

Formal design methodology for transforming ladder diagram to Petri nets

**J. C. Quezada, J. Medina, E. Flores,
J. C. Seck Tuoh & N. Hernández**

**The International Journal of
Advanced Manufacturing Technology**

ISSN 0268-3768

Int J Adv Manuf Technol
DOI 10.1007/s00170-014-5715-9



Your article is protected by copyright and all rights are held exclusively by Springer-Verlag London. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".

Formal design methodology for transforming ladder diagram to Petri nets

J. C. Quezada · J. Medina · E. Flores · J. C. Seck Tuoh · N. Hernández

Received: 4 June 2013 / Accepted: 10 February 2014
© Springer-Verlag London 2014

Abstract Ladder diagram (LD) is a common programming language at industry in order to develop control algorithms of discrete event systems. Besides, it is one of the five programming languages supported by the International Electrotechnical Commission through the IEC-61131-3 standard. Petri net (PN) theory is both a graphical and mathematical tool, which allows modeling discrete event systems in order to obtain a useful formalization to analyze them in a better way. LD control algorithms are continuously developed based on the experience of control system developers. Therefore, it is still a relevant problem on how to formalize a validation for the current and new control algorithms. In the present work, an element-to-element transformation methodology from a LD program to a PN structure is proposed. The original part of this manuscript is the proposal of five PN structures where their markings represent the states and dynamic behavior of energized and de-energized coils, which are not included in previous works. Furthermore, this

methodology preserves the structural and dynamical behavior of the LD in the obtained PN. Two control algorithms of real cases are transformed using the proposed methodology.

Keywords Control algorithms · Ladder diagrams · Petri nets · Discrete event systems

Mathematics Subject Classifications (2010) 68Q60 · 68Q85

1 Introduction

Programmable logic controllers (PLCs) are widely applied in an industry for the process control, mainly for discrete event systems such as interlocks, production and/or manufacturing sequences, process alarms, among others. In the IEC-61131-3 standard [1], the syntaxes and semantics of five programming languages for logic controllers are established: ladder diagram (LD), function block diagram (FBD), instruction list (IL), structured text (ST), and sequential function chart (SFC). LD is considered a graphic-type language based on the behavior of an electromagnetic relay constituted by a coil and contacts that can be normally open (NO) and/or normally closed (NC). Due to its likeness and utilization with electrical control diagrams, LD has been the most popular language used in the industry for developing control algorithms.

The control LD algorithms are mainly based on the user experience applying trial-and-error techniques. Therefore, a formalization of the validity of the existing control algorithms remains as a relevant problem, as well as the specification of new methodologies to validate the control algorithms before their real implementation. In addition, PN is both a graphical and mathematical tool which is applied

J. C. Quezada (✉) · E. Flores
Escuela Superior de Tizayuca, Universidad Autónoma del Estado de Hidalgo, km 2.5 carretera federal, Tizayuca-Pachuca, México
e-mail: jcarlos@uaeh.edu.mx

E. Flores
e-mail: efloresg@uaeh.edu.mx

J. Medina · J. C. Seck Tuoh · N. Hernández
Centro de Investigación Avanzada en Ingeniería Industrial,
Universidad Autónoma del Estado de Hidalgo,
km 2.5 carretera federal, Tizayuca-Pachuca, México

J. Medina
e-mail: jmedina@uaeh.edu.mx

J. C. Seck Tuoh
e-mail: jsek@uaeh.edu.mx

N. Hernández
e-mail: nhromero@uaeh.edu.mx

to model discrete event systems (DES) in order to describe their static and/or dynamic behavior [2]. A PN analysis method for the dynamic behavior is the state equation composed of an incidence matrix, an initial marking vector, and a firing vector.

Various approaches have been presented in research papers to study the relationship between LD and PN for analysis, modeling, and simulation of control algorithms and its corresponding validation. In [3], Lee and Lee propose the modulus synthesis technique for the conversion from LD to PN, considering the LD base cores defined in the IEC-61131-3 standard. It is important to highlight that for the normally open and normally closed contacts, they propose one place with two transitions joined together by an enabler arc and an inhibitor arc, respectively, in order to relate their active state, open state, or closed state. The authors present an example where they show operations AND and OR, considering that all the variables are independent, that is, the coil they use has no contacts defined in the code lines of the LD.

Thapa et al. [4] define five types of PN structures of input-output units (IOU) which are equivalent to common structures used in control algorithms developed in LD: start-stop unit, basic unit, AND and OR unit, basic unit with functions block, and unit of transmission concatenation of logical flow. In [5], Peng and Zhou make reference to graphical-type constructors of logics AND, OR, and sequential modeling, showing the equivalences between PN and LD. In [6], Lee and Hsu show the equivalences between PN and LD, and IF-THEN rules which are logical structures AND, OR, and parallel or concurrent distribution. Zapata and Carrasco [7] propose a representation in a graphical way of logical constructors AND, OR, S-R memory function, edge detection, concurrency and delays on PN and in LD; however, they will validate the equivalences in a future work.

In [8], Tzafestas et al. propose a technique to generate a LD starting from a PN representing an control algorithm, standing out the importance of the self-loop in DES, and considering the LD rows as two parts: “conditional part” and “action part” which are represented in the PLC memory, assigning consequently a bit for a place in the PN at the conditional part, and other bit for a transition in the PN at the action part.

In [9], Korotkin et al. propose a methodology to implement PN models for discrete event control systems (DECS) of high level. They define a 10-tuple net called “PNDEC” and their dynamic marking equations. The PNDEC is considered as a binary net and its evaluation is performed in a single swept of the control algorithm based on Boolean conditions assigned to the output places. The dynamic marking is generalized; however, it does not consider initial states with energized coils.

In [10], the authors use truth tables that represent the state of the process (Karnaugh maps) to determine the Boolean function transitions and show an example of an engine considering the start and stop in a single control line of LD.

The application of colored PN (CPN) for obtaining automatically formal models directly from the LD is shown in [11]. They consider that each control line in LD can be expressed as a formula written in propositional logic. Their focus is based on the sequential steps of the PLC: reading inputs, cycle start, and memory access. These elements are analyzed considering energy flows as rows, without taking into account the scan operation of the PLC.

Another proposal of a systematic method to translate from LD to ordinary Petri nets is presented in [12]. They define a LD graph which is transformed into an ordinary Petri net; this approach considers only the basic parts of relays (contacts and coils). They define two states for the types of contacts and coils and perform searches of the “closed” trajectories to find energized coils and “open” trajectories for de-energized coils in LD graphs. Nevertheless, they do not include the logic operation for negated coils.

In general, to our knowledge, the transformation proposals consider only the behavior of the logical structures when the input prerogatives are fulfilled to energize coils, or there are no negated coils.

In this paper, we propose a new methodology for transforming control algorithms developed in LD to PN. In particular, we show five PN structures in order to represent the control algorithms developed in LD (logic AND, logic OR, logic AND-OR, logic set-reset and logic crossed-contacts). This transformation is called ladder diagram - Petri net (*LDPN*). With this transformation, the original contribution of this paper is to establish a new methodology to model the behavior of energize and de-energize coils.

The proposed methodology was evaluated on two real control algorithms, obtaining results that show the structural and dynamic equivalence between the original LD and the obtained *LDPN*. The first case needs to energize and de-energize a coil, and the second one shows the implementation of the methodology for a packaging system.

This paper is organized as follows. Section 2 and 3 describe the basic concepts of LD and PN, respectively. Section 4 explains the five structures to define a *LDPN* and how to transform a LD into a *LDPN*. Section 5 shows two real cases, and the final section presents the conclusions of the paper.

2 Ladder diagram

The graphic language LD models nets of electromechanical elements operating simultaneously, such as relays containing coils and contacts, timers, and counters, mainly [13],

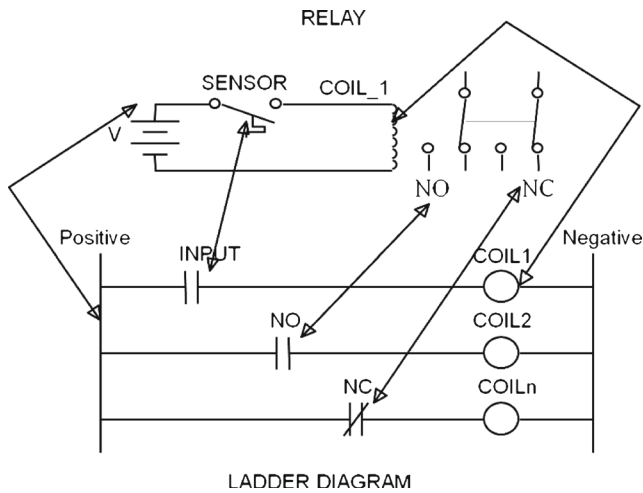


Fig. 1 Equivalence between a relay and the corresponding LD code

although the last two are blocks of their respective functions. Figure 1 shows the equivalence between the relays operating principles and LD. The LD is composed of two rails, positive rail and negative rail, and the contacts are connected between them. Contacts are permissive to energize or de-energize a coil which closes the circuit with the negative rail. The energy flow is analogous to the electric power flow, and it goes from left to right in the LD.

Furthermore, running the control algorithm in the PLC is given cyclically with well-defined tasks, from which three are covered in this work: physical input (sensors) reading, running of the control algorithm (LD), and physical output (actuators) writing, as shown in Fig. 2 [10]. Internally in the PLC, an snapshot about the state memory of the signals corresponding to the physical input reading is accomplished. With this snapshot, the control algorithm is executed in LD and subsequently other snapshot about the status kept by the output signals in the PLC memory is generated. Then, the snapshot is transferred to the corresponding output modules, and so on, in a cyclic way.

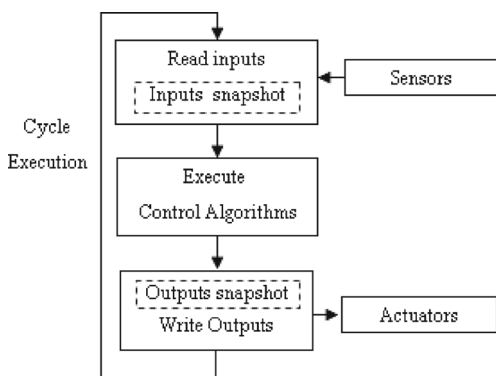


Fig. 2 Cyclic running of a PLC control algorithm

3 Petri nets

DES are modeled fundamentally by using the concepts of conditions and events, where the line contacts and places represent conditions in LD and PN, respectively, and coils and transitions represent events in LD and PN, respectively. Table 1 shows the formal definition of a PN in its basic form [2].

For the general simulation of the dynamic behavior, based on an initial marking of the PN, the following transition rule from [2] is considered:

1. A transition t is enabled if each input place p of t contains $w(p, t)$ tokens, where $w(p, t)$ is the arc weighting joining p to t .
2. An enabled transition can either be or not be firing, depending on whether the event really occurs or not.
3. The firing of an enabled transition t , moves $w(p, t)$ tokens of each input place and adds $w(t, p)$ to each output place.

There are two special transitions: the *source* transition which is the one having no input places and the *sink* transition which has no output places. A *self-loop* in a PN arises when the input place is also the output place in the same transition. A net free of self-loops is called *pure* and a PN where each arc weight is 1 is named *ordinary*.

3.1 Incidence matrix and state equation

In order to represent the dynamic behavior of the PNs, the incidence matrix is used, which relates the weightings of input and output arcs from transitions to places and vice versa. For a PN with n transitions and m places, its incidence matrix $A = [a_{ij}]$ is an integer number matrix representing the weighing of the input and output arcs; a_{ij}^+ represents the weighting of outputs arcs from transitions, and a_{ij}^- represents input arcs to transitions. Equation 1 represents the incidence matrix.

$$a_{ij} = a_{ij}^+ - a_{ij}^- \tag{1}$$

The state equation shows the marking in a sequence state through the relationship between the vector of a preceding state with certain system marking M_{k-1} , the transpose of the

Table 1 Formal definition of a PN

A Petri net is a 5-tuple, $PN=(P,T,F,W,M)$ where:
 $P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places,
 $T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions,
 $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
 $W : F \rightarrow \{1, 2, 3, \dots\}$ is a weight function,
 $M_0 : P \rightarrow \{0, 1, 2, \dots\}$ is an initial marking, and
 $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

incidence matrix A and a firing vector u_k determining the process firing sequence. Equation 2 shows the relationship between them.

$$M_k = M_{k-1} + A^T u_k \tag{2}$$

Taking into account the previous PN concepts, the next section describes the methodology applied to represent a given LD by means of an equivalent PN, which is called *LDPN*.

4 Formal definition of LDPN

In previous works, only the behavior of energizing coils has been considered to transform LD into PN. In actual control algorithms, however, depending on the system control, it is necessary to energize or de-energize coils. For this reason, it proposed a methodology to transform LD control lines to PN structures. This transformation is based on the definition of five PN structures which represent the most common instruction lines in LD. With these PN structures, this methodology is able to obtain a structural and dynamic equivalent PN for a given LD considering as well the behavior of energized and de-energized coils.

4.1 Representing LD signals with PN

A particular interpretation of the PN is to consider places as input and output signals from sensors and actuators, respectively, and transitions as logical conditions of the systems. This interpretation is the generalization of a PLC-based system: input signals (sensors) — control algorithm (LD) — output signals (coils).







The analysis and evaluation of the transformed PN are accomplished considering a single PLC scan cycle, that is, considering the snapshot charts keeping the states of the physical and memory signals during a cycle. Thus, the dynamic markings of the proposed PN structures conserve their marking in the respective analysis. Table 2 presents the graphical symbols used in the transformation from LD to PN.

Considering symbols of [2], for a preset and post-set of places, are defined:

- $\bullet t = \{p : (p, t) \in F\}$, the set of input places of t .
- $t \bullet = \{p : (t, p) \in F\}$, the set of output places of t .

The problem of token accumulation in PN places is mainly solved in two forms. In the first one, the authors of [9] and [11] consider the execution of control algorithms with a state snapshot of signals in the PLC, which are only updated, and so this updated should be represented in the PN, i.e., they consider that tokens in places are only updated. The second one uses inhibitor arc from output place to transition to prevent the tokens accumulation [3].

Table 2 Graphical representation of signals

Symbol	Description
	Place for physical signal of input, output, and memory
	Tokens for energy flow condition
	Cumulative transition for timed or counting events
	Transition as logical condition
	Arc
	Inhibitor arc

We proposed logical and arithmetic functions at places to avoid the token accumulation. Equations 3 and 4 are applied to avoid the token accumulation in output places.

$$M(t \bullet)_{AND} = \sum M(\bullet t) = |\bullet t| \text{ OR } \sum M(t \bullet) = 1 \tag{3}$$

$$M(t \bullet)_{OR} = \left\lceil \sum M(\bullet t) / |\bullet t| \right\rceil = 1 \text{ OR } \sum M(t \bullet) = 1 \tag{4}$$

where the symbol $\lceil a \rceil$ is the ceil function for a real number a .

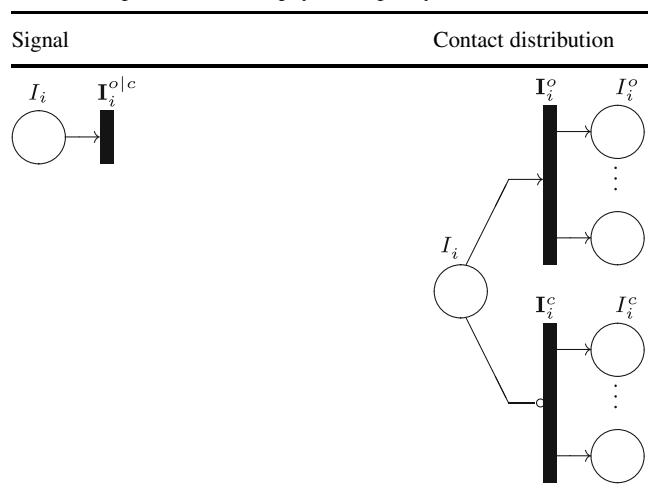
Now, to reset output places, the Eqs. 5 and 6 are proposed for the marking of input places, called $G(t)$. As in PLC scan, tokens are updated to evaluate the functions of reset places.

$$G(t)_{AND} = \overline{\prod M(\bullet t)} \text{ AND } M(t \bullet) = 1 \tag{5}$$

$$G(t)_{OR} = \overline{\sum M(\bullet t)} \text{ AND } M(t \bullet) = 1 \tag{6}$$

Table 3 shows the signal distribution (NO and/or NC contacts) for a physical input, using PN elements. Notice that this representation is analogous for a physical output or a physical memory signal. Equations 7a, 7b, and 7c give the

Table 3 Representation of a physical input by PN elements



number of NO and/or NC contacts for these types of signals. The transitions I_i^o are enabled when its input place has a token, and transitions I_i^c are enabled when its input place no has a token.

$$I_i = I_i^o I_i^o + I_i^c I_i^c \tag{7a}$$

$$O_o = O_o^o O_o^o + O_o^c O_o^c \tag{7b}$$

$$B_b = B_b^o B_b^o + B_b^c B_b^c \tag{7c}$$

where I_i^o , O_o^o , and B_b^o are the number of NO contacts, and I_i^c , O_o^c , and B_b^c are the number of NC contacts for the signals I_i , O_o , and B_b at the corresponding LD.

Marking in output places I_i^o and I_i^c is in function of the Eqs. 8 and 9 and is similar to the marking of output places of signals O_o and B_b .

$$M(I_i) = 1 \text{ then } M(I_i^c) = 0 \tag{8}$$

$$M(I_i) = 0 \text{ then } M(I_i^o) = 0 \tag{9}$$

Five types of control lines are the most common ones in control algorithms developed in LD. These types are the following: serial contacts (*logical AND*), parallel contacts (*logical OR*), contacts using the previous connections at the same time (*logical AND-OR*), logical *set-reset* coils, and logical *interlocking contacts*.

1. Serial contacts.

In Fig. 3, when the input contacts allow the flow of energy, the coil *OUT1* is energized. If the status of any of the input contacts changes, the coil *OUT1* is de-energized. Control lines with only serial contacts allow that coils to behave as a logical AND.

2. Parallel contacts.

Control lines with parallel contacts allow that coils to behave as a logical OR (see Fig. 4). If any contact allows the flow of energy, then the coil *OUT1* is energized. Only when all input contacts are inactive, the coil *OUT1* is de-energized.

3. Logical AND-OR contacts.

In order to obtain a latching behavior, it is necessary to combine the logical AND and OR operations. In Fig. 5, the output *OUT1* is de-energized by means of a sequence of input contacts and it keeps energized through a contact of the output *OUT1* in parallel to *IN1*. The coil is de-energized when any of the serial contacts

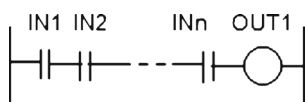


Fig. 3 Serial contacts

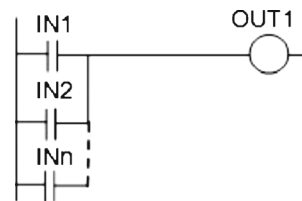


Fig. 4 Parallel contacts

of the sequence changes its state. In general, a coil can be energized by several possible arrangements of serial and parallel contacts.

4. Logical set-reset coils.

The logic of asynchronous signals is used as well in LD through coils called set and reset with the same addressing. These coils need one or more control signals to energize *OUT1-S* or de-energize it by means of *OUT-R* as shown in Fig. 6.

5. Logical interlocking contacts.

The interlock coils in control lines are widely used in LD. They are constructed by placing a contact of a coil in the control line of another one, as shown in Fig. 7. There, a contact of *OUT1* is in the control line of the coil *OUT2* and vice versa.

With the elements of Table 2, the signals distributed in Table 3, and considering the Eqs. 3, 4, 5, and 6, five PN structures are proposed below.

4.2 PN structures representing control lines of a LD

Initial marking of the PN structures is based on the types of contacts in the LD, which are shown in Eq. 10. Their dynamic markings describe the behavior of energized or de-energized coils.

$$M_{(I_i|O_o|B_b)} = \begin{cases} 1, & \text{if it only has NO or NC contacts,} \\ 2, & \text{if it has NO and NC contacts.} \end{cases} \tag{10}$$

1. Logical AND.

In Fig. 8, the transition I_1^o is enabled when all its input places have at least one token. The I_1^o firing puts a marking at place O_1 , which denotes the behavior in a scan of the LD. $G(I_1^o)_{AND}$ resets the place O_1 if there is a token in O_1 and any place in $\bullet I_1^o$ does not have a token.

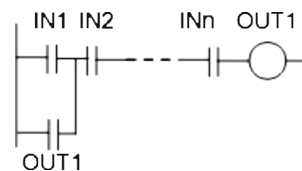


Fig. 5 Serial and parallel contacts

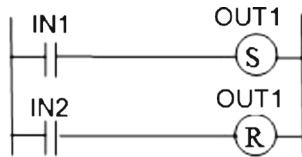


Fig. 6 Logical set-reset coils

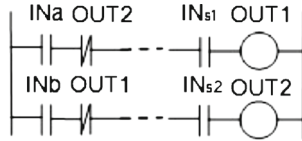


Fig. 7 Interlocking contacts

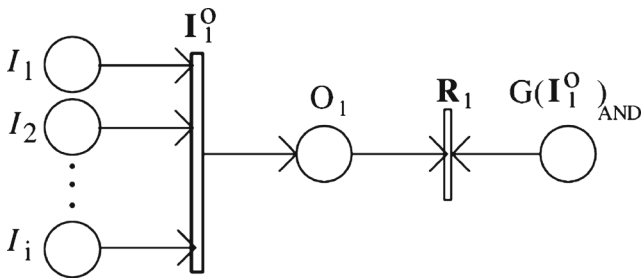


Fig. 8 Logical AND

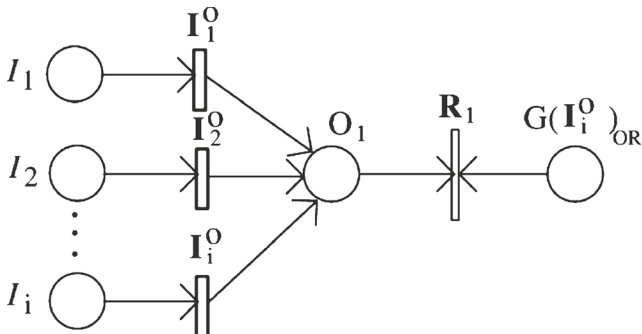


Fig. 9 Logical OR

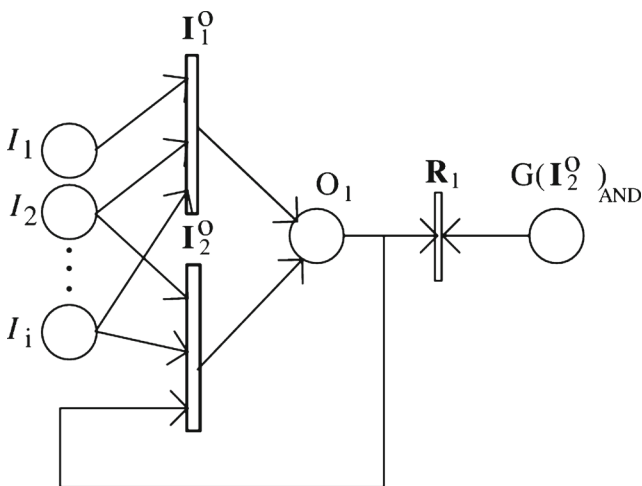


Fig. 10 Logical AND and OR

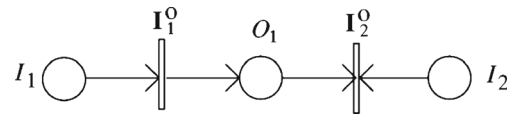


Fig. 11 Logical set-reset

2. Logical OR.

Figure 9 shows the logical OR. A token in any of the input places in $\bullet I_i^O$ will fire the corresponding transition and put a token in O_1 , for $i = 1, \dots, n$. Only when all the input places do not have a token and O_1 has a token, then a token is put in the place $G(I_i^O)_{OR}$ enabling the transition R_1 to consume the token of O_1 . This is equivalent to de-energize the coil OUT_1 in the LD of Fig. 4.

3. Logical AND-OR.

The PN structure with both logical AND and OR is shown in Fig. 10. When the transition I_i^O is fired, it puts a token in O_1 in order to feed back I_2^O to keep the token in O_1 . $G(I_2^O)_{AND}$ resets O_1 if there is a token in O_1 and any place in $\bullet I_2^O$ does not have a token. In this case, it is necessary to put two tokens in all the places of $\bullet I_1^O \cap \bullet I_2^O$.

4. Logical set-reset.

This PN structure set-reset, is a particular case (see Fig. 11). A token is put in I_1 if there is neither a token in O_1 nor I_2 (Eq. 11). On the other hand, a token is put in

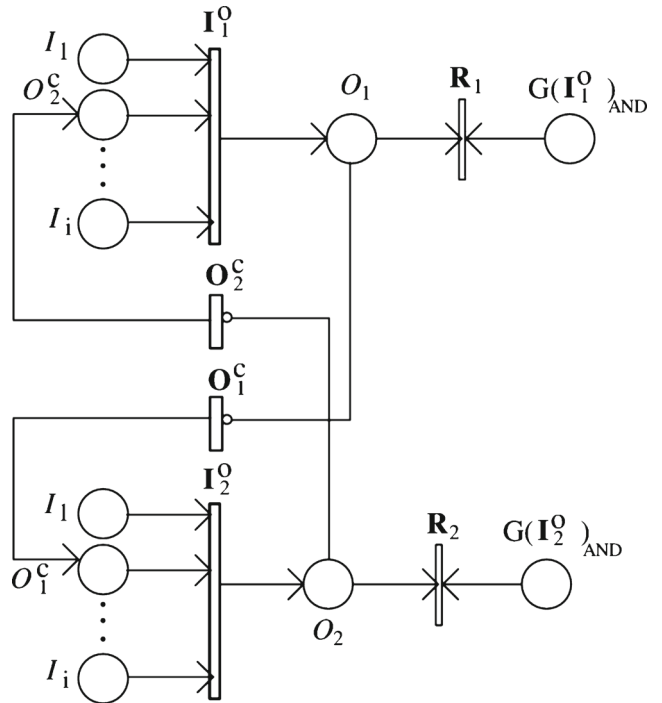


Fig. 12 Logical interlocking contacts

Table 4 Formal definition of *LDPN*

A *LDPN* is a 5-tuple, where:

$P = \{I \cup O \cup B \cup A\}$ is a finite set of places, where:

$I = \{I_1, I_1^{c/o}, I_2, I_2^{c/o}, \dots, I_i, I_i^{c/o}\}$ is a finite set of places representing physical input signals. Those places with the super-index *c* or *o* identifies the places representing NC or NO contacts, respectively.

$O = \{O_1, O_1^{c/o}, O_2, O_2^{c/o}, \dots, O_o, O_o^{c/o}\}$ is a finite set of places representing physical output signals.

$B = \{B_1, B_1^{c/o}, B_2, B_2^{c/o}, \dots, B_b, B_b^{c/o}\}$ is a finite set of places representing internal memory signals.

$A = \{A_1, A_2, \dots, A_a\}$ is a finite set of auxiliary places.

$G = \{G_1, G_2, \dots, G_g\}$ is a finite set of places whose marking depends on Eqs. 5 and 6.

$T = \{I^{c/o} \cup O^{c/o} \cup B^{c/o} \cup R\}$ is a finite set of transitions, where:

$I^{c/o} = \{I_1^{c/o}, I_2^{c/o}, \dots, I_i^{c/o}\}$ is a finite set of transitions representing physical inputs signals. The super-index *c* or *o* identifies the transitions representing NC or NO contacts, respectively.

$O^{c/o} = \{O_1^{c/o}, O_2^{c/o}, \dots, O_o^{c/o}\}$ is a finite set of transitions representing physical outputs signals.

$B^{c/o} = \{B_1^{c/o}, B_2^{c/o}, \dots, B_b^{c/o}\}$ is a finite set of transitions representing internal memory signals.

$R = \{R_1^{c/o}, R_2^{c/o}, \dots, R_r^{c/o}\}$ is a finite set of transitions to reset places.

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,

$W : F \rightarrow \{1\}$ all arcs weights are equal to 1, and

$M_0 = P \rightarrow \{0, 1, 2\}$ initial marking.

I_2 if there is a token in O_1 and there is not a token in I_1 (Eq. 12).

$$M(I_1) = \begin{cases} 1, & \text{if } M(O_1) = 0 \text{ AND } M(I_2) = 0 \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

$$M(I_2) = \begin{cases} 1, & \text{if } M(O_1) = 1 \text{ AND } M(I_1) = 0 \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

5. Logical interlocking.

The PN structure of the Fig. 12 represents the logical interlocking, which is composed of two logical AND and each of them depends on the other one. In this structure, the marking in O_2^c depends directly on the marking in O_2 . This is analogous for O_1^c and OUT_1 . The

transitions O_1^c and O_2^c are enabled only when the corresponding output place O_1 or O_2 does not have a token because of the inhibitor arcs. Thus, the first marked output place disables the other one. The $G(I_1^o)_{AND}$ and $G(I_2^o)_{AND}$ places reset the output places O_1 and O_2 , respectively.

With the previous LD analysis and the five PN logical structures proposed above, a formal definition of *LDPN* is given in next section.

4.3 Definition of *LDPN*

Table 4 shows the elements of a *LDPN*. Places are considered for input and output physical signals, memory internal

Fig. 13 A reversible motor control in LD

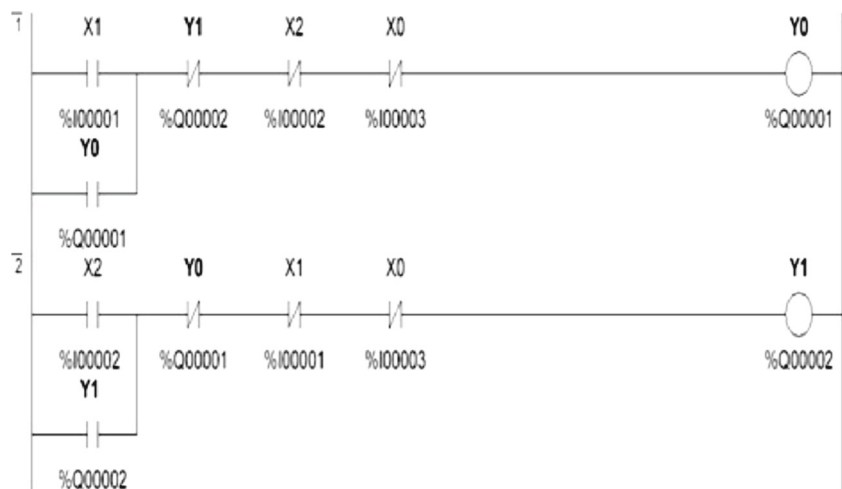
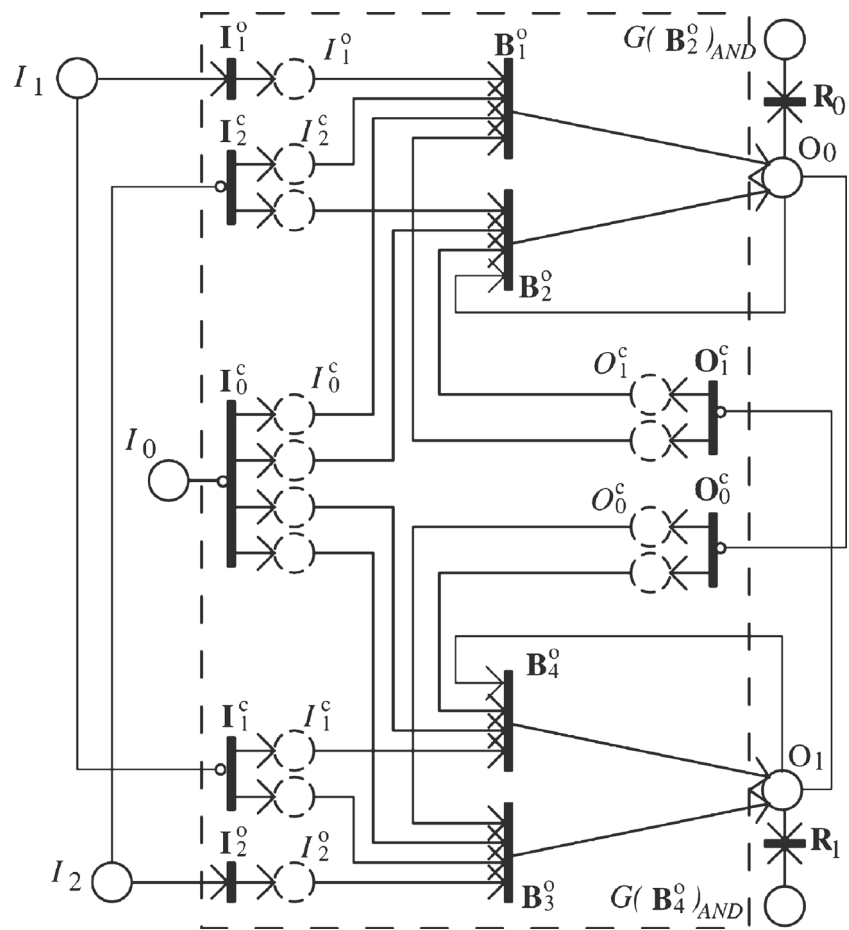


Fig. 14 LDPN of the reversible motor control



signals, and auxiliary places to diversify a signal to its NO and NC contacts if necessary.

The obtained *LDPN* of the control algorithm in LD, graphically, is ordinary because it has the unit weight in all its arcs, and all its places can only have one token for each scan in the PLC. In the incidence matrix, the number of output places for physical input signal transitions correspond to NO and/or NC contacts.

4.4 Methodology to obtain a *LDPN* from a LD

Based on the formal definition of a *LDPN* and the Eqs. 3–9, defined to represent its dynamic behavior, nine steps are proposed to convert a LD into a *LDPN*. The first three steps correspond to the structural representation. The *LDPN* marking represents the energy flow through the contacts and coils of the LD.

Fig. 15 Incidence matrix of the *LDPN* of the reversible motor

$$A_{ij} = \begin{bmatrix} & I_0 & I_0^c & I_1 & I_1^o & I_1^c & I_2 & I_2^o & I_2^c & O_0 & O_0^c & O_1 & O_1^c & G(B_2^o)_{AND} & G(B_4^o)_{AND} \\ I_0^c & -1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ I_1^c & 0 & 0 & -1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ I_2^c & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ O_0^c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \\ O_1^c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 \\ I_1^o & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ I_2^o & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B_1^o & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 0 & 0 \\ B_2^o & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 0 & 0 \\ B_3^o & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ B_4^o & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ R_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ R_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

1. Identify the physical signals of inputs and outputs and internal memory signals.
2. Distribute the physical inputs and outputs from Table 3.
3. Perform the corresponding transformations line-by-line, based on the five PN logical structures.
4. Obtain the initial marking M_0 from the PLC scan.
5. Fire first the transitions of NC contacts and those representing the general protections of the system.

With the initial firing vector specified by the last step, the LDPN is able to simulate the original LD by firing all the enabled transitions. To evaluate the methodology proposed, in the next section, two real control algorithms in LD are analyzed.

5 Case study 1: reversible motor control

From [14], the control algorithm in LD of a reversible motor control is taken, as presented in Fig. 13. X_1 and X_2 are physical input signals (pushbutton) to start the motor, Y_0 and Y_1 are physical output signals for the engine starting with clockwise or counterclockwise rotation, respectively, and X_0 to stop the motor independently of the rotation.

$$\begin{matrix}
 & I_0 & I_0^c & I_1 & I_1^o & I_1^c & I_2 & I_2^o & I_2^c & O_0 & O_0^c & O_1 & O_1^c & G(\mathbf{B}_2^o)_{AND} & G(\mathbf{B}_4^o)_{AND} \\
 M_0 = [& 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 M_1 = [& 0 & 4 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 0 \\
 M_2 = [& 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 M_2 = [& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
 \end{matrix} \tag{13}$$

In order to start the engine to the left side (counterclockwise rotation), a token is needed in place I_0 to enable the transitions $R_0(M_3)$. Consuming the token in place O_0 restores the system conditions. A token in place I_2 allows the starting of the engine.

6 Case study 2: packaging system

The second case study is the packaging system shown in Fig. 16. Its control algorithm involves general protections and the possibility of working the packaging system with a single scale, by turning off the corresponding operating selector. It is also possible to download manually the scales. There are interlocking contacts used in the sequence to determine that the scale to download is the first that has got its corresponding weight.

With the selectors in working position, the operating sequence of the overall system is as follows: both scales

The LD has two control lines with logical AND-OR and an interlocking of the signals Y_0 and Y_1 . Figure 14 shows the LD transformation from Fig. 13, based on the first three steps of the previously described methodology.

The incidence matrix A_{ij} of the net is shown in the Fig. 15.

The control algorithm of the motor only allows the engine to start with a clockwise or counterclockwise rotation. Physical output signal Y_1 or Y_0 locks the control line of the opposite turn engine. So if you require to start the engine in the opposite rotation, it is necessary that the physical output signals Y_0 or Y_1 are de-energized. In the LDPN of the reversible motor control, it needs fire for the transitions R_0 or R_1 to drain the token in O_0 or O_1 , respectively. Thus, the LDPN has the same dynamic behavior that of the original LD of the reversible motor control.

To show the behavior of energize and de-energize action by a coil, consider that it is required to start the engine turning right (mark in I_1). The initial marking M_0 is shown in Eq. 13. All transitions that correspond to the normally closed state are enabled, except I_1^c for the inhibitor arc. Firing these transitions and the transition I_1^o enabled the transitions sequentially B_1^o and B_2^o to kept a mark in the place O_0 . For Eq. 9, the marking in places $O_0^c = 0$, as it is shown in M_2 .

are started to be filled, if the coarse sensor of weight is activated, then the associated piston valve is actuated, reducing the flow of product to the corresponding scale. If the fine sensor of weight is activated, then the associated piston valve is activated, obstructing the flow of the product into the corresponding scale. If the system is in automatic mode and there is a bag in the discharged duct, then the unloading

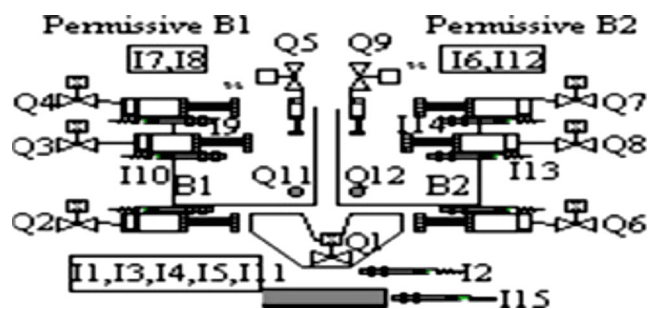


Fig. 16 Packaging system to be modeled with LDPN

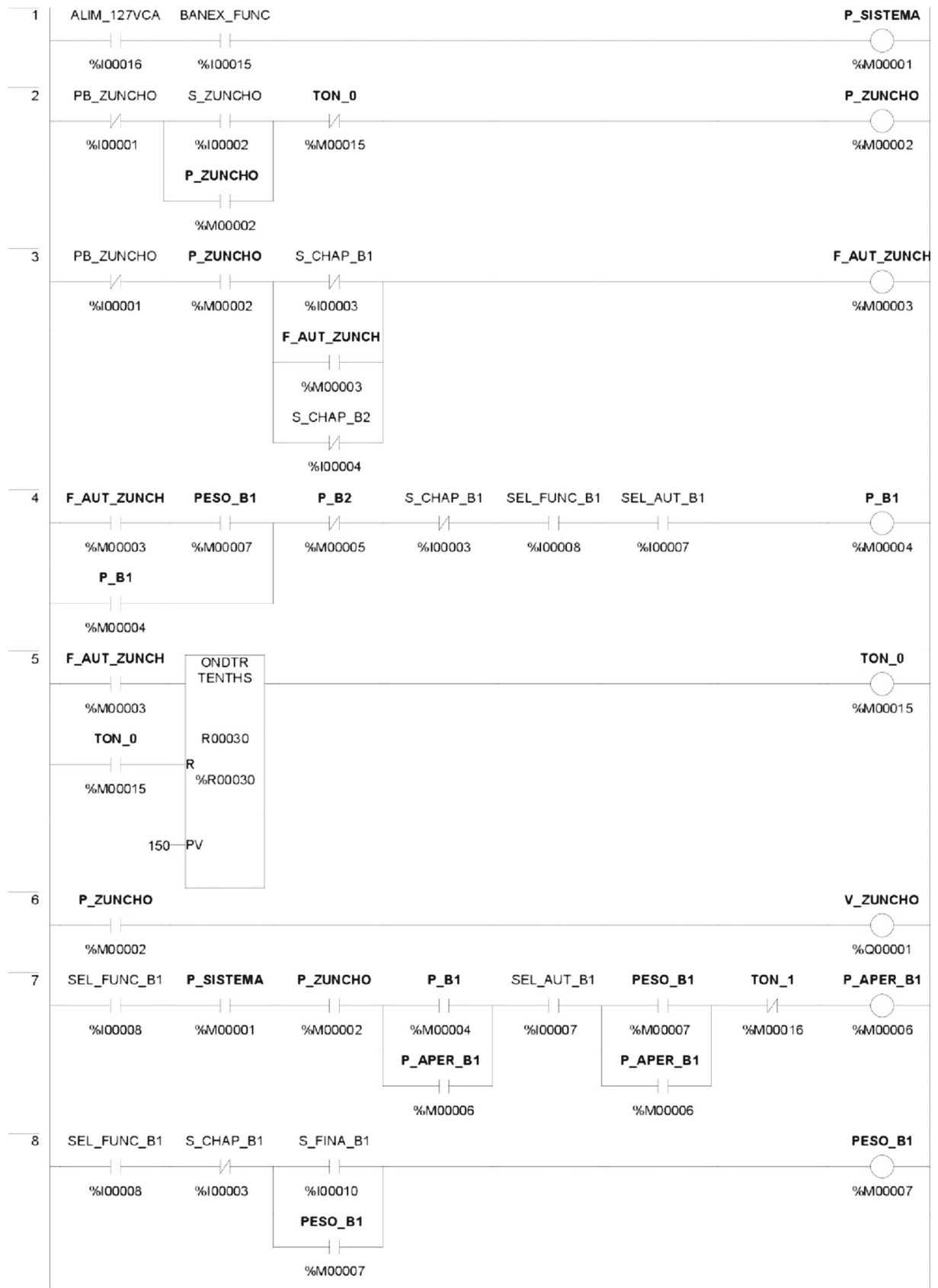


Fig. 17 LD of the packaging system, only one scale, part 1

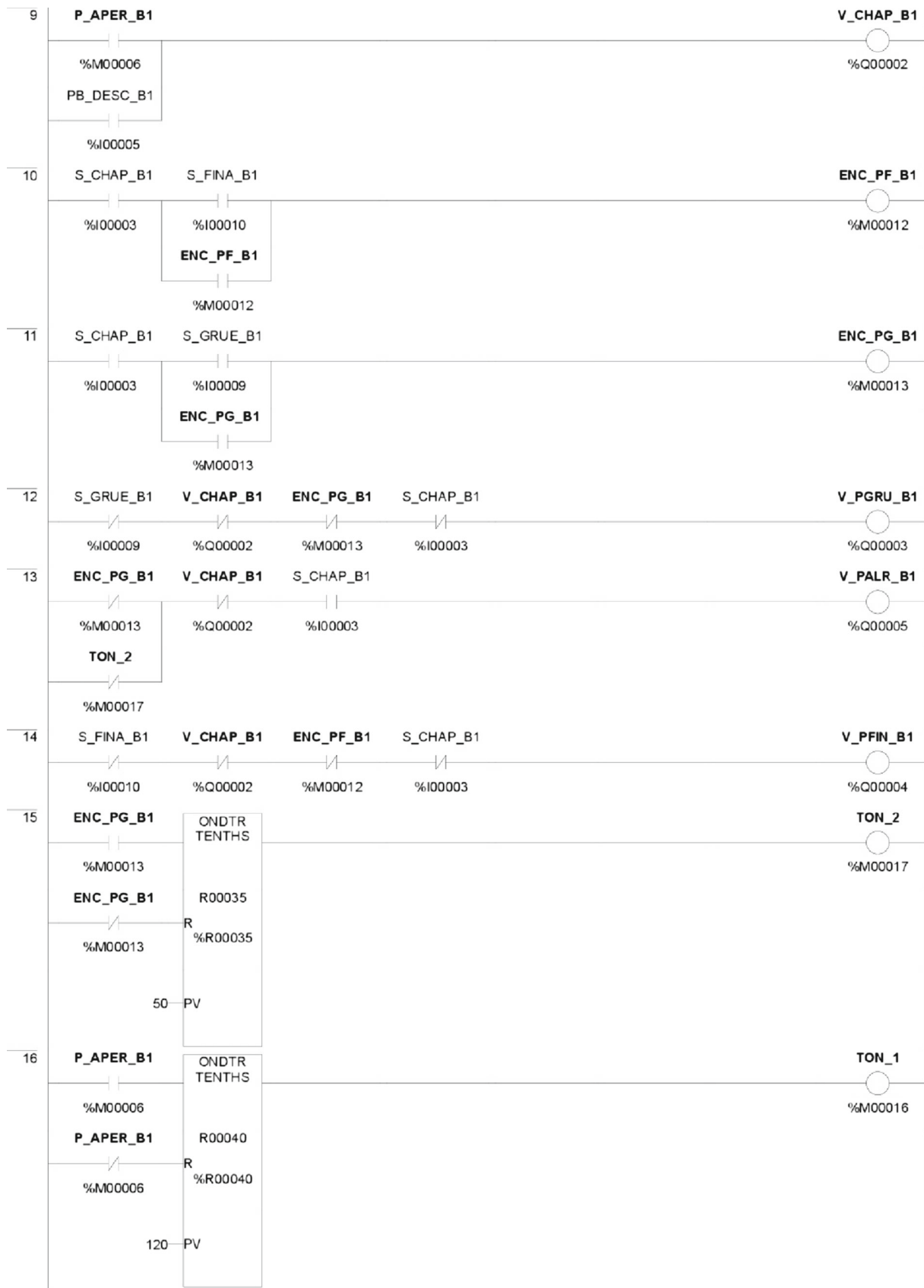


Fig. 18 LD of the packaging system, only one scale, part 2

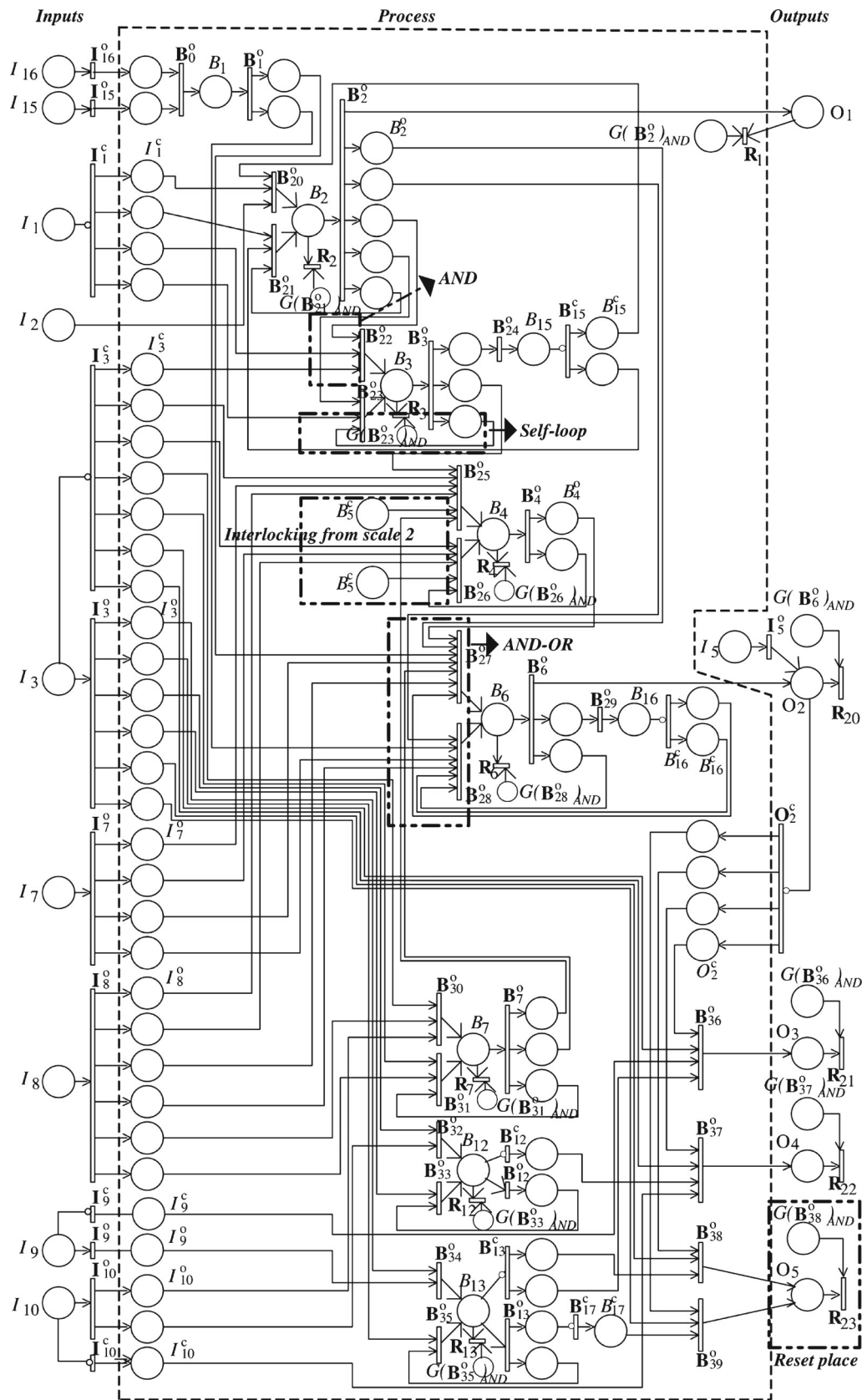


Fig. 19 LDPN of bascule, only one scale

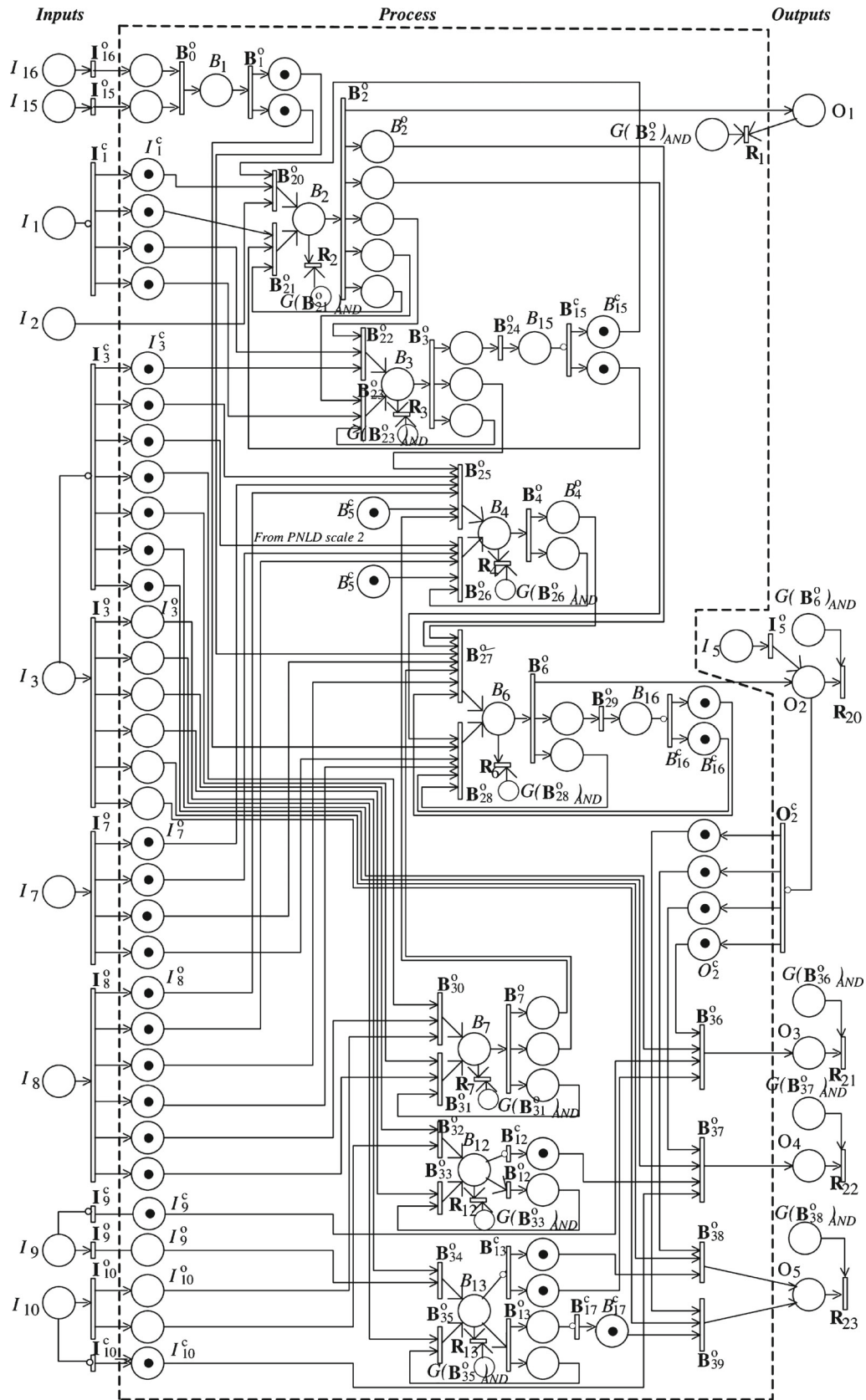
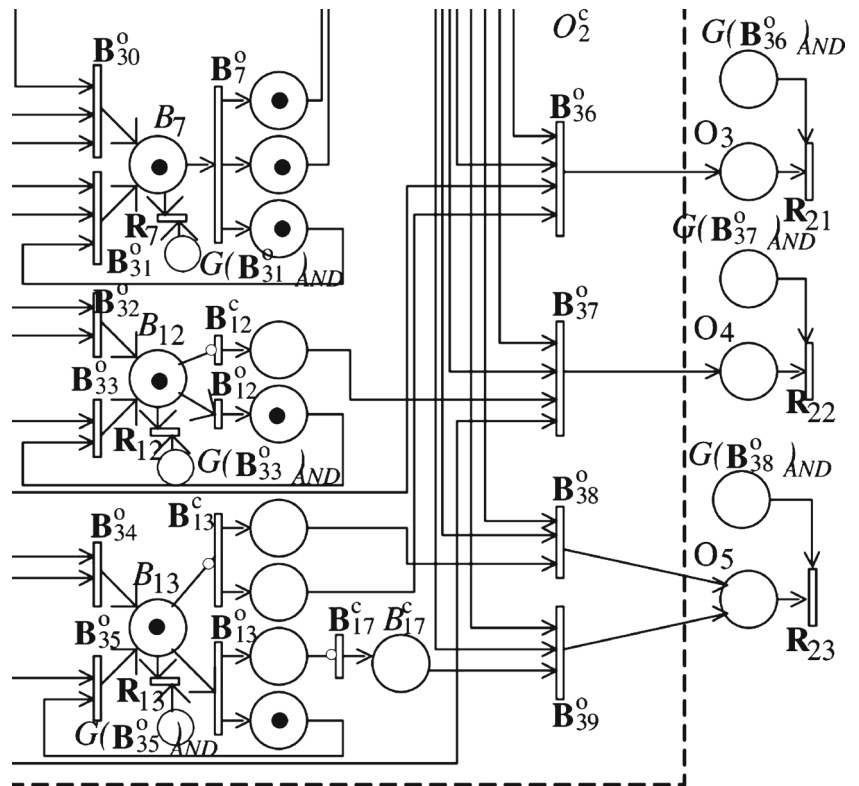


Fig. 20 Marking of safe and operation conditions of the scale LDPN

Fig. 21 Marking of full scale LDPN



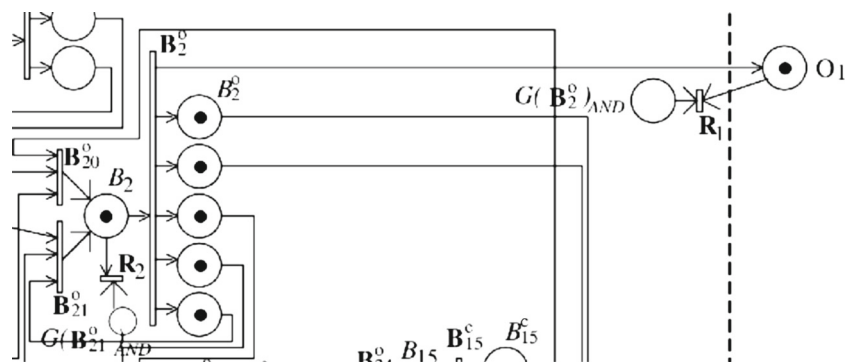
sensor activates the valve of the bottom cover, discharging the scale which was firstly filled. The system works in the same way until there is a change in the operation conditions (an emergency stop, a change in the operation selectors, etc.). The activation sequence for physical inputs and outputs of the packaging system for two scales is described above (Fig. 16):

- a. System selectors in automatic mode and both scales in operation (I6, I7, I8, I12, I15, I16),
- b. Scales bottom cover sensors are deactivated (I3, I4),
- c. Wide and fine weight piston valves are deactivated (Q3, Q4, Q7, Q8),
- d. Piston valves from the scales bottom cover are deactivated (Q2, Q6). Scales filling starts,
- e. Activation of wide weight sensors (I9, I13),
- f. Wide weight piston valves are deactivated (Q3, Q8),
- g. Activation of fine weight sensors (I10, I14),
- h. Fine weight piston valves are deactivated (Q4, Q7),
- i. Activation of the scales unloading sensor (I2),
- j. Sack holder piston valve is activated (Q1),
- k. The bottom cover piston valve of the scales that had been filled first is activated (Q2, Q6),
 - l. Scales high-performance piston valves area activated (Q5, Q9),
- m. Unloading time ends, bottom cover is closed, the process starts again.

The LD of a packaging system with one scale is shown in Figs. 17 and 18.

The LD of a packaging system has several logic structures: AND logic, AND-OR logic, self-loop logic,

Fig. 22 Marking of download scale LDPN



interlocking logic (place $M5$ on control line 4), and timers logic. The last is considered as standard transitions for evaluation of the general behavior of the corresponding $LDPN$. The LD of the packaging system with one scale has 16 control lines, and it can be seen that it is based on the basic control structures studied in this work.

Figure 19 shows the $LDPN$ of packaging systems, for only one scale. $LDPN$ (left side) shows the places representing physical input signals, the right places representing the physical output signals, and the dashed rectangle PN structure representing the LD algorithm control. Importantly, the locations that require modeling the behavior of energize and de-energize or vice versa have R_r transitions with an input place, which is a function of the input signals of the corresponding logical structure.

Due to space limitations, the incidence matrix of 61 columns and 57 rows is not shown, but it is easy to determine it from the corresponding $LDPN$. The dynamic behavior of the scale $PLND$ is explained by showing a part of the marking in the net. The safety signals I_{15} and I_{16} , and operation signals for one scale I_7 and I_8 , are needed for the correct operation of the system. Equation 14 only shows the initial marking of the system protections. Others, the rest of the places, have zero in their markings.

$$M_0 = [\begin{matrix} I_1 & I_2 & I_3 & I_5 & I_7 & I_8 & I_9 & I_{10} & I_{15} & I_{16} & B_5^c & \dots & O_3 & O_4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & \dots & 0 & 0 \end{matrix}] \quad (14)$$

Firings enabled transitions obtained the operate marking, which is shown in Fig. 20. The transitions B_{36}^o and B_{37}^o are enabled, and when they fired, a token is put in places Q_3 and Q_4 . This is the corresponding behavior in the correspondent to energize the coils $Q3$ and $Q4$ in LD to start the filling of the scale. Then, the coils $Q3$ and $Q4$ have initial conditions to be energized.

When the wide sensor I_9 is activated, the transition B_{34}^o is enabled; after firing this, a token is put in place B_{13} , disabling the transition B_{13}^c , hence, a token is put in place $G(B_{36}^o)_{AND}$ to drain the token from place O_3 , which is equivalent to de-energize the coil $Q3$ in scale LD. Likewise, when the scale has the requested weight, the fine sensor I_{10} is activated; the transitions B_{32}^o is enabled, then a token is put in place B_{12} , disabling the transition B_{12}^c . Therefore, a token is put in place $G(B_{37}^o)_{AND}$ to drain the token from place O_4 , which is equivalent to de-energize the coil $Q4$ in scale LD. In addition, the transition B_{30}^o is enabled to put tokens in places B_7^o , which means that those places have conditions of full scale. Only the part of this marking is show in Fig. 21. In order to add a token to places $G(B_{31}^o)_{AND}$, $G(B_{33}^o)_{AND}$, and $G(B_{35}^o)_{AND}$ and consume the tokens in places B_7 , B_{12} and B_{13} , a token in place I_3 is necessary.

Now, unloading sensor is activated; a token is put in place I_2 to enable the transition B_{20}^o and assigns tokens to places B_2^o and O_1 as shown in Fig. 22. Tokens in places B_2 and O_1 depend on $G(B_{21}^o)_{AND}$ and $G(B_2^o)_{AND}$, respectively. Under these conditions and with a token in the place B_5^c , a token can be obtained in place O_2 , which is equivalent to unload the scale.

When the scale download starts, bottom cover sensor I_3 is activated; when the scale download finishes and the bottom cover closes, sensor $I3$ is deactivated and end the download of scale to deactivate. In the scale $PNLD$, tokens in places I_3^c are applied to enable transitions and then a token is sent to O_2 . Tokens in places I_3^o activate the marking in places $G(B_{23}^o)_{AND}$, $G(B_{26}^o)_{AND}$, $G(B_{28}^o)_{AND}$, $G(B_6^o)_{AND}$, $G(B_{31}^o)_{AND}$, $G(B_{33}^o)_{AND}$, and $G(B_{35}^o)_{AND}$, restoring the system condition, which is equivalent to an empty scale.

7 Conclusions and future work

PN theory has the advantage of graphically showing models of DES, and it allows the mathematical analysis of dynamic behavior of the respective control algorithms. Thus, the proposed methodology for transforming a LD to PN model performed line-by-line from a control program guarantees the equivalence between the two methods (LD and PN) in both structure and general behavior. The dynamic marking depends on the inputs of each logical structure proposed, and it allows the modeling of energizing and de-energizing coils as part of the system behavior. Furthermore, the $LDPN$ allows the application of formal PN theory in order to simulate, verify, and validate control algorithms implemented in any industrial process or those that had not been implemented yet, but it is necessary to analyze them before their implementation has marked dynamic function of their own inputs in each logical structure, allowing the modeling of the behavior of energize and/or de-energize coils as part of behavior systems and permitting the application to $LDPN$ - formalizing-theories or methods for the corresponding simulation, verification, and validation of the control algorithms having been implemented or to be developed for some DES.

The methodology was formulated in a general way; however, there exist elements of the transformed $LDPN$ which could be reduced, and hence its graphical complexity could be reduced too, as in the case when a signal has only one contact.

Future work includes carrying out the study case of a control algorithm in LD with a greater number of input and output signals, therefore with more control lines; the definition of new transforming blocks for more control logics; and a proposal to perform an analysis by parcels of the $LDPN$ for larger control algorithms.

References

1. International Electrotechnical Commission, IEC 61131-3 (2003) Programmable Controllers: Programming Languages, International standard, segunda edición
2. Murata T (1989) Petri nets: properties, analysis and applications. *Proc IEEE* 77(4):541,580
3. Lee J, Lee JS (2009) Conversion of ladder diagram to petri net using module synthesis technique. *Int J Model Simul* 29(1):79–88
4. Thapa D, Dangol S, Wang G-N (2005) Transformation from petri nets model to programmable logic controller using one-to-one mapping technique. In: International conference on computational intelligence for modelling, Control and automation, 2005 and international conference on intelligent agents, Web technologies and internet commerce, vol 2, pp 228–233, 28–30
5. Peng SS, Zhou M-C (2004) Ladder diagram and Petri-net-based discrete-event control design methods. *IEEE Trans Syst Man Cybern Part C Appl Rev* 34(4):523,531
6. Lee J.-S., Hsu P.-L. (2004) An improved evaluation of ladder logic diagrams and Petri nets for the sequence controller design in manufacturing systems. *Int J Adv Manuf Technol* 24:279–287
7. Zapata G, Carrasco E (2002) Estructuras Generalizadas para Controladores Lógicos Modeladas mediante Redes de Petri. *Revista Dyn* 135:65–74
8. Tzafestas SG, Pantelelis MG, Kostis DL (2002) Design and implementation of a logic controller using Petri nets and ladder logic diagram. *Syst Anal Model Simul Taylor Fancis* 42(1):135–167
9. Korotkin S, Zaidner G, Cohen B, Ellenbogen A, Arad M, Cohen Y (2010) A Petri net formal design methodology for discrete-event control of industrial automated systems. In: 2010 IEEE 26th Convention of electrical and electronics engineers in israel (IEEEI), vol 17–20, pp 431–435
10. Valentin N, Mircea P, Ioan D, Aurel I (2011) Designing the control of an electrical drive using Petri nets. 7th International symposium on advanced topics in electrical engineering (ATEE)
11. da Silva Oliveira EA, da Silva LD, Gorgonio K, Perkusich A, Martins AF (2011) Obtaining formal models from ladder diagrams. In: 2011 9th IEEE international conference on industrial informatics (INDIN), vol 26–29, pp 796–801
12. Xuekum C, Lilian L, Pengfei Q (2012) Method for translating ladder diagram to ordinary Petri nets. 51st IEEE Conference on Decision and Control
13. International Electrotechnical Commission, IEC 61131-8 (2003): Programmable controllers: guidelines for the application and implementation of programming languages, International standard, segunda edición
14. Zhang H, Jiang Y, Hung WN, Yang G, Gu M, Sun J (2012) New strategies for reliability analysis of programmable logic controllers. *Math Comput Model* 55(7/8):1916–1931