Safety properties verification of ladder diagram programs

Jean-Marc Roussel, Bruno Denis

Safety Properties Verification of Ladder Diagram Programs

Jean-Marc Roussel — **Bruno Denis**

LURPA-ENS de Cachan
61 Avenue du Pt Wilson
94235 Cachan Cedex France
{Jean-Marc.Roussel,Bruno.Denis}@lurpa.ens-cachan.fr

ABSTRACT. *Programmable Logic Controllers ensure the control of many reactive systems. These controllers are most of the time programmed with the languages defined in the IEC 61131–3 standard. Our goal is the verification of safety properties of programs written in one of these languages: the Ladder Diagram. The main approaches in this field are based on Model-Checking. We propose in this article a Theorem-Proving method by defining a formal framework to express and handle the Ladder Diagram programs with a specific algebra. Firstly, we translate the specific statements of the language into this algebra and we give some general theorems. Then, we present on an example an analysis leading to the verification of safety properties.*

RÉSUMÉ. *Les automates programmables industriels assurent le contrôle-commande d'un grand nombre de systèmes réactifs. Leur programmation se fait le plus souvent avec des langages définis dans la norme IEC 61131–3. Notre objectif est la vérification de propriétés de sûreté dans les programmes écrits dans l'un de ces langages : le "Ladder Diagram". Les principales approches dans le domaine abordent le problème par "Model-Checking". Pour notre part, nous nous proposons d'explorer la voie du "Theorem-Proving" en définissant un cadre formel pour exprimer et manipuler les programmes "Ladder Diagram" dans une algèbre adaptée. Après avoir traduit les primitives de ce langage dans cette algèbre et donné des théorèmes généraux, nous présentons sur un exemple une analyse conduisant à la vérification de propriétés de sûreté.*

KEYWORDS: *Properties Proof, Boolean Algebra, Safety, Programmable Logical Controllers.*

MOTS-CLÉS : *Preuve de propriétés, Algèbre de Boole, Sûreté de Fonctionnement, Automates Programmables Industriels.*

1. Introduction

The control of industrial systems is very often carried out by Programmable Logic Controllers (PLC) programmed with one or several of the five languages described in the IEC 61131-3 standard: SFC (Sequential Function Chart), IL (Instruction List), ST (Structured Text), LD (Ladder Diagram) and FBD (Function Block Diagram). Two approaches can be used to check that a PLC program behaves safely. The first one is the well-known simulation method whose main drawback is not to be exhaustive. The second solution is the use of formal validation methods [LAM 99] [FRE 00].

Most of the works on formal validation methods of Ladder Diagram programs employ Model-Checking techniques. They are generally limited to simple programs, without function blocks as in [RAU 98]. Some works explicitly take into account the temporal aspects of programs as in [ROS 00] which introduces a semantics associated with function blocks or as in [DIE 98] which gives a semantics associated with the cyclic evaluation of programs within PLC. Few works deal with LD programs using Theorem-Proving. [SU 97] which detects relay-races or constant wires in a program can be quoted. However this work does not take into account the safety aspects.

The works presented in this article are based on an algebra called $\mathbb{I}$ firsly introduced in [ROU 93]. This algebra provides a formal framework to represent and manipulate Boolean variables states, Boolean events and physical delays between events. The present paper introduces new operations on this algebra allowing to give a semantics for LD language primitives. The benefit of this work is to obtain a theoretical background to develop theorem-proving techniques enabling to verify, on LD programs, safety properties including temporal aspects.

In section 2, an example of a reactive system is presented with the corresponding LD program and the safety properties which to have to be checked. The section 3 presents the formal framework we have developped to express and handle LD programs. The translation of two kinds of function blocks illustrates this section: standard timer function blocks (as `TON`) and standard bistable function blocks (as `RS`) illustrates this section.The last section is devoted to the exhaustive treatment of the example described in section 2.

2. Example

The size of the selected example allows us to study it entirely in this article. It is based on a water distribution plant. This example consists of a water tank, two pumping lines, an output valve and a backflow valve (see figure 1). Each pumping line is composed of a pump placed between two valves (upstream and downstream valves).

The safety properties required for this installation are stated below.

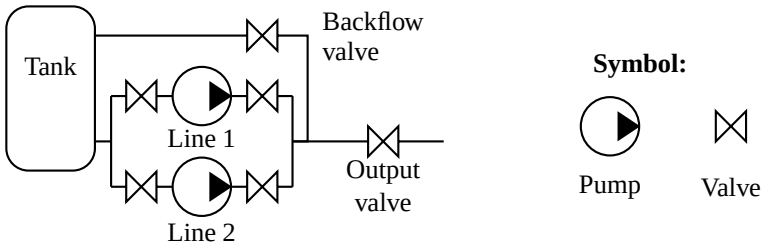


Figure 1. *The controlled system*

Properties	Formulation in natural language
P1	If a pump is on then its upstream valve has been opened for more than 5 seconds.
P2	The two pumping lines shall never work at the same time.
P3	In the case of a distribution stoppage, no actuator must be activated.
P4	In the case of pump failure, no actuator of the corresponding pumping line must be activated.

The control program of this plant is implemented into a PLC with the following input/output:

`L_flow`	operator request for a water distribution with a low flow-rate
`H_flow`	operator request for a water distribution with a high flow-rate
`Line_swap`	operator request for swapping the priority between the pumping lines
`Li_fail`	failure information from pump i
`SP_fail`	distribution stoppage information
`Li_pump`	activation order for the pump of pumping line i
`Li_up`	opening order for the upstream pump of pumping line i
`Li_down`	opening order for the downstream pump of pumping line i
`Out_valve`	opening order for output valve
`Bf_valve`	opening order for backflow valve

The control system (PLC with the control program) must respect the four here-above mentioned properties. Each property describes a part of the expected behavior of the control system. For example, P1 gives a relationship between the control system outputs, and P3 gives a causality relationship between the inputs and the outputs.

The outputs values are obtained from the inputs values and from the values of the internal variables according to the control program given on figure 2. Written Ladder Diagram language, this control program consists of 10 rungs labelled from `L01` to `L10`. After the inputs scan, the PLC evaluates the rungs from top to bottom as they appear in the LD program (for instance, in the studied example, rungs are evaluated from `L01` to `L10`), and then updated simultaneously all its outputs.

The language LD as been designed to describe the behavior of the outputs of a logical system according to its inputs and its internal variables. This graphic language

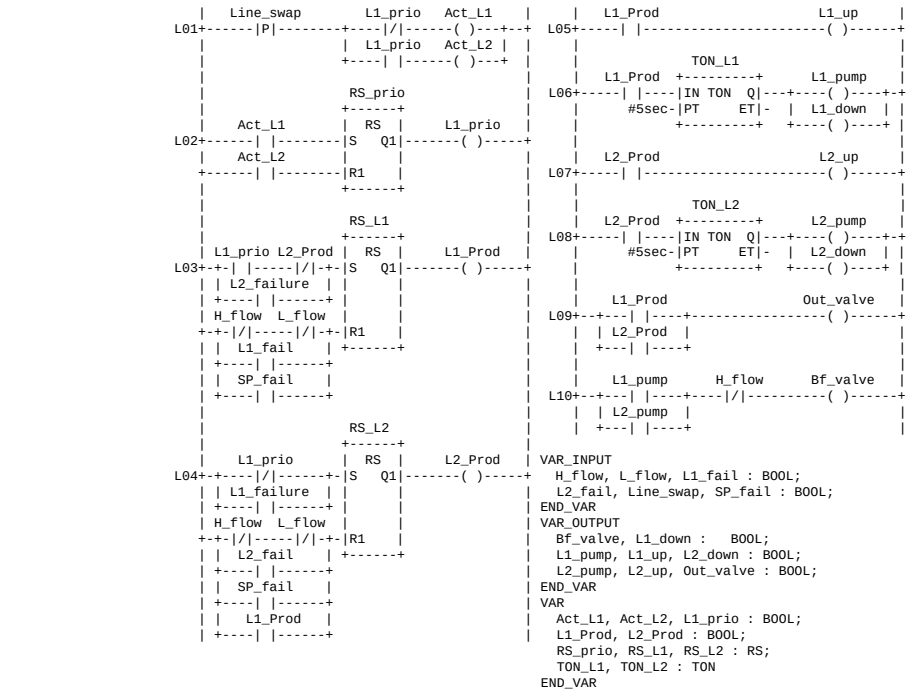


Figure 2. *Control program in Ladder Diagram language*

includes two basic operations: contacts and coils. The contacts test the value of a variable (1:-| |-, 0:-|/|-). The coils -( )- assign a value to a variable according to the logical value of the left side located expression.

When more complex behaviors are required, the user can use specific function blocks. In our example, two standardized function blocks are used: the `RS` block to memorize a Boolean information and the `TON` block to wait for a given time delay (see table 1).

3. Formal framework: a Boolean algebra for binary signals

3.1. *Binary signals modelling*

As mentioned in the introduction, the $\mathbb{II}$ algebra shall provide a formal framework to represent and manipulate Boolean variables states, Boolean events and physical delays between events. The main idea for the definition of this algebra has been to consider binary signals, *i.e.* variables describing the evolution during time of Boolean values.

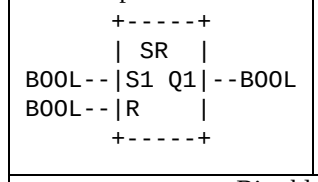
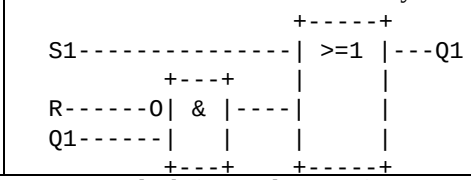
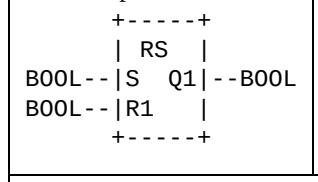
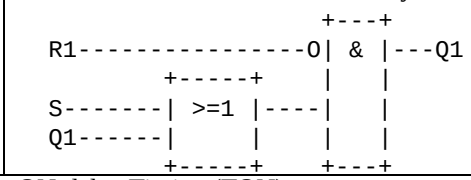
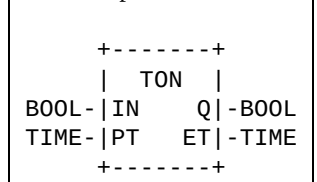
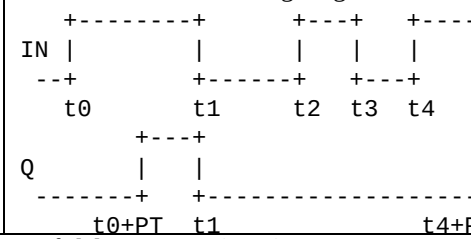
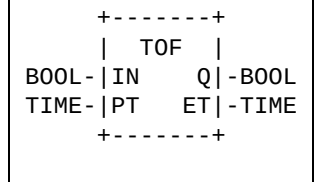
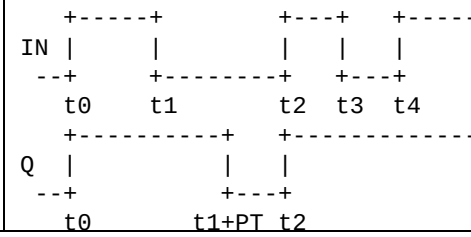
<table>
<tr><td colspan="2">Bistable Function Block (Set dominant)</td></tr>
<tr><td>Graphical form</td><td>Function Block body</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">Bistable Function Block (Reset dominant)</td></tr>
<tr><td>Graphical form</td><td>Function Block body</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">ON-delay Timing (TON)</td></tr>
<tr><td>Graphical form</td><td>Timing diagram</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">OFf-delay Timing (TOF)</td></tr>
<tr><td>Graphical form</td><td>Timing diagram</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

Table 1. *IEC 61131-3 standard definitions for* `SR`, `RS`, `TON`, `TOF` *function blocks*

These evolutions are usually represented by timing diagrams. This representation is quite useful for control engineers but is not at all based on a sound formalism. In order to provide a formal framework for binary signals, we propose to represent them as piecewise-continuous functions from $\mathbb{R}^{+*}$ to $\mathbb{B} = \{0, 1\}$. The elements of $\mathbb{I}$ are consequently formally defined in the following way:

$$\mathbb{I} = \left\{ \begin{array}{l} f : \mathbb{R}^{+*} \to \mathbb{B} \quad | \\ \forall t \in \mathbb{R}^{+*} : \left(\exists \epsilon_t > 0 : (\forall (\epsilon_1, \epsilon_2) \in \ (0, \epsilon_t)^{\,2}, f(t - \epsilon_1) = f(t - \epsilon_2)) \right) \end{array} \right\}$$

The figure 3 shows an example of a function element of $\mathbb{I}$. Attention shall be paid to the right-continuity used for the edges (at the dates t_1 and t_3) and to the double-discontinuity (for the dates t_2 and t_4), mandatory to model events. A more detailed presentation is given in [THI 00].

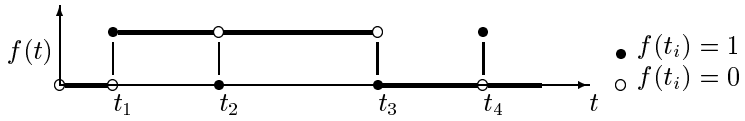


Figure 3. *Example of a function element of $\mathbb{I}$*

To distinguish the operations on the elements of $\mathbb{I}$ from the operations on the booleans, we use different notations. We note "$\wedge$" the logical AND operation between two booleans, "$\vee$" the logical OR operation between two booleans, "$\neg$" the NOT operation on a boolean. The notations ".", "$+$" and "" will be used for operations on $\mathbb{I}$. Furthermore, we have carefully distinguished the function from the boolean, *i.e.* the value taken at a given time by this function. For instance, f, g, h are three functions elements of $\mathbb{I}$ and $f(t)$, $g(t)$, $h(t)$ are three booleans.

$\mathbb{I}$ contains two special elements 1^* (the one element) and 0^* (the zero element) defined as follows:

$$\begin{array}{rccc} 1^* : & \mathbb{R}^{+*} & \rightarrow & \mathbb{B} \\ & 1^*(t) & \mapsto & 1 \end{array} \qquad \begin{array}{rccc} 0^* : & \mathbb{R}^{+*} & \rightarrow & \mathbb{B} \\ & 0^*(t) & \mapsto & 0 \end{array}$$

3.2. *Structure of Boolean Algebra*

To compose the elements of $\mathbb{I}$, we have defined three closed operations:

The AND operation	The OR operation	The NOT operation
$\mathbb{I}^2 \rightarrow \mathbb{I}$	$\mathbb{I}^2 \rightarrow \mathbb{I}$	$\mathbb{I} \rightarrow \mathbb{I}$
$(f,g) \mapsto (f.g)$	$(f,g) \mapsto (f+g)$	$f \mapsto \overline{f}$
with $\forall t \in \mathbb{R}^{+*}$,		
$(f.g)(t) = f(t) \wedge g(t)$	$(f+g)(t) = f(t) \vee g(t)$	$\overline{f}(t) = \neg f(t)$

$(\mathbb{I}, .\ , +\ , \bar{}\ , 1^*, 0^*)$ is a Boolean algebra [GRI 99] because the following conditions are satisfied for all $f, g, h \in \mathbb{I}$:

$f.g = g.f \quad f+g = g+f$	Commutative Laws
$f.(g+h) = (f.g) + (f.h)$ $f + (g.h) = (f+g).(f+h)$	Distributive Laws
$f.1^* = f \quad f + 0^* = f$	Identity Laws
$f.\overline{f} = 0^* \quad f + \overline{f} = 1^*$	Inverse Laws
$0^* \neq 1^*$	

As $(\mathbb{I}, .\ , +\ , \bar{}\ , 1^*, 0^*)$ is a Boolean algebra, the properties hereafter are satisfied:

$f.f = f \quad f + f = f$	Idempotent Laws
$f.0^* = 0^* \quad f + 1^* = 1^*$	Dominance Laws
$f.(f + g) = f \quad f + (f.g) = f$	Absorption Laws
$f.(g.h) = (f.g).h \quad f + (g + h) = (f + g) + h$	Associative Laws
$\overline{\overline{f}} = f$	Law of the Double Complement
$\overline{f.g} = \overline{f} + \overline{g} \quad \overline{f + g} = \overline{f}.\overline{g}$	DeMorgan's Laws

A partial order between the elements of $\mathbb{I}$ can be introduced by the subset relation "implication". This relation is defined as follows:

$$f \overset{\mathbb{I}}{\Rightarrow} g \quad \text{if and only if, } \forall t \in \mathbb{R}^{+*} : \quad f(t) \overset{\mathbb{B}}{\Rightarrow} g(t)$$

where $\overset{\mathbb{B}}{\Rightarrow}$ is the implication operation on $\mathbb{B}$.

This relation between elements of $\mathbb{I}$ is very useful for expressing and checking properties as shown in the last part. For all $f, g \in \mathbb{I}$, the six following relations are equivalent:

$$f \overset{\mathbb{I}}{\Rightarrow} g \quad \overline{f} + g = 1^* \quad f.\overline{g} = 0^* \quad \overline{g} \overset{\mathbb{I}}{\Rightarrow} \overline{f} \quad f.g = f \quad f + g = g$$

Once the algebra defined, it is possible to obtain a formal description of the function blocks of the IEC 61131-3 standard. In this article we will deal only with boolean memories and timers.

3.3. *Memory operations*

The bistable function blocks are defined in the IEC 61131-3 standard as shown in the table 1. Two operations on $\mathbb{I}$ have been defined for giving an algebraic semantics to bistable function blocks:

The SR operation			The RS operation		
$\mathbb{I}^2$	$\rightarrow$	$\mathbb{I}$	$\mathbb{I}^2$	$\rightarrow$	$\mathbb{I}$
(s, r)	$\mapsto$	$\mathsf{SR}(s, r)$	(s, r)	$\mapsto$	$\mathsf{RS}(s, r)$

with $\forall t \in \mathbb{R}^{+*}$,

$$\mathsf{SR}(s,r)(t) = s(t) \vee (\exists t_1 < t \mid (s(t_1) = 1) \wedge (\forall d \in\ (t_1, t],\ r(d) = 0))$$

$$\mathsf{RS}(s,r)(t) = (s(t) \wedge \overline{r}(t)) \vee (\exists t_1 < t \mid (s(t_1) = 1) \wedge (\forall d \in [t_1, t],\ r(d) = 0))$$

The value of the $\mathsf{SR}(s, r)$ (respectively $\mathsf{RS}(s, r)$) function 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 $s.\overline{r}$) at t. The second boolean is the value of a predicate at the same date. The truthfulness of the predicate depends on the existence of a former date t_1, such as $s(t_1)$ (respectively $s.\overline{r}(t_1)$) was 1 and since which the value of the function r has remained always equal to 0. Figure 4 shows 2 functions s and r of $\mathbb{I}$ and the functions resulting from SR and RS operations.

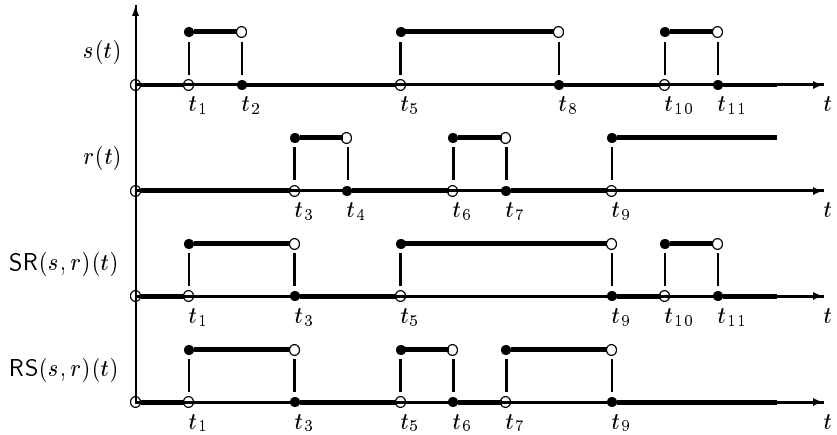


Figure 4. *Timing diagrams of functions s, r,* SR*(s,r) and* RS*(s,r)*

With these definitions, we have estabished the following theorems:

$$s \overset{I\!I}{\Rightarrow} \mathsf{SR}(s,r) \qquad \mathsf{SR}(s,\overline{s}) = s \qquad \mathsf{SR}(s,r).r = s.r \qquad \mathsf{RS}(s,r) = \mathsf{SR}(s.\overline{r},r)$$
$$r \overset{I\!I}{\Rightarrow} \overline{\mathsf{RS}(s,r)} \qquad \mathsf{RS}(s,\overline{s}) = s \qquad \mathsf{RS}(s,r).s = s.\overline{r} \qquad \mathsf{RS}(s,s) = 0^*$$
$$\mathsf{SR}(s,1^*) = s \qquad \mathsf{SR}(1^*,r) = 1^* \qquad \mathsf{SR}(0^*,r) = 0^* \qquad \mathsf{RS}(s+r.f,r) = \mathsf{RS}(s,r)$$
$$\mathsf{RS}(s,1^*) = 0^* \qquad \mathsf{RS}(1^*,r) = \overline{r} \qquad \mathsf{RS}(0^*,r) = 0^*$$
$$\mathsf{SR}(s,r_1+r_2) = \mathsf{SR}(s,r_1).\mathsf{SR}(s,r_2) \qquad \mathsf{SR}(s_1+s_2,r) = \mathsf{SR}(s_1,r) + \mathsf{SR}(s_2,r)$$
$$\mathsf{RS}(s,r_1+r_2) = \mathsf{RS}(s,r_1).\mathsf{RS}(s,r_2) \qquad \mathsf{RS}(s_1+s_2,r) = \mathsf{RS}(s_1,r) + \mathsf{RS}(s_2,r)$$

3.4. *Timing operations*

The timer function blocks are defined in the standard as shown in the table 1. The algebraic semantics of these function blocks is the following:

The TON operation			The TOF operation		
$I\!I$	$\rightarrow$	$I\!I$	$I\!I$	$\rightarrow$	$I\!I$
f	$\mapsto$	d/f	f	$\mapsto$	f/d

with $\forall t \in I\!R^{+*}$,

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

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

The TON and TOF operations transform a f function 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 $(t-d,t]$. Figure 5 shows a f function of $I\!I$ and the functions resulting from TON and TOF operations.

Theses definitions enable to establish the following theorems:

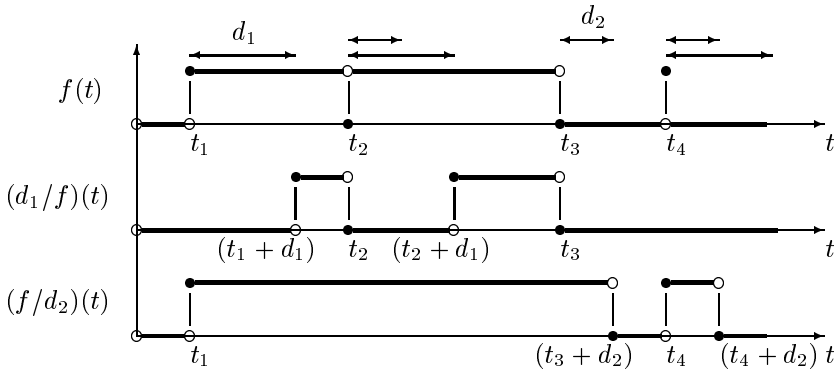


Figure 5. *TON and TOF operations for a function f of $\mathbb{I}$*

$d/f \overset{\mathbb{I}}{\Rightarrow} f$	$f \overset{\mathbb{I}}{\Rightarrow} f/d$
$d/(f.g) = (d/f)\ .\ (d/g)$	$(f+g)/d = (f/d)\ +\ (g/d)$
$(d_1/f)\ .\ (d_2/f) = max(d_1, d_2)/f$	$(f/d_1) + (f/d_2) = f/max(d_1, d_2)$
$(d_1/f) + (d_2/f) = min(d_1, d_2)/f$	$(f/d_1)\ .\ (f/d_2) = f/min(d_1, d_2)$
$d_1/(d_2/f) = sum(d_1, d_2)/f$	$(f/d_1)/d_2 = f/sum(d_1, d_2)$
$\forall t \geq d, \quad \overline{d/f} = \overline{f}/d$	$\forall t \geq d, \quad \overline{f/d} = d/\overline{f}$

4. Application to Ladder Diagram programs verification

4.1. *Expression of properties*

Defined to represent binary signals, the Boolean algebra $\mathbb{I}$ allows to us to express simply many relations between these binary signals. For a system with inputs E_i and outputs S_i, it is possible to express for instance the following relations:

Properties given in natural language	Properties written on $\mathbb{I}$
if I_1 is true, then O_1 is true.	$I_1 \overset{\mathbb{I}}{\Rightarrow} O_1$
O_1 and O_2 are never simultaneously true.	$O_1\ .\ O_2\ =\ 0^*$
The binary signal O_1 is never true more than 3 seconds.	$3s/O_1\ =\ 0^*$

Safety properties given in section 2 shall be written on $\mathbb{I}$ as shown in table 2.

4.2. *Modelling of the Ladder Diagram program behavior*

Only the temporal aspects of the process will be considered in this paper. The temporal aspects inherent in the PLC technology will be neglected compared to those of the process. Thus, the model retained for the PLC monitor is an infinitely reactive model *i.e.* with a delay response time equal to zero. With the algebraic semantics given to LD operations, the LD program behavior is a set of expressions defined on $\mathbb{I}$.

Properties	Formulation on $\mathbb{I}$
P1	$\begin{cases} L1\_pump \overset{\mathbb{I}}{\Rightarrow} 5s/L1\_up \\ L2\_pump \overset{\mathbb{I}}{\Rightarrow} 5s/L2\_up \end{cases}$
P2	$(L1\_up + L1\_pump + L1\_down)\,.$ $(L2\_up + L2\_pump + L2\_down) \;=\; 0^*$
P3	$SP\_fail \overset{\mathbb{I}}{\Rightarrow} \overline{L1\_pump}\,.\,\overline{L1\_up}\,.\,\overline{L1\_down}\,.\,\overline{L2\_pump}$ $.\,\overline{L2\_up}\,.\,\overline{L2\_down}\,.\,\overline{Out\_valve}\,.\,\overline{Bf\_valve}$
P4	$\begin{cases} L1\_fail \overset{\mathbb{I}}{\Rightarrow} \overline{L1\_pump}\,.\,\overline{L1\_up}\,.\,\overline{L1\_down} \\ L2\_fail \overset{\mathbb{I}}{\Rightarrow} \overline{L2\_pump}\,.\,\overline{L2\_up}\,.\,\overline{L2\_down} \end{cases}$

Table 2. *Formulation on $\mathbb{I}$ of safety properties*

From a theoretical point of view, the sequential program written in LD can be seen as a machine which processes input binary signals to produce ouput binary signals. Each input and ouput can be model with a $\mathbb{I}$ function. Then each output signal can be model as an algebraic function of $\mathbb{I}^n$ in $\mathbb{I}$ (where n is the number of input signals). In other words the transfer function of PLC can be expressed algebraically in $\mathbb{I}$. For our example, the model would be:

$$\begin{cases} Bf\_valve &= f_1\,(H\_flow,\ L\_flow, L1\_fail, \\ & \qquad L2\_fail,\ Line\_swap, SP\_fail) \\ & \vdots \\ Out\_valve &= f_8\,(H\_flow,\ L\_flow, L1\_fail, \\ & \qquad L2\_fail,\ Line\_swap, SP\_fail) \end{cases}$$

From a practical point of view, it is not necessary to know explicitly this transfer function to check most of safety properties. The knowledge of some properties of this transfer function is often sufficient. Using the structure of the program LD, a certain number of these properties are simple to express because control programs of industrial applications are always written with strict methodological rules in order to facilitate the program maintenance. Popular rules are the following: modular design of program, assignment of the variables at only one place, restricted (or even prohibited) use of the operation modifying the order of rung execution.

The LD program proposed in section 2 was designed in a modular manner. Its structure is presented on the figure 6.

The behavior model retained for this LD program is a set of algebraic equations which represent either a part of the transfer function of the LD program, or relations between variables. For instance, for the module including the rungs from L05 to L10, the transfer functions are as follows (from [1] to [8]).

$$L1\_up = L1\_prod \qquad [1]$$

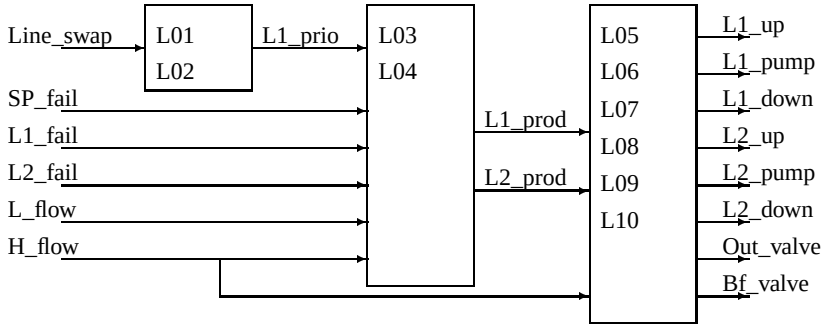


Figure 6. *Modular structure of the LD program*

$$L1\_pump = 5s/L1\_prod \quad [2]$$
$$L1\_down = 5s/L1\_prod \quad [3]$$
$$L2\_up = L2\_prod \quad [4]$$
$$L2\_pump = 5s/L2\_prod \quad [5]$$
$$L2\_down = 5s/L2\_prod \quad [6]$$
$$Out\_valve = L1\_prod \ + \ L2\_prod \quad [7]$$
$$Bf\_valve = (5s/L1\_prod \ + \ 5s/L2\_prod) \ . \ \overline{H\_flow} \quad [8]$$

On the opposite side, it is not possible to express the transfer function of the module including the rungs L03 and L04 with the only operations described in section 3. These transfer functions will be replaced by relations derived from the transfer function. These properties are expressed using implications in $I\!I$.

$$\overline{H\_flow}.\overline{L\_flow} + L1\_fail + SP\_fail \overset{I\!I}{\Rightarrow} \overline{L1\_Prod} \quad [9]$$
$$\overline{H\_flow}.\overline{L\_flow} + L2\_fail + SP\_fail + L1\_Prod \overset{I\!I}{\Rightarrow} \overline{L2\_Prod} \quad [10]$$

4.3. *Verification of the LD program properties*

The proof of properties is obtained by applying a succession of theorems on $I\!I$.
To prove **P1**, it is necessary to establish the two relations: "$L1\_pump \overset{I\!I}{\Rightarrow} 5s/L1\_up$" and "$L2\_pump \overset{I\!I}{\Rightarrow} 5s/L2\_up$". The first relation is demonstrated in figure 7. The demonstration of the second relation is similar.
The property **P2** is proved in figure 8.
The properties **P3** and **P4** are proved in the same way. To prove theses properties, it is necessary to establish each elementary relation such as "$SP\_fail \overset{I\!I}{\Rightarrow} \overline{Bf\_valve}$".
The demonstration of "$SP\_fail \overset{I\!I}{\Rightarrow} \overline{Bf\_valve}$" is made in figure 9.
Finally, the four safety properties expected for the system are obtained for the given LD program.

	Reasons
$L1\_up = L1\_prod$	using [1]
$L1\_pump = 5s/L1\_prod = 5s/L1\_up$	using [2]
$L1\_pump \overset{I\!I}{\Rightarrow} 5s/L1\_up$	consequently

Figure 7. *Demonstration of "$L1\_pump \overset{I\!I}{\Rightarrow} 5s/L1\_up$"*

	Reasons
$(L1\_up + L1\_pump + L1\_down) = L1\_prod + 5s/L1\_prod$	using [1], [2], [3]
$L1\_prod + 5s/L1\_prod = L1\_prod$	using theorem $5s/f \overset{I\!I}{\Rightarrow} f$
$(L1\_up + L1\_pump + L1\_down) = L1\_prod$	using precedent results
$(L2\_up + L2\_pump + L2\_down) = L2\_prod$	result obtained in the same way
$(L1\_up + L1\_pump + L1\_down).(L2\_up + L2\_pump + L2\_down) = L1\_prod \,.\, L2\_prod$	using precedent results
$L1\_Prod \overset{I\!I}{\Rightarrow} \overline{L2\_Prod}$	using [10]
$(L1\_up + L1\_pump + L1\_down).(L2\_up + L2\_pump + L2\_down) = (L1\_prod \,.\, \overline{L2\_Prod}) \,.\, L2\_prod = 0^*$	Consequently

Figure 8. *Demonstration of property **P2***

5. Conclusion and perspectives

The $I\!I$ algebra provides a formal framework to represent Boolean variables states, events and physical delays and has permitted to develop the verification method presented in this article. This method has been tested in several cases with success. It is particularly well-suited for structured programs as industrial ones. We have developped under Mathematica® a solver enabling automatic verification. With this solver, the four properties of the example have been automatically proved.

The example described in this article is written in LD; the same equations and reasoning would be obtained with a program in Function Block Diagram (FBD). Moreover the presented function blocks (`RS` , `TON`) are defined for all the IEC 61113-3 languages (*e.g.* SFC) ; the results obtained can be therefore applied to any program developed in these languages.

The perspectives of these works are both formal and methodological. New operations increasing the potentiality of checking in $I\!I$ are under development. They will be usefull to express properties. From a methodological point of view, we have to consider the cooperation between the two verification methods nowadays used in our laboratory: model-checking and symbolic reasoning in $I\!I$. Rational and comple-

	Reasons
$SP\_fail \overset{\text{II}}{\Rightarrow} \overline{L1\_Prod}$	using [9]
$SP\_fail \overset{\text{II}}{\Rightarrow} \overline{L2\_Prod}$	using [10]
$SP\_fail \overset{\text{II}}{\Rightarrow} \overline{L1\_Prod} \,.\, \overline{L2\_Prod}$	using precedent results
$\overline{L1\_Prod} \,.\, \overline{L2\_Prod} \overset{\text{II}}{\Rightarrow} \overline{5s/L1\_Prod} \,.\, \overline{5s/L2\_Prod}$	using theorem $5s/f \overset{\text{II}}{\Rightarrow} f$
$\overline{5s/L1\_Prod} \,.\, \overline{5s/L2\_Prod} \overset{\text{II}}{\Rightarrow} \overline{Bf\_valve}$	using [8]
$SP\_fail \overset{\text{II}}{\Rightarrow} \overline{Bf\_valve}$	consequently

Figure 9. *Demonstration of "*$SP\_fail \overset{\text{II}}{\Rightarrow} \overline{Bf\_valve}$*"*

mentary use of these two approaches will be of benefit for large size industrial PLC programs verification.

6. References

[DIE 98] DIERKS H., FEHNKER A., MADER A., VAANDRAGER F., "Operational and Logical Semantics for Polling Real-Time Systems", RAVN A., RISCHEL H., Eds., *Proceeding of Fifth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, vol. 1486 of *Lecture Notes in Computer Science*, Lyngby, Denmark, April 1998, Springer, p. 29–40.

[FRE 00] FREY G., LITZ L., "Formal methods in PLC programming", *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Nasville, Tennessee, USA, October 2000.

[GRI 99] GRIMALDI R. P., *Discrete and Combinatorial Mathematics: An Applied Introduction*, Addison-Wesley, 4th edition, 1999, ISBN 0-201-19912-2.

[LAM 99] LAMPÉRIÈRE-COUFFIN S., ROSSI O., ROUSSEL J.-M., LESAGE J.-J., "Formal validation of PLC programs: a survey", *Proceedings of European Control Conference 1999 (ECC'99)*, Karlsruhe, Germany, August/September 1999.

[RAU 98] RAUSCH M., KROGH B. H., "Formal validation of PLC programs", *Proceedings of American Control Conference*, Philadelphia, PA, USA, June 1998.

[ROS 00] ROSSI O., SCHNOEBELEN P., "Formal modeling of timed function blocks for the automatic verification of ladder diagram programs", *Proceeding of International Conference Automation of Mixed Processes (ADPM'2000)*, Dortmund, Germany, September 2000, p. 177–182.

[ROU 93] ROUSSEL J.-M., LESAGE J.-J., "Une algèbre de Boole pour l'approche événementielle des systèmes logiques", *APII-AFCET/CNRS*, vol. 27, n° 5, 1993, p. 541–560.

[SU 97] SU Z., "Automatic analysis of Relay Ladder Logic programs", report n°UCB/CSD-97-969, September 1997, Computer Science Division, University of California, Berkeley, CA, USA.

[THI 00] THIERRY C., ROUSSEL J.-M., LESAGE J.-J., "An extended boolean algebra for the control of logical systems", *Proceedings of 16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation*, Lausanne, Switzerland, August 2000.