Algebraic approach for dependable logic control systems design

Jean-Marc Roussel, Jean-Marc Faure, Jean-Jacques Lesage, Antonio Medina

An algebraic approach for dependable logic control systems design

J.M. ROUSSEL[(1)*], J.M. FAURE[(1)(2)], J.J. LESAGE[(1)] and A. MEDINA[(1)(3)]

(1) LURPA, ENS de CACHAN, 61 avenue du président Wilson, F-94235 CACHAN Cedex, France.
(2) SUPMECA, 3 rue Fernand Hainaut, F-93407 Saint-Ouen Cedex, France.
(3) Pre-doctoral fellow of CONACYT, Mexico.

* To whom correspondence should be addressed. e-mail: jean-marc.roussel@lurpa.ens-cachan.fr

This article presents a formal method that enables to design a logic control system from specifications given in natural language. The aim of the proposed method is to prevent designer's faults coming from specifications misinterpretation. A significant part of the article is devoted to the presentation of the formal framework that underlies this formal design method: the algebra II. The operations and relations on this algebra allow to state formally specifications of a logic control system and to detect inconsistencies in a specifications set. This inconsistency problem is solved by introducing priority levels. From the consistent set of specifications obtained, control laws can be generated by using theorems and properties of the algebra II. An application of this formal design method to an industrial example permits to illustrate its main advantages.

1 Introduction

The fast-paced development of information and communication technologies has led to introducing an increasing number of automatic operations within everyday objects as well as within production systems for goods and services. The role of these control systems is not limited to replacing the operator during the execution of basic tasks but instead extends to include operator assistance in the completion of complicated tasks, such as the detection of potentially-hazardous situations, failures or deterioration and the safeguarding of an installation under acceptable security conditions. A broad array of examples of such systems are encountered in the fields of manufacturing systems, transport, production and energy distribution, even for functions pertaining to the safety of both people and goods. Whole components of our daily lives and of the economy at large thereby rely upon the successful operations of control systems.

This evolution, related to the increasing demand on the part of society to better control technological risks, explains the significant development efforts devoted to methods that enable guaranteeing, as of the design phase, that the control system meets all requirements imposed by the application. A control system can in fact only be qualified as dependable if no flaw due to a misinterpretation of its specifications has been introduced during design. By adopting the taxonomy proposed in (Laprie 1992), these methods have thus staked a position in the removal of faults caused by the control system designer.

Only the control of discrete event systems (DES), which represent a sizable share of all industrial systems (all the more so given the focus on safety functions), will be considered herein. Under such conditions, two main categories of methods for fault removal during design have been identified (Faure and Lesage 2001):

- A posteriori verification of the design result;
- (Semi-)automatic design.

The first approach consists of letting the control system designer develop control laws based on the requirements contained in the set of specifications and then automatically analyse a formal representation of these control laws. Such an analysis relies on formal techniques that

are either analysis techniques for state automata (i.e. model-checking techniques (Bérard and al. 1999)) or symbolic calculus techniques (i.e. theorem-proving techniques (Roussel and Denis 2002, Roussel and Faure 2002)).

The second approach, qualified as synthesis by some authors (Ramadge and Wonham 1989, Zaytoon and Carré-Ménétrier 2001, Gouyon and al. 2004), is intended to directly deduce the control laws from the specifications, without any involvement of a designer (or at least in limiting involvement to a strict minimum). This approach necessitates, in exchange, a formal modelling set-up as regards both the specifications and the rules for manipulating the representative formal models.

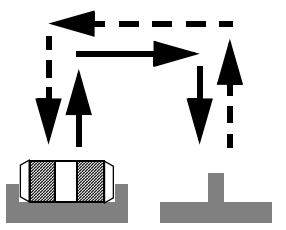
The work presented in this article lies within this latter category and is aimed at contributing to fault removal during the design of a control system for a logic DES, by means of proposing a method that enables deducing from specifications expressed using natural language a complete and consistent formal description of the control laws. This formal model may then be easily implemented in either hardware or software form.

The following section is devoted to presenting this method's main objective and serves to introduce the formalism used to ground the methodology. This formalism will then be discussed in the third section; the various operations and relations admitted that provide a formal basis for the design method will be explored therein as well. This construction serves in the fourth section to describe the proposed design method for a dependable control system. The fifth and final section illustrates a simple example using this method.

2 Objective of the present research work

The starting point for the proposed design method is the set of specifications inherent in the control system, as expressed with natural language. These specifications describe the expected behaviour of the control system, in the form of vivacity constraints (what the control system must accomplish) and safety constraints (what the system must not accomplish), and may include constraints coming from actuator and sensor technology choices. All of these constraints are to be expressed in the form of logic assertions, i.e. propositions that must be true for the desired control system.

As an example, let's consider the pneumatic manipulator represented in figure 1. Its purpose is to transport mechanical parts from the picking station to the placing station. Due to the presence of obstacles between these two stations, horizontal movements can only be performed when the manipulator gripper is in the raised position. The desired movement is thus that indicated in figure 1 («U-shaped» cycle). The following technological choices have been carried out:



Horizontal movement:
- Double-acting cylinder driven by a bistable valve
- Two sensors (rightmost and leftmost positions)

Vertical movement:
- Double-acting cylinder driven by a monostable valve
- Two sensors (raised and low positions)

Manipulator gripper:
- Drawing up system using a Venturi device and a monostable valve
- No vacuum sensor

Figure 1. Manipulator to control

The control specifications to be used herein must therefore comprise the following assertions:

- The manipulator gripper may only move horizontally while in the raised position.
- The manipulator gripper may only drop down at the picking station or at the placing station.
- When the manipulator gripper is in the raised position at the picking station, pressing the «Start» button causes the gripper to drop down.
- The product is considered to be seized if the drawing up system is triggered within a second of contact (low position at the picking station).
- For a double-acting cylinder driven by a bistable valve, the movement controls must not be simultaneous.
- ...

The complete list of assertions necessary to design the control for this manipulator will be provided in section 5.1. At this point however, it is important to highlight the following:

- As shown by the narrower previous set of assertions, the control specifications of a logic system can make reference to logic variable states, to state changes in these logic variables (events), or to physical time values. The formalism that supports the design method is to be endowed with the capability of expressing these three types of variables.
- A set of specifications does not necessarily have to be consistent. The design method must therefore be capable of detecting possible inconsistencies in the specifications and then proposing solutions that enable resolving these problems.

Control laws are to be designed on the basis of this list of assertions. For a dynamic logic control system with n boolean input variables $U_1(t)$ to $U_n(t)$ and m boolean output variables $Y_1(t)$ to $Y_m(t)$, the target control laws must specify at each time t the output values as functions of input values, which leads to (Cassandras, 1999):

$$\begin{cases} Y_1(t) = f_1(U_1(t), \ldots, U_n(t)) \\ \quad\quad \ldots \\ Y_m(t) = f_m(U_1(t), \ldots, U_n(t)) \end{cases}$$

The search for solutions to this system of m equations generally requires a reformulation of the problem in the form of a state model, thereby necessitating the introduction of other variables, state variables X_j recording the system evolution. Regardless of the advantages inherent in this approach, it displays the disadvantage of merely providing specific solutions, at a given point in time, as exemplified in the form of «Y_i becomes true when U_k becomes false and if variables X_2, X_5 and X_8 are false while variables X_3 and X_{10} true», but never a general solution that holds true regardless of the date considered. Moreover, modelling a state automaton-based dynamic system corresponds to an imperative design approach, whereas control system specifications are in most cases given in declarative form. The transition from one of these representation modes to the other always requires a major effort.

For both of these reasons, the proposed design method relies upon a special formalism that has been developed by our research team. The basic elements of this formalism consist of the time functions $U_i(t)$, $Y_i(t)$. The operations that enable composing these functions lead to defining an algebraic structure in which the simultaneous manipulation of boolean variable states, state changes of these variables (events) and physical time values is possible. As a consequence,

this algebra has been denoted algebra II since it yields a framework for integrating the three types of variables; the next section presents a detailed examination of this algebra.

In sum, the proposed design method for dependable logic control systems calls for developing, on the basis of specifications given in the form of assertions in natural language, a set of control laws in the form of time functions. This method requires (see figure 2) formalising specifications into relations within the algebra II as well as checking the consistency of the set of assertions and generating the expected control laws from the consistent set of specifications obtained.

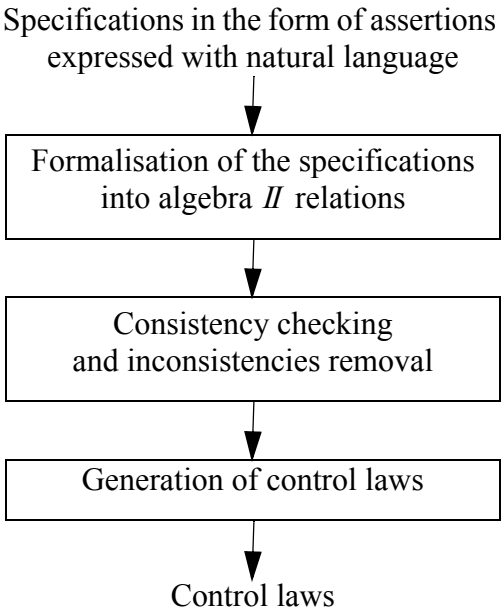


Figure 2. Method overview

3 Formal framework: a boolean algebra for binary signals

3.1 Binary signal modelling

As mentioned in the previous section, the algebra II shall provide a formal framework to represent and to manipulate boolean variable states, events and physical delays. When defining this algebra, the main idea was to consider binary signals, i.e. functions describing the evolutions of boolean values in time.

These evolutions are usually represented by timing diagrams. Though this representation is quite useful for control engineers, it is not at all based on a sound formalism. Hence the first step in the definition of the algebra II is aimed at giving a formal definition of binary signals: piecewise-continuous functions from IR^{+*} to $\mathbb{B} = \{0, 1\}$. The elements of II are consequently formally defined in the following way:

$$\mathit{II} = \{f : IR^{+*} \to IB \mid \forall t \in IR^{+*} : (\exists \varepsilon_t > 0 : (\forall (\varepsilon_1, \varepsilon_2) \in (0, \varepsilon_t)^2, f(t - \varepsilon_1) = f(t - \varepsilon_2)))\}$$

The figure 3 shows an example of a function element of II. Attention shall be paid to the right-continuity used at the dates t_1 and t_3 and to the double-discontinuity at the dates t_2 and t_4, mandatory to model events.

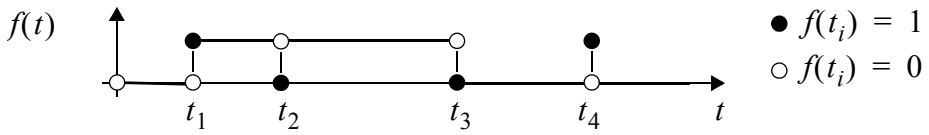


Figure 3. Graphical representation of a binary signal

To distinguish operations on elements of II from operations on booleans, different notations are used. « $\wedge$ » will denote the logical AND operation between two booleans, « $\vee$ » the logical OR operation between two booleans, «$\neg$» the NOT operation on a boolean. The notations « $\cdot$ », « $+$ » and « $\bar{}$ » will be used for operations on II. Furthermore, differents notations will be used for functions and for values of functions at a given time t. For instance, f, g, h will denote three functions, elements of II, and $f(t), g(t), h(t)$ three booleans.

II contains two special elements 1^* (the one element) and 0^* (the zero element) defined as follows:

$$1^* : \quad IR^{+*} \to IB, \quad 1^*(t) \to 1 \qquad\qquad 0^* : \quad IR^{+*} \to IB, \quad 0^*(t) \to 0$$

3.2 Structure of boolean algebra

To compose the elements of II, three closed operations have been defined:

AND operation	OR operation	NOT operation
$\mathit{II}^2 \to \mathit{II}$	$\mathit{II}^2 \to \mathit{II}$	$\mathit{II} \to \mathit{II}$
$(f, g) \to (f \cdot g)$	$(f, g) \to (f + g)$	$f \to \bar{f}$

Where $\forall t \in IR^{+*}$,

$$(f \cdot g)(t) = f(t) \wedge g(t) \qquad (f+g)(t) = f(t) \vee g(t) \qquad \bar{f}(t) = \neg f(t)$$

$(\mathit{II}, ., +, \bar{}, 1^*, 0^*)$ is a boolean algebra (Grimaldi 2000) because the following conditions are satisfied for all $f, g, h \in \mathit{II}$

$f \cdot g = g \cdot f$	$f + g = g + f$	Commutative laws
$f \cdot (g + h) = (f \cdot g) + (f \cdot h)$	$f + (g \cdot h) = (f + g) \cdot (f + h)$	Distributive laws
$f \cdot 1^* = f$	$f + 0^* = f$	Identity laws
$f \cdot \bar{f} = 0^*$	$f + \bar{f} = 1^*$	Inverse laws

$$0^* \neq 1^*$$

As $(\mathbb{B}, ., +, \bar{}, 1^*, 0^*)$ is a boolean algebra, the following statements hold:

$f \cdot f = f$	$f + f = f$	Idempotent laws
$f \cdot 0^* = 0^*$	$f + 1^* = 1^*$	Dominance laws
$f \cdot (f + g) = f$	$f + (f \cdot g) = f$	Absorption laws
$f \cdot (g \cdot h) = (f \cdot g) \cdot h$	$f + (g + h) = (f + g) + h$	Associative laws
$\bar{\bar{f}} = f$		Law of the double complement
$\overline{(f \cdot g)} = \bar{f} + \bar{g}$	$\overline{(f + g)} = \bar{f} \cdot \bar{g}$	De Morgan's laws

The three basis operations (AND, OR, NOT) enable to combine binary signals only in a combinatory way, i.e. to obtain a signal whose value at each date is obtained from the values of the operands at the same time. Sequential and timed operations are to be defined to describe more complex behaviours, such as those included in the specifications of industrial control systems.

3.3 Sequential operations

Two binary operations, denoted $SR(s, r)$ and $RS(s, r)$, have been therefore defined. The formal statements of these operations are:

SR operation	RS operation
$\mathbb{B}^2 \to \mathbb{B}$	$\mathbb{B}^2 \to \mathbb{B}$
$(s, r) \to SR(s, r)$	$(s, r) \to RS(s, r)$

Where $\forall t \in IR^{+*}$,

$$SR(s, r)(t) = s(t) \vee [\exists t_1 < t \,|\, ((s(t_1) = 1) \wedge (\forall d \in (t_1, t], (r(d) = 0)))]$$

$$RS(s, r)(t) = (s(t) \wedge \neg r(t)) \vee [\exists t_1 < t \,|\, ((s(t_1) = 1) \wedge (\forall d \in [t_1, t], (r(d) = 0)))]$$

The figure 4 shows the graphical representation of two binary signals s and r as well as the results of SR and RS operations on these signals. It matters to highlight that SR describes the behaviour of a set dominant memory, while RS describes the behaviour of a reset dominant memory.

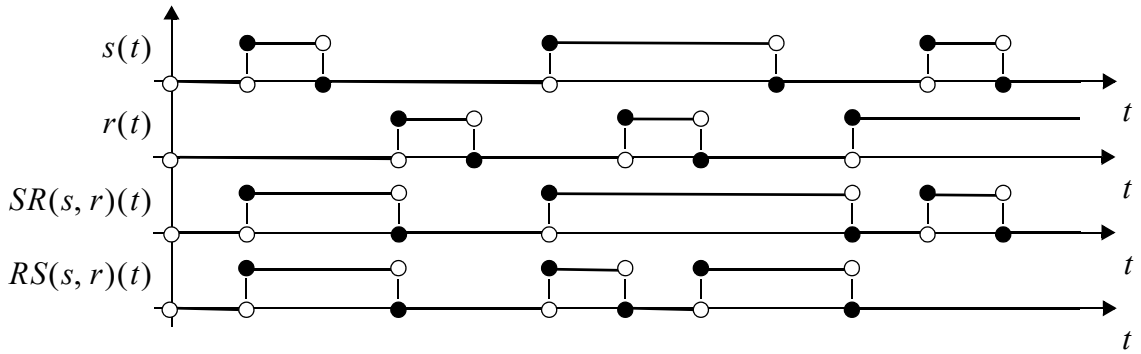


Figure 4. Graphical representation of $SR(s, r)$ and $RS(s, r)$ functions

The value of the function $SR(s, r)$ (respectively $RS(s, r)$) is thus determined at each instant t as the logical OR between two booleans. The first boolean is the value of the function s (respectively $s \cdot \bar{r}$) at t. The second boolean is the value of a predicate at the same date. The truthfulness of this predicate depends on the existence of a former date t_1, such as $s(t_1)$ (respectively $(s \cdot \bar{r})(t_1)$) was 1 and since which the value of the r function has remained always equal to 0.

With these definitions, the following laws have been established:

$$SR(s, r_1 + r_2) = SR(s, r_1) \cdot SR(s, r_2)$$
$$RS(s, r_1 + r_2) = RS(s, r_1) \cdot RS(s, r_2)$$
$$SR(s_1 + s_2, r) = SR(s_1, r) + SR(s_2, r)$$
$$RS(s_1 + s_2, r) = RS(s_1, r) + RS(s_2, r)$$

$$RS(s, r) = SR(s \cdot \bar{r}, r)$$
$$SR(s, r) = SR(s, r) + s$$
$$RS(s, r) = RS(s + r \cdot f, r)$$
$$SR(s, r) \cdot r = s \cdot r$$
$$RS(s, r) \cdot s = s \cdot \bar{r}$$

$$SR(s, \bar{s}) = s$$
$$RS(s, \bar{s}) = s$$
$$RS(s, s) = 0^*$$
$$r \cdot RS(s, r) = 0^*$$

$$SR(s, 1^*) = s$$
$$SR(1^*, r) = 1^*$$
$$SR(0^*, r) = 0^*$$

$$RS(s, 1^*) = 0^*$$
$$SR(1^*, r) = \bar{r}$$
$$RS(0^*, r) = 0^*$$

3.4 Timed operations

To state delayed signals in a formal way, two unary operations, denoted TON and TOF, have been defined. The formal statements of these operations are:

TON operation	TOF operation
$II \rightarrow II$	$II \rightarrow II$
$f \rightarrow d/f$	$f \rightarrow f/d$

Where $\forall t \in IR^{+*}$,

$$(d/f)(t) = \begin{cases} 0 & \forall t < d \\ (\forall t_1 \in (t-d, t], (f(t_1) = 1)) & \forall t \geq d \end{cases}$$

$$(f/d)(t) = \begin{cases} (\exists t_1 \in (0, t], (f(t_1) = 1)) & \forall t < d \\ (\exists t_1 \in (t-d, t], (f(t_1) = 1)) & \forall t \geq d \end{cases}$$

The figure 5 shows the graphical representation of a binary signal f and of the results of TON and TOF operations on this signal. TON behaves as a «ON -delay Timer». TOF behaves as a «OFF -delay Timer».

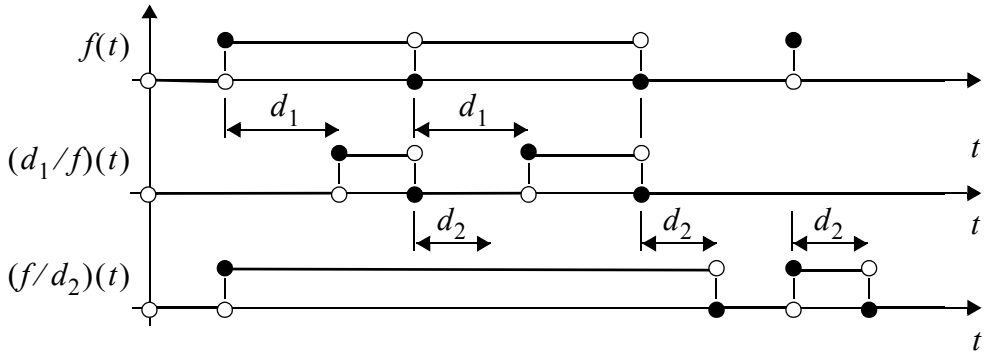


Figure 5. Graphical representation of the results of TON and TOF operations

The TON and TOF operations transform a function f into two new functions d/f and f/d. For each date t, the value of these new functions depends on the value of a predicate that checks the value of the f function on a period of time $(t-d, t]$.

With these definitions, the following laws have been established:

$$f = f + d/f \qquad f/d = f + f/d$$

$$d/(f \cdot g) = (d/f) \cdot (d/g) \qquad (f+g)/d = f/d + g/d$$

$$(d_1/f) \cdot (d_2/f) = max(d_1, d_2)/f \qquad (f/d_1) + (f/d_2) = f/max(d_1, d_2)$$

$$(d_1/f) + (d_2/f) = min(d_1, d_2)/f \qquad (f/d_1) \cdot (f/d_2) = f/min(d_1, d_2)$$

$$d_1/(d_2/f) = sum(d_1, d_2)/f \qquad (f/d_1)/d_2 = f/sum(d_1, d_2)$$

$$\forall t \geq d \qquad \overline{(d/f)} = \bar{f}/d \qquad \forall t \geq d \qquad \overline{(f/d)} = d/\bar{f}$$

3.5 Events modelling

An event related to a given signal f must be true only when the value of this signal changes (see figure 6).

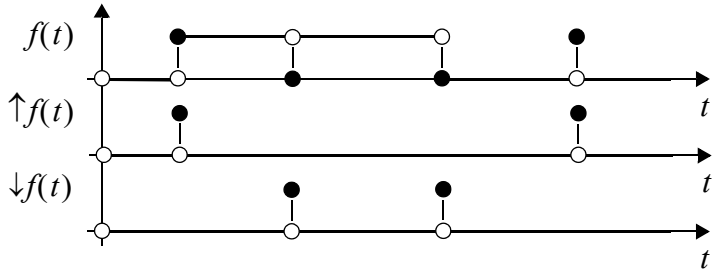


Figure 6. Graphical representation of the results of RE and FE operations

To state formally that kind of signal, two unary operations: Rising Edge (RE), and Falling Edge (FE), must be defined as follows:

RE operation	FE operation
$II \rightarrow II$	$II \rightarrow II$
$f \rightarrow \uparrow f$	$f \rightarrow \downarrow f$

Where $\forall t \in IR^{+*}$,

$$\uparrow f(t) = f(t) \wedge (\exists \varepsilon_0 > 0 : \forall \varepsilon \in (0, \varepsilon_0), f(t-\varepsilon) = 0)$$

$$\downarrow f(t) = \neg f(t) \wedge (\exists \varepsilon_0 > 0 : \forall \varepsilon \in (0, \varepsilon_0), f(t-\varepsilon) = 1)$$

The RE and FE operations transform a function f into two new functions $\uparrow f$ and $\downarrow f$. For each date t, the value of these new functions depends on the value of f at the date t and the value of a predicate that checks the value of the function f on a period of time $(t-\varepsilon_0, t)$.

With these definitions, the following laws have been established:

$$(\uparrow f) + f = f$$

$$(\uparrow f) \cdot f = \uparrow f$$

$$\uparrow \bar{f} = \downarrow f$$

$$(\downarrow f) + \bar{f} = \bar{f}$$

$$(\downarrow f) \cdot \bar{f} = \downarrow f$$

$$\downarrow \bar{f} = \uparrow f$$

$$\uparrow(\uparrow f) = \uparrow f$$

$$\uparrow(\downarrow f) = \downarrow f$$

$$\uparrow\left(\prod_{i \in \{1,n\}} f_i\right) = \sum_{i \in \{1,n\}} \left(\uparrow f_i \cdot \prod_{\substack{j \in \{1,n\} \\ j \neq i}} f_j\right)$$

$$\uparrow\left(\sum_{i \in \{1,n\}} f_i\right) = \sum_{i \in \{1,n\}} \left(\uparrow f_i \cdot \prod_{\substack{j \in \{1,n\} \\ j \neq i}} (\uparrow f_j + (\bar{f}_j \cdot \overline{\downarrow f_j}))\right)$$

$$\downarrow\left(\prod_{i \in \{1,n\}} f_i\right) = \sum_{i \in \{1,n\}} \left(\downarrow f_i \cdot \prod_{\substack{j \in \{1,n\} \\ j \neq i}} (\downarrow f_j + (f_j \cdot \overline{\uparrow f_j}))\right)$$

$$\downarrow\left(\sum_{i \in \{1,n\}} f_i\right) = \sum_{i \in \{1,n\}} \left(\downarrow f_i \cdot \prod_{\substack{j \in \{1,n\} \\ j \neq i}} \bar{f}_j\right)$$

3.6 Equality and partial ordering relations

An obvious relation in the algebra II is the equality between two signals, denoted =, which states that the values of these signals are equal, whatever the considered date.

Moreover as $(\mathit{II}, ., +, \bar{}, 1^*, 0^*)$ is a boolean algebra, the relation $\leq$ defined as follows is a partial ordering relation (Grimaldi, 2000).

$$\text{If } (f, g) \in \mathit{II}, \text{ define } f \leq g, \text{ if } f \cdot g = f$$

As the relation $\leq$ is a partial ordering relation, this relation is reflexive, antisymmetric and transitive. This relation can be also stated in the following way:

$$\forall t \in IR^{+*}, f(t) \wedge \neg g(t) = 0 \text{ or } \forall t \in IR^{+*}, \neg f(t) \vee g(t) = 1$$

That means in natural language: «For all dates t_i such as $f(t_i)$ is true, the value of $g(t_i)$ is true too».

For all f, g elements of II, the 6 following relations have been proved equivalent:

$$f \leq g \qquad f \cdot g = f \qquad \bar{f} + g = 1^*$$

$$\bar{g} \leq \bar{f} \qquad f + g = g \qquad f \cdot \bar{g} = 0^*$$

The following results whose usefulness when consistency checking will be shown in section 5.3 have been proved too:

$$(f+g) \le h \quad \equiv \quad \begin{cases} f \le h \\ g \le h \end{cases} \qquad f \le (g \cdot h) \quad \equiv \quad \begin{cases} f \le g \\ f \le h \end{cases}$$

This partial ordering relation is the cornerstone of the design method of dependable control systems as described in the next section.

4 Contribution of the algebra II to the design of dependable control systems

As sketched in the second section of this article, the algebra II is the underlying theory of the developed design method. The objective of the current section is to show how this algebra is employed in the three steps of the design method:

- Formalisation of the specifications into algebra II relations,
- Consistency checking and inconsistencies removal,
- Control laws generation.

The latter two steps will be presented jointly thanks to a simple example: a single output control system.

4.1 Specifications formalisation

The operations and relations of the algebra II enable to state formally specifications given in the form of assertions in natural language and including boolean variable states, events and physical delays. Some usual assertions that can easily be obtained from control systems speci-

fications and the equivalent formal relations are presented in table 1. A larger set of assertions will be given in the fifth section.

Assertions given in natural language	Equivalent formal relations
The values of the f and g signals are always equal.	$f = g$
The values of the f and g signals are never simultaneously true.	$f \cdot g = 0^*$
At each time, at least one of the values of the f and g signals is true.	$f + g = 1^*$
When the value of the signal f is true, the value of the signal g is true.	$f \leq g$
It is sufficient that the value of the signal f is true to get the value of the signal g true.	$f \leq g$
The value of the signal f must be true to obtain the value of the signal g true.	$g \leq f$ or $\bar{f} \leq \bar{g}$
The value of the signal f is never true more than 3 seconds.	$3s/f = 0^*$
When the value of the signal f becomes true, the value of the signal g signal is true.	$(\uparrow f) \leq g$
When the value of the signal f becomes false, the value of the signal g signal is true.	$(\downarrow f) \leq g$

Table 1: Assertions and formal relations

4.2 Generation of a control law from a formal specification

Let us consider an elementary single output control system whose output O is assumed to be specified by the following two relations:

$$\begin{cases} (1a)\ A \leq O \\ (1b)\ B \leq \overline{O} \end{cases}$$

A, B and O are binary signals. A and B can be two inputs of the control system or complex statements using the operations of the algebra II. The searched control law shall relate O to an expression involving A and B. The solution of each of these relations is quite easy to obtain:

- $O = A + f_1$ with $f_1 \in \mathit{II}$, for the first relation (1a). This relation sets indeed that $O(t)$ must be true when $A(t)$ is true, no matter what its value is when $A(t)$ is false.
- $O = \overline{B} \cdot f_2$ with $f_2 \in \mathit{II}$, for the second one (1b). $O(t)$ must be false when $B(t)$ is true, no matter what its value is when $B(t)$ is false.

4.2.1 Relations set analysis

When analysing the set of the two relations, three cases can be pointed out:

- There is no solution if the statement $A \cdot B \neq 0^*$ holds. In that case, there is indeed at least

one date such as $(A \cdot B)(t_i) = 1$, that leads to inconsistency ($O(t_i)$ should be at the same time true and false).

- There is an unique solution: $O = A$, if the statement $A = \overline{B}$ hold.
- There is an infinite number of solutions (the output is not completely specified) if $A \cdot B = 0^*$ and $A \neq \overline{B}$. These solutions can be written in the form: $O = (A + \overline{B} \cdot f_1)$ or $O = (A + f_2) \cdot \overline{B}$ with $f_1 \in \mathit{II}$ and $f_2 \in \mathit{II}$.

The previous analysis showed how inconsistency and incompleteness can be detected thanks to relations of the algebra II. The inconsistency of a set of relations must be removed to generate a control law; this shall be performed by introducing priority levels between relations. Incompleteness does not prevent to generate a control law; all you have to do is to choose a possible solution.

4.2.2 Control laws

To establish a control law, whatever the value of $A \cdot B$, the three situations described in table 2 are to be considered.

Situation # 1: No inconsistency ($A \cdot B = 0^*$) The consistent specification is: $\begin{cases} (1a)\ A \leq O \\ (1b)\ B \leq \overline{O} \\ A \cdot B = 0^* \end{cases}$ The form of the control law is: $O_1 = (A + \overline{B} \cdot f_1)$ or $O_1 = (A + f_2) \cdot \overline{B}$ with $(f_1, f_2) \in \mathit{II}^2$.
Situation # 2: Initial inconsistency ($A \cdot B \neq 0^*$) The relation (1a) is dominant beside the relation (1b), denoted $(1b) \ll (1a)$. The consistent specification is: $\begin{cases} (1a)\ A \leq O \\ (1b)\ B \leq \overline{O} \\ (1b) \ll (1a) \end{cases}$ equivalent to $\begin{cases} (1a)\ A \leq O \\ (1b)\ \overline{A} \cdot B \leq \overline{O} \end{cases}$ The form of the control law is: $O_2 = (A + \overline{B} \cdot f)$ with $f \in \mathit{II}$.
Situation # 3: Initial inconsistency ($A \cdot B \neq 0^*$) The relation (1b) is dominant beside the relation (1a). The consistent specification is: $\begin{cases} (1a)\ A \leq O \\ (1b)\ B \leq \overline{O} \\ (1a) \ll (1b) \end{cases}$ equivalent to $\begin{cases} (1a)\ A \cdot \overline{B} \leq O \\ (1b)\ B \leq \overline{O} \end{cases}$ The form of the control law is: $O_3 = \overline{B} \cdot (A + f)$ with $f \in \mathit{II}$.

Table 2: General solutions

Attention shall be paid that for the dates t_i such as $A(t_i) \vee B(t_i) = 0$, the value $O(t_i)$ of the solution O can be true or false according to the choice made for f, f_1, f_2. In order to obtain deterministic control laws, i.e. control laws involving only the A and B signals and no specific designer's choices, one of the following solutions is to be chosen:

- At these dates t_i, $O(t_i) = 0$, that leads to:

$$O_1 = A \qquad O_2 = A \qquad O_3 = A \cdot \overline{B}$$

- At these dates t_i, $O(t_i) = 1$, that leads to:

$$O_1 = \overline{B} \qquad O_2 = A + \overline{B} \qquad O_3 = \overline{B}$$

- At these dates t_i, $O(t_i)$ keeps the last value determined by the assertions set (memory behaviour), that leads to:

$$O_1 = SR(A, B) \text{ or } O_1 = RS(A, B) \qquad O_2 = SR(A, B) \qquad O_3 = RS(A, B)$$

The last solution will be adopted herein for it seems closer to industrial concerns and practice.

5 Application example

This section is aimed at dealing with a real example coming from an industrial assembly line and depicted in the figure 1. The expected operation mode («U-shaped» cycle) of this example has been sketched previously; the inputs and the outputs of the control system to design are given in the figure 7.

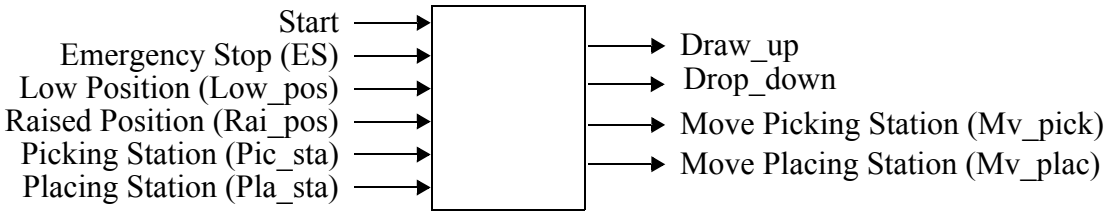


Figure 7. Inputs and outputs of the control system to design

5.1 Control system specifications

The expected behaviour of the control system with regard to the application requirements may be expressed by the set of assertions given hereafter. Among these 16 assertions, the first three ones (A1 to A3) are related to safety requirements, the following ten ones (A4 to A13) to vivacity requirements (what must be done to perform the production task), the assertions A14 à and A15 express constraints coming from actuators features and the last one (A16) is an assumption on the correct operation of the sensors (the problem of sensors monitoring will not be dealt with in this study).

A1 The manipulator gripper may only move horizontally while in the raised position.

A2 The manipulator gripper may only drop down at the picking station or at the placing station.

A3 In case of Emergency Stop, all the movement controls must be reset.

A4 When the manipulator gripper is in the raised position at the picking station, pressing the «Start» button causes the gripper to drop down.

A5 When the manipulator gripper is in the low position at the picking station, the part must be

seized using the drawing up system.

A6 When the part is held at the picking station, the manipulator gripper must move up.

A7 When the part is held at the picking station, the manipulator gripper must move towards the placing station.

A8 When the part is held at the placing station, the manipulator gripper must drop down.

A9 When the part is put down at the placing station, the manipulator gripper must move up.

A10 When the part is put down at the placing station, the manipulator gripper must come back to the picking station.

A11 The drawing up system must be reset in the low position at the placing station.

A12 The part is considered to be held if the drawing up system is triggered within a second of contact (low position at the picking station).

A13 If the drawing up system is reset, the part is no more held.

A14 For a double-acting cylinder driven by a bistable valve, the two movement controls must not be simultaneous.

A15 For a double-acting cylinder driven by a bistable valve, the end of any movement shall reset the control of this movement.

A16 No sensors failure may occur.

5.2 Specifications formalisation

The previous set of assertions can be translated into a set of formal relations (table 3) that include input signals (Start, Emergency Stop (ES), Low Position (Low_pos), Raised Position (Rai_pos), Picking Station (Pic_sta), Placing Station (Pla_sta)), output signals (Draw_up, Drop_down, Move Picking Station (Mv_pick), Move Placing Station (Mv_plac)), and internal signals of the control system (Held_part). The main objective of the design method is to state from this relations set the control laws that link the output signals to the input signals and, if necessary, to internal signals. Equations defining these internal signals are also to be defined.

It matters to highlight that some relations (A1, A2, A3) may be written in several equivalent forms. This feature is quite interesting when checking consistency. It will be possible in-

deed at this step of the design method to chose the most appropriate statement, as explained in the next paragraph.

Assertion	Assertions written on II
A1	$(\text{Mv\_plac} + \text{Mv\_pick}) \leq \text{Rai\_pos}$ or $\begin{cases} \overline{\text{Rai\_pos}} \leq \overline{\text{Mv\_plac}} \\ \overline{\text{Rai\_pos}} \leq \overline{\text{Mv\_pick}} \end{cases}$
A2	$\text{Drop\_down} \leq (\text{Pla\_sta} + \text{Pic\_sta})$ or $\overline{\text{Pla\_sta}} \cdot \overline{\text{Pic\_sta}} \leq \overline{\text{Drop\_down}}$
A3	$\text{ES} \leq \overline{\text{Mv\_plac}} \cdot \overline{\text{Mv\_pick}} \cdot \overline{\text{Drop\_down}}$ or $\begin{cases} \text{ES} \leq \overline{\text{Mv\_plac}} \\ \text{ES} \leq \overline{\text{Mv\_pick}} \\ \text{ES} \leq \overline{\text{Drop\_down}} \end{cases}$
A4	$\text{Rai\_pos} \cdot \text{Pic\_sta} \cdot \uparrow\text{Start} \leq \text{Drop\_down}$
A5	$\text{Low\_pos} \cdot \text{Pic\_sta} \leq \text{Draw\_up}$
A6	$\text{Pic\_sta} \cdot \text{Held\_part} \leq \overline{\text{Drop\_down}}$
A7	$\text{Pic\_sta} \cdot \text{Held\_part} \leq \text{Mv\_plac}$
A8	$\text{Pla\_sta} \cdot \text{Held\_part} \leq \text{Drop\_down}$
A9	$\text{Pla\_sta} \cdot \overline{\text{Held\_part}} \leq \overline{\text{Drop\_down}}$
A10	$\text{Pla\_sta} \cdot \overline{\text{Held\_part}} \leq \text{Mv\_pick}$
A11	$\text{Low\_pos} \cdot \text{Pla\_sta} \leq \overline{\text{Draw\_up}}$
A12	$1\text{s} / (\text{Draw\_up} \cdot \text{Low\_pos} \cdot \text{Pic\_sta}) \leq \text{Held\_part}$
A13	$\overline{\text{Draw\_up}} \leq \overline{\text{Held\_part}}$
A14	$\text{Mv\_pick} \cdot \text{Mv\_plac} = 0^*$
A15	$\begin{cases} \text{Pla\_sta} \leq \overline{\text{Mv\_plac}} \\ \text{Pic\_sta} \leq \overline{\text{Mv\_pick}} \end{cases}$
A16	$\begin{cases} \text{Pic\_sta} \cdot \text{Pla\_sta} = 0^* \\ \text{Rai\_pos} \cdot \text{Low\_pos} = 0^* \end{cases}$

Table 3: Formal specifications of the control system

5.3 Consistency checking

When dealing with a set of formal relations involving several output and internal signals, consistency checking comprises three steps:

- Assertions labelling
- Dependency analysis
- Consistency checking for each output or internal signal (this step has been already described in 4.2)

Consistency of a set of relations depends indeed not only on the consistency of the relations defining a given output signal, as previously shown, but moreover on the lack of cross-references in the relations set, e.g. the output O_i is defined from the internal signal IS_j that is itself defined from O_i. The first two steps are aimed at checking cross-references while the last step looks for inconsistencies in the specification of each signal.

5.3.1 Assertions labelling

This step is aimed at ranking each of the relations of the set of specifications into one of the following categories:

- Relations stating formally assertions that involve only input signals. These relations are assumptions.
- Relations stating formally assertions that involve only one output or internal signal and input signals. Each of this relation will be used to build the control law of the given output or internal signal from input signals.
- Relations stating formally assertions that involve several output or internal signals. In that case, either the relation can be decomposed in elementary relations (relations comprising only one output or internal signal) as stated in section 3.6, that leads to the previous case, or the designer has to decide which signal will be function of the other ones.

From this analysis it is possible to determine the assertions that must be employed to elaborate an internal or output signal and on which other signals the considered signal depends. The result of this analysis for the control system of the pneumatic manipulator is presented in table 4.

Internal or output signal	Assertions to be used to generate the signal	Signals involved in the assertions
Draw_up	A5, A11	Low_pos (A5, A11), Pla_sta (A11), Pic_sta (A5)
Held_part	A12, A13	Low_pos (A12), Pic_sta (A12), Draw_up (A12, A13)
Drop_down	A2, A3c, A4, A6, A8, A9	ES (A3c), Start (A4), Rai_pos (A4), Pla_sta (A2, A8, A9), Pic_sta (A2, A4, A6), Held_part (A6, A8, A9)
Mv_plac	A1a, A3a, A7, A15a	ES (A3a), Rai_pos (A1a), Pla_sta (A15a), Pic_sta (A7), Held_part (A7)
Mv_pick	A1b, A3b, A10 A14, A15b	ES (A3b), Rai_pos (A1b), Pla_sta (A10), Pic_sta (A15b), Held_part (A10), Mv_plac (A14)

Table 4: Dependency relations derived from assertions labelling

5.3.2 Dependency analysis

From the previous results a dependency graph can be easily built. If this graph does not include any cycle, no cross-reference lies within the assertions set. Conversely any cycle found when analysing the dependency graph enables to point out specifications inconsistency. In that case, part of the assertions set must be modified in order to eliminate the inconsistencies.

The table 4 shows that the set of assertions of the studied example does not include any inconsistency.

5.3.3 Consistency checking for each output or internal signal

The principle of this step has been already described in section 4.2. The results of this analysis for this example are given in table 5.

Internal or output signal	Consistent specification
Draw_up	(A5) $\text{Low\_pos} \cdot \text{Pic\_sta} \le \text{Draw\_up}$ (A11) $\text{Low\_pos} \cdot \text{Pla\_sta} \le \overline{\text{Draw\_up}}$ $\text{Pick\_sta} \cdot \text{Pla\_sta} = 0^*$
Held_part	(A12) $1s/(\text{Draw\_up} \cdot \text{Low\_pos} \cdot \text{Pic\_sta}) \le \text{Held\_part}$ (A13) $\overline{\text{Draw\_up}} \le \overline{\text{Held\_part}}$
Drop_down	(A2) $\overline{\text{Pla\_sta}} \cdot \overline{\text{Pic\_sta}} \le \overline{\text{Drop\_down}}$ (A3c) $\text{ES} \le \overline{\text{Drop\_down}}$ (A4) $\text{Rai\_pos} \cdot \text{Pic\_sta} \cdot \uparrow\text{Start} \le \text{Drop\_down}$ (A6) $\text{Pic\_sta} \cdot \text{Held\_part} \le \overline{\text{Drop\_down}}$ (A8) $\text{Pla\_sta} \cdot \text{Held\_part} \le \text{Drop\_down}$ (A9) $\text{Pla\_sta} \cdot \overline{\text{Held\_part}} \le \overline{\text{Drop\_down}}$ $\text{Pic\_sta} \cdot \text{Pla\_sta} = 0^*$ (A4) « (A3c) (A8) « (A3c) (A4) « (A6)
Mv_plac	(A1a) $\overline{\text{Rai\_pos}} \le \overline{\text{Mv\_plac}}$ (A3a) $\text{ES} \le \overline{\text{Mv\_plac}}$ (A7) $\text{Pic\_sta} \cdot \text{Held\_part} \le \text{Mv\_plac}$ (A15a) $\text{Pla\_sta} \le \overline{\text{Mv\_plac}}$ $\text{Pic\_sta} \cdot \text{Pla\_sta} = 0^*$ (A7) « (A1a) (A7) « (A3a)
Mv_pick	(A1b) $\overline{\text{Rai\_pos}} \le \overline{\text{Mv\_pick}}$ (A3b) $\text{ES} \le \overline{\text{Mv\_pick}}$ (A10) $\text{Pla\_sta} \cdot \overline{\text{Held\_part}} \le \text{Mv\_pick}$ (A14) $\text{Mv\_plac} \le \overline{\text{Mv\_pick}}$ (A15b) $\text{Pic\_sta} \le \overline{\text{Mv\_pick}}$ $\text{Pic\_sta} \cdot \text{Pla\_sta} = 0^*$ (A10) « (A1b) (A10) « (A3b) (A10) « (A14)

Table 5: Consistent specification of the control system

Attention shall be paid on the use of a constraint coming from the assertion A16 to obtain consistent specifications for the signals Draw_up, Mv_pla, Mv_pick and Drop_down as well as on the priority levels introduced for the same purpose in the relations sets of the latter three signals.

5.4 Control laws

Applying the synthesis technique indicated in the section 4.2.2 leads to the control laws as well as to the internal signal definition herein:

$$\begin{cases} \text{Draw\_up} = RS[(\text{Low\_pos} \cdot \text{Pic\_sta}), (\text{Low\_pos} \cdot \text{Pla\_sta})] \\ \text{Held\_part} = RS[1\text{s}/(\text{Draw\_up} \cdot \text{Low\_pos} \cdot \text{Pic\_sta}), \overline{\text{Draw\_up}}] \\ \text{Drop\_down} = RS\left[\begin{array}{l}(\text{Rai\_pos} \cdot \text{Pic\_sta} \cdot \uparrow\text{Start} + \text{Pla\_sta} \cdot \text{Held\_part}), \\ (\text{Pic\_sta} \cdot \text{Held\_part} + \text{Pla\_sta} \cdot \overline{\text{Held\_part}} + \overline{\text{Pla\_sta}} \cdot \overline{\text{Pic\_sta}} + \text{ES})\end{array}\right] \\ \text{Mv\_plac} = RS[(\text{Pic\_sta} \cdot \text{Held\_part}), (\text{Pla\_sta} + \overline{\text{Rais\_pos}} + \text{ES})] \\ \text{Mv\_pick} = RS[(\text{Pla\_sta} \cdot \overline{\text{Held\_part}}), (\text{Pic\_sta} + \overline{\text{Rai\_pos}} + \text{ES} + \text{Mv\_plac})] \end{cases}$$

This set of formal statements can be easily translated into a Programmable Logic Controller (PLC) program written in a standardized language (IEC 1993), like Ladder Diagram, a widespread programming language for PLCs (figure 8). It matters to highlight that the instruction lines of this program are ordered according to the dependency constraints previously obtained. The variable Held_part, for instance, must be elaborated once the output Draw_up is computed,

for Held_part depends on Draw_up. Conversely the current state of this variable is used to compute the three outputs Drop_down, Mv_plac, Mv_plick.

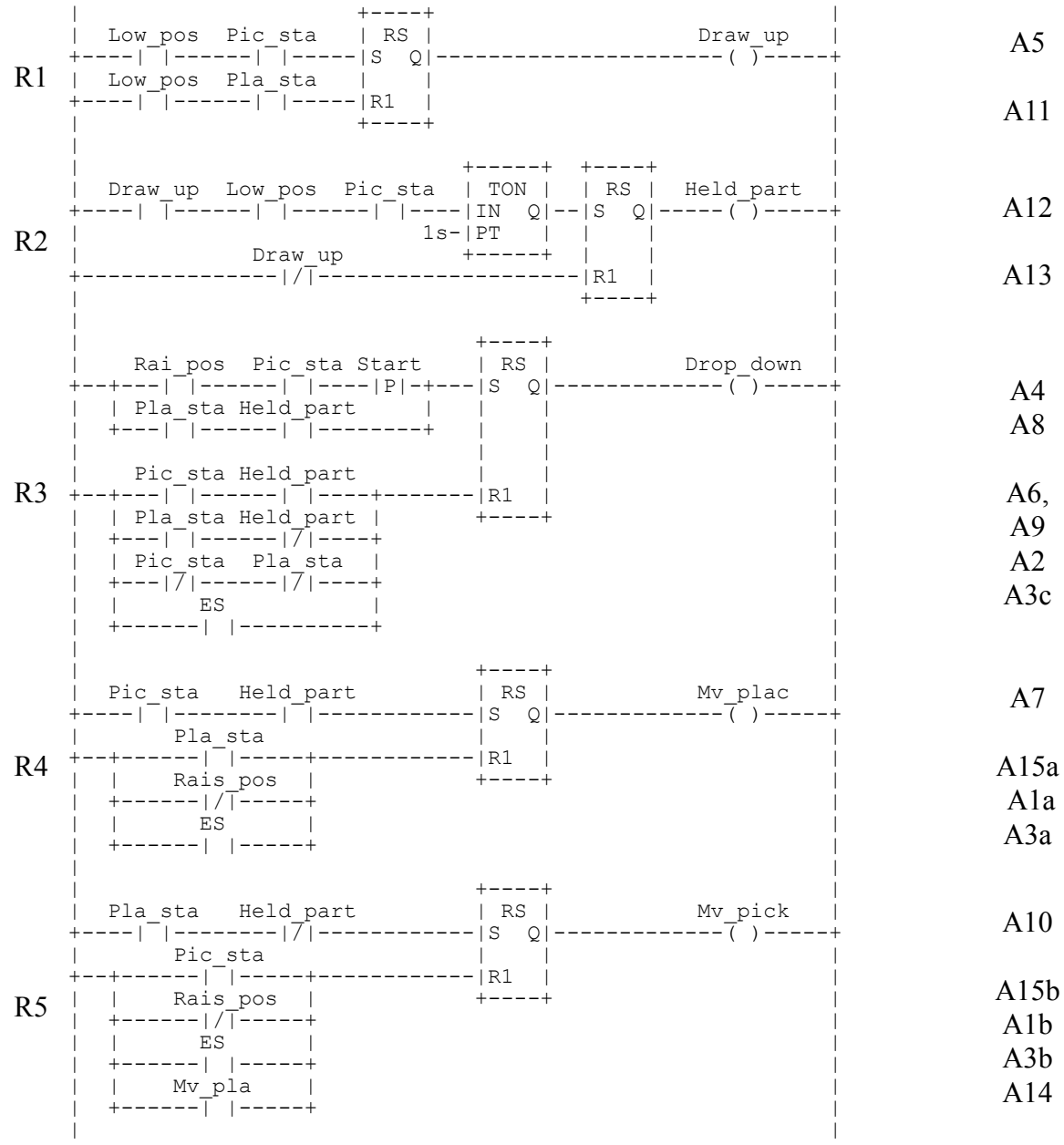


Figure 8. PLC program developed from the control laws of the example

Bridging the gap between control laws design and control software development is therefore an indirect benefit of the proposed design method.

6 Conclusion

Formal design methods may contribute efficiently to improve control systems dependability by preventing designer's faults. The first advantage of the formal method presented in this article is the ability to detect and to remove inconsistencies within the specifications of a given control system; these inconsistencies may take place in the specifications of a single output or of a set of outputs. From the consistent set of specifications obtained, control laws are stated by formal manipulations in the algebra II, the formal framework that underlies this design method.

At last, the control laws may be used to obtain a control software compliant with the specifications. The whole design method has been applied successfully to several industrial cases.

The main weak point of this method lies in the need of manipulating formal statements that are not user-friendly for automation engineers. In order to overcome this problem, our current works are aimed at developing a software assistance tool supporting the method. This software will embed operations, relations and theorems of the algebra II. An other further prospect concerns the development of a library of formal relations stating formally assertions commonly found in industrial systems specifications. Using this library will ease industrial acceptance of this design method.

References

BÉRARD, B., BIDOIT, M., FINKEL, A., LAROUSSINIE, F., PETIT, A., PETRUCCI, L., SCHNOEBELEN, P., 1999, *Systems and Software Verification: Model-Checking techniques and tools,* (Heidelberg: Springer-Verlag).

CASSANDRAS, C.G., LAFORTUNE, S., 1999, *Introduction to Discrete Event Systems,* (Boston: Kluwer).

FAURE, J.-M., LESAGE, J.-J., 2001, Methods for safe control systems design and implementations, Proceedings of 10th IFAC Symposium on Information Control Problems in Manufacturing, Vienna (Austria), CDRom paper, 6 pages.

GOUYON, D., PETIN, J.F., GOUIN, A., 2004, A pragmatic approach for modular control synthesis and implementation, *International Journal of Production Research*, same issue.

GRIMALDI, R.P., 2000, *Discrete and Combinatorial Mathematics: An Applied Introduction*, (New-York: Addison-Wesley).

INTERNATIONAL ELECTROTECHNICAL COMMISSION, 1993, IEC 61131-3, Programmable controllers - programming languages.

LAPRIE, J.C., 1992, *Dependability: basic concepts & terminology*, (Springer-Verlag).

RAMADGE, P.J., WONHAM, W.M., 1989, The control of discrete-event systems. *Proceedings of the IEEE*, **77**, 81-97.

ROUSSEL, J.-M., DENIS, B., 2002, Safety properties verification of ladder diagram programs, *Journal Européen des Systèmes Automatisés*, **36**, 905-917.

ROUSSEL, J.-M., FAURE, J.-M., 2002, An algebraic approach for PLC programs verification, Proceedings of 6th International Workshop on Discrete Event Systems (WODES'02), Zaragoza, Spain, pp. 303-308.

ZAYTOON, J., CARRÉ-MÉNÉTRIER, V., 2001, Synthesis of control implementation for discrete manufacturing systems, *International Journal of Production Research*, **39**, 329-345