal{F} \tag{7}$$

The existence condition of solutions is $A \cdot B = \mathcal{F}$ and the solutions form is $Y = A + \overline{B} \cdot P$.

- Existence condition of solutions:

$$\begin{aligned} & A \cdot \overline{Y} + B \cdot Y = \mathcal{F} \\ \Leftrightarrow\ & A \cdot \overline{Y} \cdot \mathcal{T} + B \cdot Y \cdot \mathcal{T} = \mathcal{F} \\ \Leftrightarrow\ & A \cdot \overline{Y} \cdot (B + \mathcal{T}) + B \cdot Y \cdot (A + \mathcal{T}) = \mathcal{F} \\ \Leftrightarrow\ & A \cdot B \cdot (\overline{Y} + Y) + A \cdot \overline{Y} \cdot \mathcal{T} + B \cdot Y \cdot \mathcal{T} = \mathcal{F} \\ \Leftrightarrow\ & A \cdot B \cdot \mathcal{T} + A \cdot \overline{Y} + B \cdot Y = \mathcal{F} \\ \Leftrightarrow\ & \begin{cases} A \cdot B = \mathcal{F} \\ A \cdot \overline{Y} + B \cdot Y = \mathcal{F} \end{cases} \end{aligned}$$

Condition $A \cdot B = \mathcal{F}$ is then necessary for the existence of solutions of (7). This condition is sufficient too, because when this condition is true, $Y = A$ is a solution.

$$\begin{cases} A \cdot B = \mathcal{F} \\ Y = A \end{cases} \Rightarrow$$
$$A \cdot \overline{Y} + B \cdot Y = A \cdot \overline{A} + A \cdot B = \mathcal{F} + A \cdot B = A \cdot B = \mathcal{F}$$

- $Y = A + \overline{B} \cdot P$ is a solution of (7):

$$\begin{cases} A \cdot B = \mathcal{F} \\ Y = A + \overline{B} \cdot P \end{cases} \Rightarrow$$
$$\begin{aligned} A \cdot \overline{Y} + B \cdot Y &= A \cdot \overline{(A + \overline{B} \cdot P)} + B \cdot (A + \overline{B} \cdot P) \\ &= A \cdot \overline{A} \cdot \overline{(\overline{B} \cdot P)} + (A \cdot B + B \cdot \overline{B} \cdot P) \\ &= \mathcal{F} + A \cdot B + \mathcal{F} = A \cdot B = \mathcal{F} \end{aligned}$$

- All the solutions of (7) can be expressed under the form $Y = A + \overline{B} \cdot P$ where P is a parameter, element of Γ_m.

For this proof, this intermediate result is needed:

$$\forall (G_1, G_2) \in \Gamma_m^2, \exists (P_1, P_2) \in \Gamma_m^2 \quad G_1 = \overline{G_2} \cdot P_1 + G_2 \cdot P_2$$

This result is a particular case of the generalisation of the expansion of Shannon.

According to this result, we may assume:

$$\forall (Y, A, B) \in \Gamma_m^3, \exists (P_1, P_2, P_3, P_4) \in \Gamma_m^4$$

$$Y = \overline{A} \cdot \overline{B} \cdot P_1 + \overline{A} \cdot B \cdot P_2 + A \cdot \overline{B} \cdot P_3 + A \cdot B \cdot P_4 \tag{8}$$

As Y is a solution of (7), we have:

$$A \cdot \overline{Y} + B \cdot Y = \mathcal{F} \quad \Leftrightarrow \quad \begin{cases} A \cdot \overline{Y} = \mathcal{F} \\ B \cdot Y = \mathcal{F} \end{cases}$$

$$\begin{aligned} A \cdot \overline{Y} = \mathcal{F} \quad & \Leftrightarrow A \leq Y \\ & \Leftrightarrow A \cdot Y = A \\ & \Leftrightarrow A \cdot (\overline{A} \cdot \overline{B} \cdot P_1 + \overline{A} \cdot B \cdot P_2 \\ & \qquad + A \cdot \overline{B} \cdot P_3 + A \cdot B \cdot P_4) = A \\ & \Leftrightarrow A \cdot \overline{B} \cdot P_3 + A \cdot B \cdot P_4 = A \end{aligned}$$

Thus (8) becomes: $Y = \overline{A} \cdot \overline{B} \cdot P_1 + \overline{A} \cdot B \cdot P_2 + A$

$$\begin{aligned} B \cdot Y = \mathcal{F} \quad & \Leftrightarrow B \cdot (\overline{A} \cdot \overline{B} \cdot P_1 + \overline{A} \cdot B \cdot P_2 + A) = \mathcal{F} \\ & \Leftrightarrow A \cdot B + \overline{A} \cdot B \cdot P_2 = \mathcal{F} \\ & \Leftrightarrow \mathcal{F} + \overline{A} \cdot B \cdot P_2 = \mathcal{F} \end{aligned}$$

Finally (8) is: $Y = \overline{A} \cdot \overline{B} \cdot P_1 + A = A + \overline{B} \cdot P_1$

By coming back to the first notations, this result can be expressed by:

$$Y = R(\mathcal{F}) + \overline{R(\mathcal{T})} \cdot P_1$$

□

2) Case of several unknowns: $R(Y_1, \ldots, Y_n) = \mathcal{F}$: Starting from the simplified case of one equation with only one unknown, we have generalized the previous result to one equation with n unknown BFs.

For 2 unknowns, the general form of equation (5) is:

$$R(Y_1, Y_2) = \mathcal{F}$$

By applying the generalization of Shannon's decomposition, we have:

$$\begin{aligned} R(Y_1, Y_2) = & R(\mathcal{F}, \mathcal{F}) \cdot (\overline{Y_1} \cdot \overline{Y_2}) + R(\mathcal{F}, \mathcal{T}) \cdot (\overline{Y_1} \cdot Y_2) \\ & + R(\mathcal{T}, \mathcal{F}) \cdot (Y_1 \cdot \overline{Y_2}) + R(\mathcal{T}, \mathcal{T}) \cdot (Y_1 \cdot Y_2) \end{aligned}$$

$R(Y_1, Y_2)$ take the form of a sum of 2^2 terms (as much as combinations on $\{Y_1, Y_2\}$). Each of these terms (for example: $R(\mathcal{F}, \mathcal{T}) \cdot (\overline{Y_1} \cdot Y_2)$) is the product of two expressions:

- $\overline{Y_1} \cdot Y_2$ is one of the 2^2 combinations on $\{Y_1, Y_2\}$.
- $R(\mathcal{F}, \mathcal{T})$ is the image by function Requirements of the values ($\mathcal{F}$ or $\mathcal{T}$) of the same combination on $\{Y_1, Y_2\}$.

$$R(\mathcal{F}, \mathcal{T}) \cdot (\overline{Y_1} \cdot Y_2)$$

$Y_2 = \mathcal{T}$

$Y_1 = \mathcal{F}$

For n unknowns, the general form of $R(Y_1, ..., Y_n) = \mathcal{F}$ is:

$$\begin{aligned} R(Y_1, Y_2, \ldots, Y_n) = & R(\mathcal{F}, \mathcal{F}, \ldots, \mathcal{F}) \cdot (\overline{Y_1} \cdot \overline{Y_2} \cdot \ldots \cdot \overline{Y_n}) \\ & + \ldots + R(\mathcal{T}, \mathcal{T}, \ldots, \mathcal{T}) \cdot (Y_1 \cdot Y_2 \cdot \ldots \cdot Y_n) = \mathcal{F} \end{aligned} \tag{9}$$

Complementary notations must be introduced in order to express solutions of (9). Let Ω_n be the set of the 2^n n-tuples whose components are $\mathcal{F}$ or $\mathcal{T}$. Let ω_j be an element of this set and ω_j^l the lth component of n-tuple ω_j.

$$R(\omega_j) = R(\underbrace{\mathcal{F}}_{\omega_j^1}, \mathcal{T}, \ldots, \underbrace{\mathcal{T}}_{\omega_j^l}, \ldots, \mathcal{F}, \underbrace{\mathcal{T}}_{\omega_j^n})$$

With this notation, the generic notation of $R(Y_1, Y_2, \ldots, Y_n)$ becomes:

$$R(Y_1, ..., Y_n) = \sum_{j=0}^{2^n-1} \left(R(\omega_j) \cdot \prod_{l=1}^{n} (\omega_j^l \cdot Y_l + \overline{\omega_j^l} \cdot \overline{Y_l}) \right) \tag{10}$$

Taking into account these notations, the fundamental theorem, which gives the set of the solutions of an equation with n unknowns, is:

Theorem 2. $R(Y_1, Y_2, \ldots, Y_n) = \mathcal{F}$ has solutions iff:

$$\prod_{j=0}^{2^n-1} R(\omega_j) = \mathcal{F} \tag{11}$$

In this case, solutions are:

$$\begin{cases} Y_1 = & \dfrac{\prod_{j=0}^{2^{n-1}-1} R(\mathcal{F}, \omega_j^1, \ldots \omega_j^{n-1})}{+ \prod_{j=0}^{2^{n-1}-1} R(\mathcal{T}, \omega_j^1, \ldots \omega_j^{n-1})} \cdot P_1 \\ \ldots & \\ Y_i = & \dfrac{\prod_{j=0}^{2^{n-i}-1} R(Y_1, \ldots, Y_{i-1}, \mathcal{F}, \omega_j^1, \ldots \omega_j^{n-i})}{+ \prod_{j=0}^{2^{n-i}-1} R(Y_1, \ldots, Y_{i-1}, \mathcal{T}, \omega_j^1, \ldots \omega_j^{n-i})} \cdot P_i \\ \ldots & \\ Y_n = & R(Y_1, \ldots, Y_{n-1}, \mathcal{F}) + \overline{R(Y_1, \ldots, Y_{n-1}, \mathcal{T})} \cdot P_n \end{cases}$$

where $P_1, \ldots, P_n$ are constant terms of Γ_m whose set of values describes the whole space of solutions.

The solution of an equation between n BFs could be obtained with theorem 1 by solving n equations with only one unknown. The main interest of theorem 2 is the generic form of the result. Thanks to theorem 2, the existence of solution for (9) can be verified with only one symbolic calculus (11). A parametric expression of the solutions could also be obtained directly.

The proposed formal framework permits to express relations between BFs. These relations can be composed together and can be represented by only one equation. By applying theorem 2, the solutions of this equation can be algebraically calculated. In the next section, we show how this mathematical framework was used to obtain the control laws of a logical system.

V. Processing of the Example

Our synthesis method is being applied to the calculus of the six transition conditions of the SFC given in figure 3.

The first step is to formalize all the data of the problem into BFs, i.e. the functional requirements and the parametric solution given by the designer (the SFC). For that purpose only inclusion or equality relations between BFs can be used.

First, the functional requirements of the water supply can be formalized as follows:

- The two pumps never operate at the same time.

$$(R1): \quad Pump1 \cdot Pump2 = \mathcal{F}$$

- The request order is necessary for the supply.

$$(R2): \quad Pump1 + Pump2 \leq Req$$

- Pump 1 cannot operate if it is out of order.

$$(R3): \quad Pump1 \cdot P1f = \mathcal{F}$$

- Pump 2 cannot operate if it is out of order.

$$(R4): \quad Pump2 \cdot P2f = \mathcal{F}$$

- The pumps can not operate if there is a global failure.

$$(R5): \quad (Pump1 + Pump2) \cdot Gf = \mathcal{F}$$

- If there is no failure, it is sufficient that request is true for one of the pumps operate.

$$(R6): \quad Req \cdot \overline{Gf} \cdot \overline{P1f \cdot P2f} \leq Pump1 + Pump2$$

- Pump permutation is forbidden in the absence of failure.

$$(R7): \quad \overline{P1f} \cdot \overline{P2f} \cdot (Tc12 + Tc21) = \mathcal{F}$$

Moreover the SFC, the generic solution proposed by the designer, must also be formalized by BFs. To do that, it is standard to translate the SFC into a set of recurrent equations as described in [9]. The SFC in figure 3 is also formalized into the BFs algebra by the 8 following equations.

$$\begin{cases} (BS1): & S0(k) = S1(k-1) \cdot Tc10 + S2(k-1) \cdot Tc20 \\ & \quad + S0(k-1) \cdot \overline{Tc01} \cdot \overline{Tc02} \\ (BS2): & S1(k) = S0(k-1) \cdot Tc01 + S2(k-1) \cdot Tc21 \\ & \quad + S1(k-1) \cdot \overline{Tc10} \cdot \overline{Tc12} \\ (BS3): & S2(k) = S0(k-1) \cdot Tc02 + S1(k-1) \cdot Tc12 \\ & \quad + S2(k-1) \cdot \overline{Tc20} \cdot \overline{Tc21} \\ (BS4): & Pump1 = S1 \\ (BS5): & Pump2 = S2 \\ (BS6): & S0(0) = \mathcal{T} \\ (BS7): & S1(0) = \mathcal{F} \\ (BS8): & S2(0) = \mathcal{F} \end{cases}$$

Equations $BS1$, $BS2$ and $BS3$ translate the structure and the elementary evolution rules of the SFC. $Si(k)$ is a binary variable that represents the state of step Si. $BS4$ and $BS5$ express the association of the outputs to the steps. $BS6$, $BS7$ and $BS8$ fix the initialisation of the SFC (activity of each step).

Two specific rules of SFC must also be formalized. The first one is that the transition conditions of an OR divergence must be exclusive:

$$\begin{cases} (GWR1): Tc01 \cdot Tc02 = \mathcal{F} \\ (GWR2): Tc10 \cdot Tc12 = \mathcal{F} \\ (GWR3): Tc21 \cdot Tc20 = \mathcal{F} \end{cases}$$

The second rule is that the model must be stable, i.e. there must be no fugitive evolution (two successive transition conditions are exclusive):

$$\begin{cases} (GWR4): (Tc01 + Tc21) \cdot (Tc10 + Tc12) = \mathcal{F} \\ (GWR5): (Tc02 + Tc12) \cdot (Tc20 + Tc21) = \mathcal{F} \\ (GWR6): (Tc10 + Tc20) \cdot (Tc01 + Tc02) = \mathcal{F} \end{cases}$$

Finally, the system of equations to solve is composed of 21 equations. There are 7 equations that express the expected behaviour of the water supply system ($R1$ to $R7$), 8 equations translating the SFC ($BS1$ to $BS8$) and 6 equations that are needed for expressing specific rules of SFC ($GWR1$ to $GWR6$). The unknowns are the 6 transition conditions ($Tc01$, $Tc02$, $Tc10$, $Tc12$, $Tc20$ and $Tc21$). They are BFs of inputs and state variables.

This system of equations is equivalent to only one equation under the form of (5). To find this unique equation, every relation is rewritten as an equality whose the second member is $\mathcal{F}$. Some equations do not need any rewriting as $(R1)$, ... The inclusions as $(R2)$, ... are rewritten by using (2) and the equalities as $(BS1)$, ... are rewritten by using (3). Finally by using (4), all these equations are rewritten as only one equation (which is too long to be entirely written here):

$$Pump1 \cdot Pump2 + (Pump1 + Pump2) \cdot \overline{Req} \ldots = \mathcal{F} \quad (12)$$

Equation (12) can also be rewritten under the form of (10):

$$\begin{aligned} &R(\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F}) \cdot \overline{Tc01} \cdot \overline{Tc02} \cdot \overline{Tc10} \cdot \overline{Tc12} \cdot \overline{Tc20} \cdot \overline{Tc21} + \ldots \\ &+ R(\mathcal{T},\mathcal{T},\mathcal{T},\mathcal{T},\mathcal{T},\mathcal{T}) \cdot Tc01 \cdot Tc02 \cdot Tc10 \cdot Tc12 \cdot Tc20 \cdot Tc21 = \mathcal{F} \end{aligned}$$

with for example:

$$\begin{aligned} R(\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F},\mathcal{F}) = &(S1(k-1) + S2(k-1)) \cdot (\overline{Req} + Gf) \\ &+ S1(k-1) \cdot P1f + S2(k-1) \cdot P2f \\ &+ \overline{S1(k-1)} \cdot \overline{S2(k-1)} \cdot Req \cdot \overline{Gf} \cdot (\overline{P1f} + \overline{P2f}) \end{aligned}$$

The consistency of the set of equations is first verified. In the case of the water supply system, the problem to solve admits solutions. Then the set of solutions is calculated and given under a parametric form:

$$\begin{cases} Tc01 = Req \cdot \overline{Gf} \cdot \overline{P1f} \cdot (P2f + P_{Tc01}) \\ Tc02 = Req \cdot \overline{Gf} \cdot \overline{P2f} \cdot (P1f + \overline{P_{Tc01}}) \\ Tc10 = \overline{Req} + Gf + P1f \cdot P2f \\ Tc12 = Req \cdot \overline{Gf} \cdot P1f \cdot \overline{P2f} \\ Tc20 = \overline{Req} + Gf + P1f \cdot P2f \\ Tc21 = Req \cdot \overline{Gf} \cdot P2f \cdot \overline{P1f} \end{cases}$$

where P_{Tc01} is a parameter, element of Γ_m, whose set of values describes the whole space of solutions.

The last step of our approach is the choice of one specific solution. $Tc01$ and $Tc02$ are the only two transition conditions dependent on a parameter. They describe the set of possibilities to activate one of the pumps in the nominal mode (when a request of water is emitted in absence of failure). Several strategies are possible. The first solution is to give the priority to a given pump. If Pump 1 has priority over Pump 2 (respect. Pump 2 has priority over Pump 1), $Tc01$ must be maximised by fixing $P_{Tc01} = \mathcal{T}$ (respect. $P_{Tc01} = \mathcal{F}$).

Another objective can be to equilibrate the working time of the two pumps. In this case, a new input can be added (for example "c", as a clock; every 12 hours, c changes value). When $c = 1$ pump 1 has to be used in the nominal mode (pump2 must be used if $c = 0$). In this case it is sufficient to choose $P_{Tc01} = c$ to translate this policy. The solution is:

$$\begin{cases} Tc01 = Req \cdot \overline{Gf} \cdot \overline{P1f} \cdot (P2f + c) \\ Tc02 = Req \cdot \overline{Gf} \cdot \overline{P2f} \cdot (P1f + \overline{c}) \\ Tc10 = \overline{Req} + Gf + P1f \cdot P2f \\ Tc12 = Req \cdot \overline{Gf} \cdot P1f \cdot \overline{P2f} \\ Tc20 = \overline{Req} + Gf + P1f \cdot P2f \\ Tc21 = Req \cdot \overline{Gf} \cdot P2f \cdot \overline{P1f} \end{cases}$$

VI. Conclusion

The synthesis method presented has been developed to design logic controllers. In this paper we have shown how this approach can be used to solve the specific case where the know-how of the designer enables him to a priori build the global form of the solution (by a parametric state model). The specific case of SFC was chosen for an illustrative example but the control law could be expressed by using any other interpreted state model (automaton, interpreted Petri net, state charts, ...). In fact, any control problem of a logical system that can be expressed as a system of equations on the algebra of Boolean functions can be solved by using Theorem 2.

Nevertheless, two types of problems remain to be solved. The first one is connected to the complexity of the calculus to solve the system of equations. We have developed an experimental framework under Mathematica tool that enables us to process study cases. These experiments show that the complexity of the symbolic calculus requires the development of a specific tool. Moreover, the set of solutions of a given system of equations is expressed in a parametric way. We now have to develop optimization techniques that will enable us to find the best solution, modulo an optimization criterion.

The second problem is connected to the difficulties to translate the functional requirements, safety rules, switching mode conditions, ... into algebraic equations. We already have several contributions which will be developed in further research.

References

[1] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] J.-M. Roussel and A. Guia, "Designing dependable logic controllers using the supervisory control theory," 16th *IFAC World Congress, CDROM paper n°04427, Praha (CZ)*, Jul 2005.

[3] Y. Hietter, J.-M. Roussel, and J.-J. Lesage, "Algebraic synthesis of dependable logic controllers," 17th *IFAC World Congress*, 2008.

[4] R. Grimaldi, *Discrete and Combinatorial Mathematics: An Applied Introduction*, 5th ed. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2004, iSBN : 0-321-21103-0, 980 pages.

[5] C. Shannon, "A symbolic analysis of relay and switching circuits," Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, 1940.

[6] B. Bernstein, "Note on the Condition that a Boolean Equation Have a Unique Solution," *American Journal of Mathematics*, vol. 54, no. 2, pp. 417–418, 1932.

[7] R. Toms, "Systems of Boolean Equations," *The American Mathematical Monthly*, vol. 73, no. 1, pp. 29–35, 1966.

[8] V. Levchenkov, "Boolean equations with many unknowns," *Computational Mathematics and Modeling*, vol. 11, no. 2, pp. 143–153, 2000.

[9] J. Machado, B. Denis, J.-J. Lesage, J.-M. Faure, and J. Ferreira Da Silva, "Logic controllers dependability verification using a plant model," 3rd *IFAC Workshop on Discrete-Event System Design: DESDes'06, Rydzyna (Poland)*, pp. 37–42, Sep 2006.