Bridging the Gap Between Design and Implementation of Discrete-Event Controllers

Marcos Vicente Moreira and João Carlos Basilio, *Member, IEEE*

***Abstract*—Extended labeled Petri nets (ELPNs), i.e., labeled Petri nets with inhibitor arcs, are usually used to model the desired closed-loop behavior of a controlled discrete-event system, and, as such, their states are formed with both the controller and the plant states. However, the control logic is based on the controller states only and the interaction between controller and plant is carried out through sensor readings from the plant and control actions (forced events) from the controller. This makes ELPN not suitable for modeling the controller. Control interpreted Petri nets (CIPNs), on the other hand, include control actions in the places and sensor readings in the transitions as part of their formal structure, and so provide a better formalism for controller modeling. In this paper, we propose a two-step approach to discrete-event controller implementation, as follows: (i) we first propose a set of transformation rules to convert the initial ELPN to an equivalent CIPN, therefore extracting the control logic from the desired closed-loop behavior and (ii) we present a straightforward systematic way to translate the CIPN into a ladder diagram. We apply the results presented here to the implementation of the automation system of a plastic molding machine.**

***Note to Practitioners*—The usual approach to the design of manufacturing systems is to model the desired closed-loop system behavior using, for example, Petri nets, and in the sequel to extract the discrete-event controller from this model with the view to implementing the control logic on a Programmable Logic Controller (PLC). So far, the controller extraction is carried out in an ad hoc and intuitive way. In order to overcome the lack of formal methods to deal with controller extraction and implementation, we propose here a set of rules to construct, in a systematic way, the ladder diagram that implements the discrete-event controller. We illustrate the conversion technique proposed here by applying it to the implementation of the automation system of a plastic molding machine.**

***Index Terms*—Discrete-event systems (DESs), ladder diagram, manufacturing systems, Petri nets, programmable logic controller (PLC).**

I. Introduction

PETRI NET [1]–[3] is one of the formalisms that is suitable to model and visualize the behavior of discrete-event systems (DESs). They have recently been applied successfully to several problems ranging from manufacturing systems to web applications [4]–[6]. Among the advantages of Petri nets we may list their capacity to model behaviors such as concurrency, synchronization, and resource sharing. Other advantages of Petri nets include the possibility to describe both the desired closed-loop behavior of controlled DES and its discrete-event controller (DEC).

Several methods have been proposed to synthesize DECs. Holloway *et al.* [7] distinguishes three main approaches: (i) the control theoretic approach [8]–[10]; (ii) the logic controller approach [11]–[13]; and (iii) the controlled behavior approach [14]–[16]. In the logic controller approach, the objective is to directly design a controller defining its input–output behavior, in order to achieve the desired controlled behavior for the closed-loop system. This method can be applied to the control of simple processes, and simulation is necessary for the validation of the closed-loop behavior. The controlled behavior approach, on the other hand, is preferable for the design of complex manufacturing systems and consists of modeling the desired closed-loop system behavior, namely, the joint model of the plant and controller, and then to extract the DEC for implementation. This strategy allows properties of interest such as liveness, boundedness, and reversibility to be guaranteed or even previously analyzed using the desired closed-loop model.

In using the controlled behavior approach, the desired closed-loop behavior of the controlled DES is usually modeled with extended labeled Petri nets (ELPNs), i.e., labeled Petri nets with inhibitor arcs, and, as such, their states are formed with both the controller and the plant states. However, the control logic is based on the controller states only and the interaction between controller and plant is carried out through sensor readings from the plant and control actions (forced events) from the controller. This makes ELPN not suitable for modeling the controller. Control interpreted Petri nets (CIPNs) [17]–[20], on the other hand, include control actions in the places and sensor readings in the transitions as part of their formal structure, and so provide a better formalism for controller modeling.

After a CIPN, that models the controller, has been obtained from the ELPN that models the desired closed-loop behavior of the DES, the next step is to generate a code for implementation on a programmable logic controller (PLC) by converting the control logic described by the CIPN into a PLC programming language, e.g., Sequential Function Charts (SFC) [21], [22] or Ladder diagrams [18], [20], [23]–[25].

In [20], a comprehensive survey about methods for the design and implementation of DECs using ladder diagrams and Petri nets is presented. Some of the techniques reviewed in [20] consist of designing and converting Petri nets (or extensions of Petri nets) to ladder diagrams for PLC implementation. How-

Manuscript received February 09, 2013; revised May 27, 2013; accepted August 17, 2013. Date of publication October 04, 2013; date of current version January 01, 2014. This paper was recommended for publication by Associate Editor C. N. Hadjicostis and Editor M. C. Zhou upon evaluation of the reviewers' comments. The work of J. C. Basilio was supported in part by the Brazilian Research Council (CNPq), under Grant 306592/2010-0.

The authors are with the COPPE-Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, 21949-900, Rio de Janeiro, RJ, Brazil (e-mail: moreira@dee.ufrj.br; basilio@dee.ufrj.br).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2013.2281733

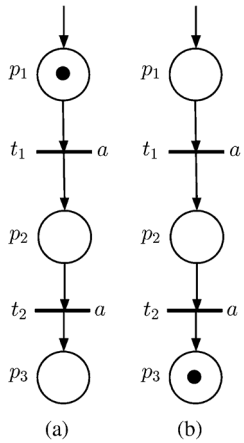

Fig. 1. (a) CIPN where only transition t_1 is enabled and (b) incorrect state reached after the occurrence of event a when the avalanche effect occurs in the controller code.

ever, there is an important problem that may appear in the implementation of controllers in ladder diagrams: the avalanche effect. The avalanche effect occurs in the PLC implementation of the Petri net controller when the receptivities of two or more consecutive transitions become true at the same scan cycle and, in the controller code, transitions that were not enabled are incorrectly transposed. For example, consider the Petri net of Fig. 1(a). In this case, if event a occurs, only transition t_1 must fire since only t_1 is enabled. Transition t_2 will fire only after the second occurrence of event a. However, depending on the programming code, both transitions can be transposed in the same scan cycle, leading to the state of the Petri net of Fig. 1(b), which is not the intended behavior of the controller.

The earliest conversion methods were based on the so-called Token Passing Logic (TPL) technique [18], [23], whose idea was to use the evolution of the tokens through the Petri net as the main mechanism for controlling the flow of the control logic. Although the methods proposed in these papers were successfully implemented on manufacturing systems, the avalanche effect was not explicitly considered. The avalanche effect was addressed for the first time in [21], who proposed a systematic procedure for avoiding it in ladder implementations of supervisory control systems modeled using automata. The only drawback of the approach presented in [21] is the lack of a formal method to deal with cycles, namely, where to break the loop in an intelligent way for implementation purpose.

Jimenez *et al.* [26] consider the conversion of timed interpreted Petri nets into ladder diagrams. In the method proposed in [26], each transition in the net is translated into a network in the ladder diagram. Differently from [18] and [23], the method presented in [26] cannot be applied to extended Petri nets. It is worth mentioning that the avalanche effect was not addressed in [26] either.

In [24], a different approach was proposed as follows: first, Boolean expressions are obtained from the CIPN, and then, these expressions are straightforwardly converted into a ladder code. The weakness of this technique is that the resulting ladder diagram does not provide a simple visualization of the control code, and thus, simple modifications in the control logic cannot be easily implemented in the existing ladder diagram.

In [27], the implementation on PLCs of Ramadge–Wonham (RW) supervisors [28] with time delay functions is considered and a method, based on the TPL technique, is proposed for obtaining a ladder diagram from the supervisor. The avalanche effect and other problems associated with the implementation of supervisors on PLC have not been addressed.

In [29], a hierarchical control of DESs is proposed. In a lower level, a DEC is designed to force control events to occur and, in a higher level, a supervisor prohibits the occurrence of some events such that the closed-loop system satisfies the specifications given by the designer. The main advantage of this hierarchical frame is that the control and supervision tasks are clearly separated. This strategy has been named as the supervised control of DESs. The supervision and control are both designed in [29] by using GRAFCET.

In [30], the implementation of the supervised control in structured text (ST)—one of the five PLC languages defined in the standard IEC 61131-3 [31]—is addressed. First, the system is partitioned into several subsystems, each one having a DEC that performs only simple sequences, with no resource sharing; in this approach, the control logic must be very simple in such a way that the controller can be directly implemented on a PLC. Then, the coordination of the subsystems is managed by a supervisor designed by using the RW supervisory control theory. An evolution algorithm for the implementation of supervisors, and a set of rules for the translation of the supervisor described by automata or Petri nets into an ST code were proposed in [30]. Although the algorithm proposed in [30] avoids the avalanche effect, the method does not approach the implementation of complex DECs with conflicts and resource sharing, and also do not consider DEC modeled by CIPNs and timed Petri nets. More recently, it has been shown in [32] that complex distributed control systems can be implemented by using IEC 61131 languages extended with object oriented programming.

The avalanche effect was, again, considered in [25], for complex DEC modeled by timed CIPN. A conversion technique that establishes transformation rules from CIPN to ladder diagrams which preserve the structure of the Petri net and also avoids the avalanche effect were given in [25].

We assume here that the controlled behavior approach is the methodology used to synthesize the DEC. Thus, starting from the ELPN model of the desired closed-loop behavior, we will propose a two-step approach to the implementation of the DEC, as follows: (i) first, a set of transformation rules is used to convert the initial ELPN to an equivalent CIPN, therefore extracting the control logic from the desired closed-loop behavior and (ii) a straightforward systematic way, which is an extension of that proposed in [25], is given to translate the CIPN into a ladder diagram. The translation rules make use of the structure and dynamics of the Petri net to obtain conditions for the firing of transitions and to describe the flow of the control logic. The ladder diagram is built in such a way that the avalanche effect is avoided and its structure allows any modification in the DEC modeled by the CIPN to be easily implemented in the existing ladder diagram. It is worth remarking that, although several works address the problem of modeling closed-loop systems by using labeled Petri nets, from the authors' knowledge, the problem of extracting a controller from this Petri net model using CIPN has not been addressed in the literature.

We present in Section II a brief review of Petri nets and in Section III we first introduce the definition of ELPNs, and, in the sequel, we define CIPNs. We propose in Section IV a set of transformation rules for obtaining a CIPN from an ELPN, and, in Section V, we introduce the method for obtaining the ladder diagram from the CIPN. We illustrate the methodology proposed here with the design of an automation subsystem of a plastic injection molding machine. We present concluding remarks in Section VI.

II. Preliminaries

A Petri net graph is a bipartite graph that contains two types of nodes: places and transitions. The places are represented by circles and the transitions by bars, and these two types of nodes are connected through directed arcs. The formal definition of a Petri net graph is as follows.

Definition 1: A Petri net graph is a weighted bipartite graph $(P, T, Pre, Post)$, where P is the finite set of places, T is the finite set of transitions, $Pre : (P \times T) \to \mathbb{N}$ is the function of ordinary arcs that connect places to transitions, $Post : (T \times P) \to \mathbb{N}$ is the function of ordinary arcs that connect transitions to places.

The set of input (output) places of a transition $t_j \in T$ is denoted as $I(t_j)(O(t_j))$, and is formed with places $p_i \in P$ such that $Pre(p_i, t_j) > 0 (Post(t_j, p_i) > 0)$.

Let $x : P \to \mathbb{N}$ denote the marking function. Then, $x(p_i)$ represents the number of tokens assigned to place p_i. A marking of a Petri net is the column vector $\underline{x} = [x(p_1)x(p_2)\dots x(p_n)]^t$ formed with the number of tokens in each place p_i, for $i = 1, \dots, n$, where n is the cardinality of P.

Definition 2: A Petri net N is a five-tuple $N = (P, T, Pre, Post, x_0)$, where $(P, T, Pre, Post)$ is, in accordance with Definition 1, a Petri net graph, and $x_0 : P \to \mathbb{N}$ is the initial marking function.

A transition $t_j \in T$ is said to be enabled when the number of tokens in each one of its input places is greater than or equal to the weight of the arcs connecting the places to transition t_j, i.e., t_j is enabled if and only if

$$x(p_i) \geq Pre(p_i, t_j), \quad \text{for all } p_i \in I(t_j). \tag{1}$$

If transition t_j is enabled for a marking $\underline{x}$ and t_j fires, then a new marking $\bar{\underline{x}}$ is reached. The evolution of the markings is given by the following equation:

$$\bar{x}(p_i) = x(p_i) - Pre(p_i, t_j) + Post(t_j, p_i), \quad i = 1, \dots, n. \tag{2}$$

Finally, we say that a place $p_i \in P$ is a safe place if $x(p_i) \leq 1$ for all markings of the Petri net reachable from $\underline{x}_0$.

In order to use the Petri net formalism to model physical systems (e.g., manufacturing systems), it is necessary to label the transitions with events from an event set E. This leads to the definition of labeled Petri nets, as follows [33].

Definition 3: A labeled Petri net (LPN) is a seven-tuple $N = (P, T, Pre, Post, E, l, x_0)$, where $(P, T, Pre, Post)$, is according to Definition 1, a Petri net graph, E is the set of events for transition labeling, $l : T \to E$ is the transition labeling function, and x_0 is the initial marking function of the system.

Now, let $O(p_i)$ be the set of output transitions of a place $p_i \in P$. When p_i is an input place of two or more transitions, we have a conflict. Three types of conflicts can be identified, as follows:

- *Structural conflict*. A structural conflict, denoted by $(p_i, O(p_i))$, exists when the cardinality of $O(p_i)$ is greater than one. Notice that the existence of a structural conflict is independent of the Petri net marking.
- *Effective conflict*. An effective conflict depends on the marking of the Petri net [3]. An effective conflict, denoted by $(p_i, O(p_i), \underline{x})$ is formed by a structural conflict and a marking $\underline{x}$, such that a subset of $O(p_i)$ with at least two transitions are enabled by $\underline{x}$ and the number of tokens in p_i is smaller than the sum of the weights of all arcs connecting place p_i to the enabled transitions of $O(p_i)$.
- *Actual conflict*. An actual conflict is an effective conflict $(p_i, O(p_i), \underline{x})$ where the events associated with two or more enabled transitions of $O(p_i)$ occur simultaneously. In the theory of supervisory control this case cannot happen since it is assumed that two distinct events never occur simultaneously. This is actually true when the events are independent [3]. However, PLCs update their inputs synchronously and even when the events are independent, they may be seen by the PLC as occurring at the same time [21].

The resolution of conflicts is not a trivial task in the design of DES and is beyond the scope of this paper. However, since we assume that the DEC synthesis is carried out by using the controlled behavior approach, it is very reasonable to assume that all possible conflicts have already been solved by the designer. We will therefore make the following assumption.

A1) All the actual conflicts have been solved by the designer before the implementation of the controller, i.e., in the construction of the ELPN.

III. Extended Labeled and Control Interpreted Petri Nets

In this section, we will extend the definition of LPN, which will be used to model the desired behavior of a closed-loop DES, and, in the sequel, we will briefly review the concepts of CIPN, which will be used to describe the controller behavior. The two formalisms are related/interconnected in accordance with the block diagram shown in Fig. 2.

A. Extended Labeled Petri Nets (ELPNs)

The definition of ELPNs is as follows.

Definition 4: An ELPN is an eight-tuple

$$M = (P, T, Pre, Post, E, l, x_0, In) \tag{3}$$

where $(P, T, Pre, Post, E, l, x_0)$ is, according to Definition 3, a labeled Petri net, and $In : (P \times T) \to \mathbb{N}$ is the inhibitor arc function.

The inhibitor arc function allows a larger class of DES to be represented, increasing the modeling power of the labeled Petri net. It can be used to eliminate conflicts since, if there is a inhibitor arc connecting a place p_i to a transition t_j, then transition t_j will be disabled if the number of tokens in p_i is greater than or equal to the weight of the arc that connects p_i

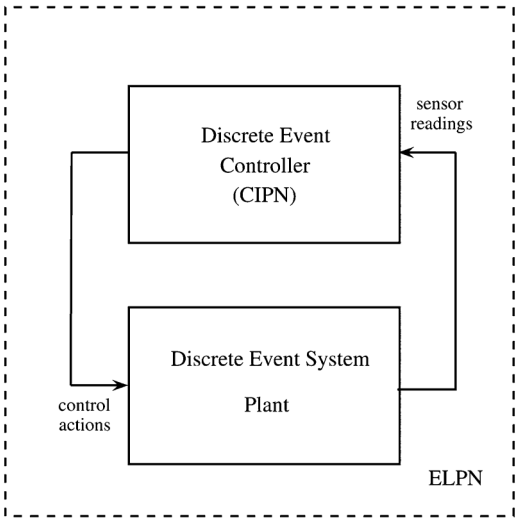

Fig. 2. Closed-loop system.

to t_j, i.e., $x(p_i) \geq In(p_i, t_j)$. The inhibitor arc is graphically represented by a solid line terminating with a small circle.

The desired closed-loop behavior of a large class of DES, mainly manufacturing systems, can be modeled by the ELPN introduced in Definition 4; for example, all Petri net models for manufacturing systems presented in [14] can be described by a labeled Petri net; it is worth remarking that all conflicts are avoided in [14] with the so-called choice-synchronization structure, which is a particular case of firing priority attribution. In [2] and [3], several DES modeled by ELPNs are presented.

We say that a transition of an ELPN is observable if one of the following conditions associated with the labeling event holds true: (i) the event has a sensor to indicate its occurrence; (ii) the event is a command signal sent by the controller to the plant; (iii) the event is associated with a time delay; and (iv) the event is associated with the synchronization of internal conditions observable by the controller; for instance, the synchronization in the end of two concurrent processes that is verified by the controller to command the start of a third process. We will suppose in this paper that all transitions of the ELPN are observable by the controller.

B. Control Interpreted Petri Nets (CIPNs)

The CIPN has additional structures to deal with sensors and actuators. The inputs of the CIPN, associated with transitions, are signals sent from sensors to inform the occurrence of events, and the outputs, associated with places, are impulse actions commanded to the plant. In order to include timers, the CIPN is also T-timed. Therefore, the set of transitions of the CIPN, T_C, can be partitioned as $T_C = T_C^0 \dot{\cup} T_C^D$, where T_C^0 is the set of transitions with no firing delays and T_C^D is the set of timed transitions.

Definition 5: A CIPN is a 13-tuple

$$N_C = (P_C, T_C, Pre_C, Post_C, x_{0,C}, In_C, C, E_C, l_C, D, l_D, A, l_A) \tag{4}$$

where $(P_C, T_C, Pre_C, Post_C, x_{0,C}, In_C)$ is an extended Petri net, C and E_C are the sets of conditions and input events associated with the transitions in T_C^0, respectively, $l_C : T_C^0 \rightarrow C \times E_C$ is the function that associates to each transition in T_C^0 an event from E_C and a condition from C, D denotes the set of delays associated with timed transitions, $l_D : T_C^D \rightarrow D$ is the timing function that associates to each timed transition a delay from D, A is the set of impulse actions, associated with safe places in P_C, and $l_A : P_{C_S} \rightarrow 2^A$ is the action assigning function, where $P_{C_S} \subseteq P_C$ denotes the set of safe places.

Notice in Definition 5 that it is assumed that every transition $t_j \in T_C^0$ is associated with a condition in C and an event in E_C. Transition t_j may fire when it is enabled and the associated condition is true, and t_j fires when the associated event occurs. If the condition associated with t_j is not explicitly specified, then the condition is equal to one, i.e., the logical condition is true. Moreover, if the input event is not explicitly specified, then the event is equal to λ, the always occurring event [3], to indicate that transition t_j must fire immediately after it becomes enabled if the associated condition c_j is true. Therefore, all transitions in T_C^0 have Boolean expressions associated with them.

According to Definition 5, impulse actions are assigned to safe places in P_{C_S}. If a place $p_{C_i} \in P_{C_S}$ does not have an action associated with it, then $l_A(p_{C_i}) = \emptyset$, and if a place p_{C_i} with assigned actions is marked, then the impulse actions $l_A(p_{C_i})$ are executed when place p_{C_i} changes its marking from zero to one. It is important to remark that the impulse action must be executed even if the marking of place p_{C_i} is unstable [3], i.e., it changes from zero to one and then back to zero instantaneously. Thus, the execution of the impulse actions are directly related to changes in the markings of the safe places $p_{C_i} \in P_{C_S}$ where $l_A(p_{C_i}) \neq \emptyset$. The reason to consider only impulse actions comes from the fact that the CIPN is obtained from a labeled Petri net—all transitions are labeled with events whose occurrence are instantaneous. Thus, the controller command events, associated with the transitions in the labeled Petri net, can be seen as impulse actions. In order to consider also level actions, modifications should be applied to the final CIPN obtained by using the method proposed in this paper. These modifications will not be addressed here.

In the following section, we show that it is always possible to extract a DEC, described by a CIPN satisfying Definition 5, from a closed-loop system modeled by an ELPN whose transitions are all observable by the controller.

IV. CIPN Extraction From an ELPN

We will now propose transformation rules to change ELPNs to CIPNs. The transformation will be carried out in two steps, as follows: first, transformations will be defined in such a way that the conditions imposed by Definition 5 are satisfied, leading therefore to a CIPN; then, the resulting CIPN will be modified so as to reduce the number of places and transitions.

A. Transformation of an ELPN to a CIPN

Let M denote an ELPN defined according to Definition 4. Partition the event set of M as

$$E = E_A \dot{\cup} E_T \dot{\cup} E_S \dot{\cup} E_I$$

where E_A denotes the set of actions, E_T the set of events associated with time delays, E_S the set of events observed by sensors, and E_I the set of events associated with the synchronization of

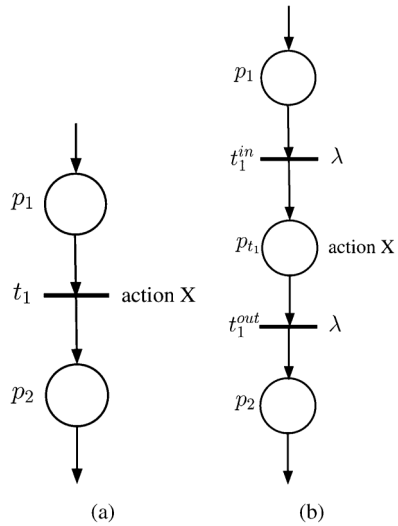

Fig. 3. (a) Transition labeled with a command event in an ELPN and (b) its expansion in a CIPN according to transformation rule T3.

internal conditions observable by the controller. A CIPN N_C equivalent to the given ELPN is formed as follows:

T1) Keep the set of places of M in N_C, i.e., $P \subset P_C$.

T2) Keep the transitions $t_j \in T$ labeled with events in the set $E \setminus E_A$ in N_C.
   (a) For a transition t_j such that $l(t_j) \in E_S$, synchronizes the corresponding transition $t_j \in T_C$ with the rising edge or the falling edge of the sensor signal.
   (b) For a transition t_j such that $l(t_j) \in E_I$, synchronizes the corresponding transition $t_j \in T_C$ with the always occurring event λ.
   (c) For a transition t_j such that $l(t_j) \in E_T$, create a timed transition $t_j \in T_C^D$ with the same delay time as described by the event of E_T.
   (d) For each transition $t_j \in T_C$ created in steps (a)–(c), define $Pre_C(p_i, t_j) = Pre(p_i, t_j)$, $Post_C(t_j, p_i) = Post(t_j, p_i)$, and $In_C(p_i, t_j) = In(p_i, t_j)$, for all $p_i \in P$.

T3) For a transition t_j such that $l(t_j) \in E_A$, create two transitions $t_j^{in}, t_j^{out} \in T_C$, both synchronized with λ, and a safe place p_{t_j} such that $x_{0,C}(p_{t_j}) = 0$, and define $Pre_C(p_i, t_j^{in}) = Pre(p_i, t_j)$, $Post_C(t_j^{out}, p_i) = Post(t_j, p_i)$, $In_C(p_i, t_j^{in}) = In(p_i, t_j)$, for all $p_i \in P$, and $Post_C(t_j^{in}, p_{t_j}) = Pre_C(p_{t_j}, t_j^{out}) = 1$. In addition, convert event $l(t_j) \in E_A$ to an impulse action associated with place p_{t_j}. Fig. 3 illustrates transformation T3.

For obvious reasons, the CIPN obtained according to transformation rules T1–T3 will be referred here to as direct control interpreted Petri net (DCIPN).

B. Reducing the Size of CIPN

The DCIPN obtained from the ELPN in the previous subsection is likely to have redundant places and transitions. Since controller implementation depends on the number of places and transitions, it would be worthwhile to carry out additional transformations in the DCIPN so as to reduce the cardinality of the place and transition sets. The Reduction rules proposed here are based on the reduction techniques used to mitigate the effort in the verification of Petri net properties [15], [34]–[37]; the difference is that here we are interested in guaranteeing that the discrete-event closed-loop system executes the correct sequence of events described by the ELPN. As in the reduction methods used to verify Petri net properties, the idea behind the reduction technique presented in this paper is to examine the Petri net structure and/or behavior and to apply appropriate Reduction rules to reduce the net size as much as possible [15]. It is worth mentioning that the application of a rule may lead to a modified CIPN that enables the application of another rule that was not possible before. Thus, the determination of a correct order for the application of the Reduction rules depends on the controller Petri net model.

In each of the following subsections, a Reduction rule is applied to a CIPN, N_C, generating a reduced CIPN (RCIPN)

$$N'_C = \left(P'_C, T'_C, Pre'_C, Post'_C, x'_{0,C}, In'_C, C', E'_C, l'_C, D', l'_D, A', l'_A\right).$$

1) Recognizing a Condition for the Firing of a Transition Associated With a Sensor: In some cases, an ELPN is constructed in accordance with a bottom-up approach, in which, several components of the system are first modeled by Petri nets, and then these modules are appropriately connected in order to represent the coupling effects among them.

Let us assume that N_k and N_l are two net components of an ELPN that satisfy the following conditions:

C1.1—The places of N_k represent only discretized states the component may achieve.

C1.2—The transition from one state to another in N_k can be observed by sensors.

C1.3—The marking of a place p_i in N_k is one of the conditions for the firing of a transition t_j in N_l; this is tantamount to saying that there is a self-loop between p_i and t_j.

If conditions C1.1–C1.3 are satisfied, we can reduce the CIPN according to the following rule.

Reduction Rule 1:

1) Eliminate the connection between the modules:
   (a) If t_j is not a transition associated with an action, eliminate the self-loop between t_j and p_i in the CIPN, i.e., set $Pre'_C(p_i, t_j) = Post'_C(t_j, p_i) = 0$, and add to t_j a condition associated with a sensor S_i ($S_i = 1$ or $S_i = 0$).
   (b) If t_j is associated with an impulse action, eliminate the arcs in the CIPN between p_i and t_j^{in}, and t_j^{out} and p_i, i.e., set $Pre'_C(p_i, t_j^{in}) = Post'_C(t_j^{out}, p_i) = 0$, and add to t_j^{in} a condition associated with sensor S_i ($S_i = 1$ or $S_i = 0$).
2) Keep the initial marking of the net, i.e., set $\underline{x}'_{0,C} = \underline{x}_{0,C}$.
3) Eliminate net component N_k if it is isolated from the rest of the net and is not part of the control system, i.e., if it satisfies the following conditions:
   (a) None of its places has an assigned action.
   (b) None of its places is an input or output place of a transition of another module.
   (c) None of its transitions is an input or output transition of a place of a different module.

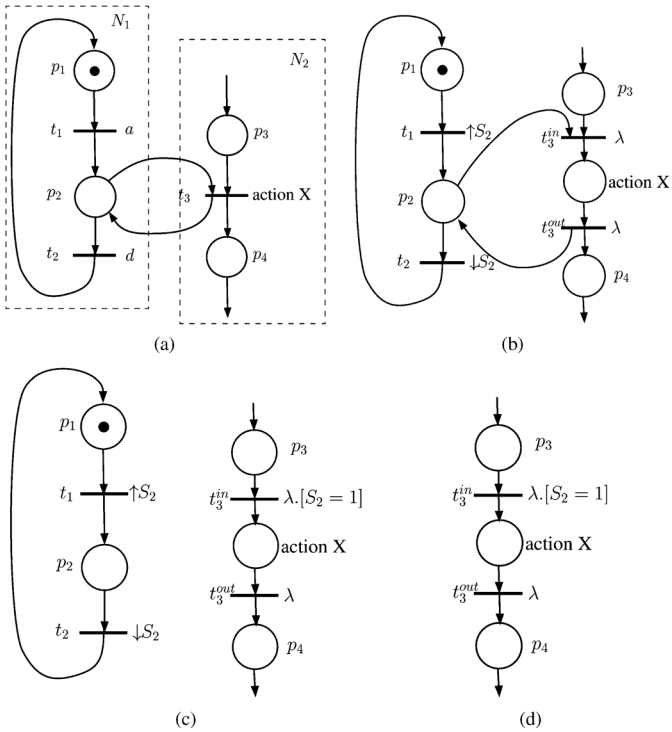

Fig. 4. (a) ELPN with two component models N_1 and N_2; (b) corresponding DCIPN; (c) CIPN obtained after eliminating the arcs between p_2 and t_3^{in}, and p_2 and t_3^{out}; (d) elimination of the isolated module N_1.

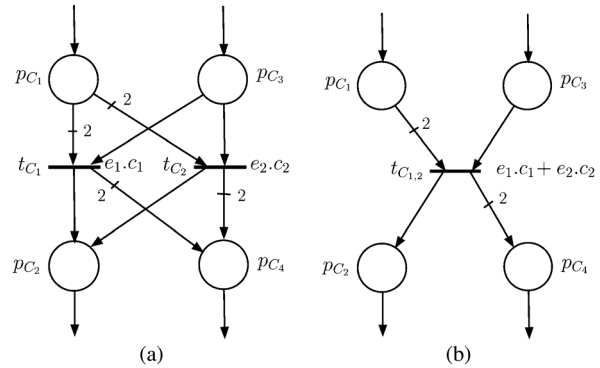

Fig. 5. (a) Part of a Petri net satisfying the conditions for the merging of transitions and (b) simplification by merging the transitions.

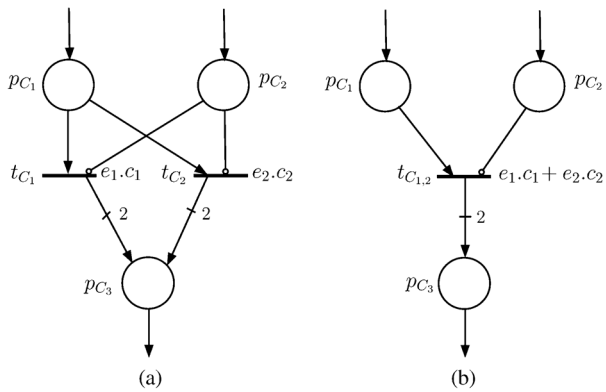

Fig. 6. (a) Part of an extended Petri net satisfying the conditions for the merging of transitions and (b) simplification by merging the transitions.

In Fig. 4(a), transition t_3 is labeled with an impulse action, and thus, applying the transformation rules T1–T3, the DCIPN of Fig. 4(b) is obtained. In this case, we can eliminate the arcs from p_2 to t_3^{in}, and from t_3^{out} to p_2. Fig. 4(c) presents the resulting reduced CIPN. Notice that module N_1 in the resulting Petri net is isolated from the rest of the net and does not represent part of the control system, and thus, it can be eliminated, leading to the reduced net of Fig. 4(d).

2) Merging Transitions: The merge of two transitions $t_{C_j}, t_{C_k} \in T_C^0$, whose associated events and conditions are (e_j, c_j) and (e_k, c_k), respectively, is possible if the following conditions are satisfied:

C2.1—The receptivities (e_j, c_j) and (e_k, c_k) are mutually exclusive.

C2.2—$Pre_C(p_{C_i}, t_{C_j}) = Pre_C(p_{C_i}, t_{C_k})$, for all $p_{C_i} \in P_C$.

C2.3—$In_C(p_{C_i}, t_{C_j}) = In_C(p_{C_i}, t_{C_k})$, for all $p_{C_i} \in P_C$.

C2.4—$Post_C(t_{C_j}, p_{C_i}) = Post_C(t_{C_k}, p_{C_i})$, for all $p_{C_i} \in P_C$.

Notice, according to Assumption A1, that t_{C_j} and t_{C_k} cannot be conflicting transitions. Thus, requirement C2.1 is not conservative when t_{C_j} and t_{C_k} also satisfy C2.2, C2.3, and C2.4. The reduction creates a new transition $t_{C_{j,k}} \in T'_C$, obtained after merging t_{C_j} and t_{C_k}, leading to a reduced net N'_C whose components are given as follows.

Reduction Rule 2:

1) $T'_C = [T_C \setminus \{t_{C_j}, t_{C_k}\}] \cup \{t_{C_{j,k}}\}$.
2) $P'_C = P_C$.
3) $Pre'_C(p_{C_i}, t_{C_y}) =$
$\begin{cases} Pre_C(p_{C_i}, t_{C_j}), & \text{if } t_{C_y} = t_{C_{j,k}}, \\ Pre_C(p_{C_i}, t_{C_y}), & \text{if } t_{C_y} \in T'_C \setminus \{t_{C_{j,k}}\}. \end{cases}$
4) $In'_C(p_{C_i}, t_{C_y}) =$
$\begin{cases} In_C(p_{C_i}, t_{C_j}), & \text{if } t_{C_y} = t_{C_{j,k}}, \\ In_C(p_{C_i}, t_{C_y}), & \text{if } t_{C_y} \in T'_C \setminus \{t_{C_{j,k}}\}. \end{cases}$
5) $Post'_C(t_{C_y}, p_{C_i}) =$
$\begin{cases} Post_C(t_{C_j}, p_{C_i}), & \text{if } t_{C_y} = t_{C_{j,k}}, \\ Post_C(t_{C_y}, p_{C_i}), & \text{if } t_{C_y} \in T'_C \setminus \{t_{C_{j,k}}\}. \end{cases}$
6) $\underline{x}'_{0,C} = \underline{x}_{0,C}$.
7) The event associated with $t_{C_{j,k}}$ is defined as $(e_j.c_j + e_k.c_k)$.

In Figs. 5(a) and 6(a), two transitions t_{C_1} and t_{C_2} satisfying conditions C2.1–C2.4 are presented. As shown in Figs. 5(b) and 6(b), a single transition $t_{C_{1,2}}$ synchronized with $(e_1c_1 + e_2c_2)$ is obtained by applying Reduction rule 2.

3) Merging Parallel Places: Two places p_{C_i} and p_{C_k} can be merged if the following conditions are satisfied.

C3.1—The set of input transitions of p_{C_i} is equal to the set of input transitions of p_{C_k}, $I(p_{C_i}) = I(p_{C_k})$.

C3.2—The set of output transitions of p_{C_i} is equal to the set of output transitions of p_{C_k}, i.e., $O(p_{C_i}) = O(p_{C_k})$.

C3.3—For each pair of transitions $t_{C_j}, t_{C_l} \in T_C$ such that $t_{C_j} \in I(p_{C_i})$ and $t_{C_l} \in O(p_{C_i})$, the following equation holds true:

$$\frac{Post_C\left(t_{C_j}, p_{C_i}\right)}{Pre_C\left(p_{C_i}, t_{C_l}\right)} = \frac{Post_C\left(t_{C_j}, p_{C_k}\right)}{Pre_C\left(p_{C_k}, t_{C_l}\right)}. \tag{5}$$

C3.4) The initial condition of p_{C_i} and p_{C_k} satisfies:

$$\frac{x_{0,C}\left(p_{C_i}\right)}{x_{0,C}\left(p_{C_k}\right)} = \frac{Post_C\left(t_{C_j}, p_{C_i}\right)}{Post_C\left(t_{C_j}, p_{C_k}\right)} \tag{6}$$

where t_{C_j} is an input transition of p_{C_i} and p_{C_k}.

It is not difficult to see that if (5) is satisfied for all pairs (t_{C_j}, t_{C_l}) of input and output transitions of p_{C_i} and p_{C_k}, then

$$\frac{Pre_C\left(p_{C_i}, t_{C_l}\right)}{Pre_C\left(p_{C_k}, t_{C_l}\right)} = \frac{Post_C\left(t_{C_j}, p_{C_i}\right)}{Post_C\left(t_{C_j}, p_{C_k}\right)} = \alpha \in \mathbb{Q}^+ \tag{7}$$

for all pairs (t_{C_j}, t_{C_l}), where $\mathbb{Q}^+$ denotes the set of positive rational numbers. Therefore

$$Pre_C\left(p_{C_i}, t_{C_l}\right) = \alpha Pre_C\left(p_{C_k}, t_{C_l}\right) \tag{8}$$

and

$$Post_C\left(t_{C_j}, p_{C_i}\right) = \alpha Post_C\left(t_{C_j}, p_{C_k}\right). \tag{9}$$

According to (9), if an input transition t_{C_j} fires then $Post_C(t_{C_j}, p_{C_k})$ tokens are added to place p_{C_k} and $\alpha Post_C(t_{C_j}, p_{C_k})$ tokens are added to place p_{C_i}. On the other hand, according to (8), if an output transition t_{C_l} fires then $Pre_C(p_{C_k}, t_{C_l})$ tokens are removed from place p_{C_k} and $\alpha Pre_C(p_{C_k}, t_{C_l})$ tokens are removed from p_{C_i}. In addition, if condition (6) is verified, the marking of p_{C_i} will be proportional to the marking of p_{C_k} for all reachable markings of the net, i.e.,

$$x_C\left(p_{C_i}\right) = \alpha x_C\left(p_{C_k}\right). \tag{10}$$

Therefore, it is not necessary to consider both p_{C_i} and p_{C_k} to the control of the firing of their output transitions, and so, places p_{C_i} and p_{C_k} can be merged, preserving the transition firing sequences of the net.

If there is an inhibitor arc from place p_{C_i} and/or p_{C_k} to a transition $t_{C_q} \in T_C$, and conditions C3.1–C3.4 are true, then it is necessary to define appropriately the weight of the inhibitor arc from the merged place $p_{C_{i,k}}$ to t_{C_q}. Let us first consider the case when $In_C(p_{C_i}, t_{C_q}) > 0$ and $In_C(p_{C_k}, t_{C_q}) = 0$. In this case two possibilities arise: (i) if $\alpha \leq 1$, then $In'_C(p_{C_{i,k}}, t_{C_q}) = In_C(p_{C_i}, t_{C_q})$, since the marking of $p_{C_{i,k}}$ will be equal to the marking of p_{C_i} for all reachable states of the Petri net and (ii) if $\alpha > 1$, then the weight of the inhibitor arc $In'_C(p_{C_{i,k}}, t_{C_q})$ can be different from $In_C(p_{C_i}, t_{C_q})$ since $x_C(p_{C_{i,k}}) = x_C(p_{C_k})$ for all reachable states of the net, and, according to (10), the marking of p_{C_i} is proportional to that of p_{C_k}, which implies that $x_C(p_{C_i}) = \alpha x_C(p_{C_{i,k}})$. Thus, the inhibitor condition $x_C(p_{C_i}) \geq In_C(p_{C_i}, t_{C_q})$ is equivalent to the condition

$$x_C\left(p_{C_{i,k}}\right) \geq \frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}. \tag{11}$$

However, the right-hand side of Inequality (11) can be a non-integer, and therefore, it cannot be implemented as the weight of an inhibitor arc in N'_C. In order to overcome this problem, notice that, if

$$x_C\left(p_{C_{i,k}}\right) \geq ceil\left(\frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}\right)$$

where $ceil(z)$ denotes the smallest integer value greater than or equal to z, then condition (11) is verified. Moreover, it can be easily seen that, since $x_C(p_{C_{i,k}})$ is an integer value, if

$$x_C\left(p_{C_{i,k}}\right) < ceil\left(\frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}\right)$$

then

$$x_C\left(p_{C_{i,k}}\right) < \frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}$$

which implies that the new inhibitor arc weight in N'_C, equivalent to $In_C(p_{C_i}, t_{C_q})$ in N_C, is given by

$$In'_C\left(p_{C_{i,k}}, t_{C_q}\right) = ceil\left(\frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}\right). \tag{12}$$

Now, consider the case when $In_C(p_{C_i}, t_{C_q}) > 0$ and $In_C(p_{C_k}, t_{C_q}) > 0$. In this case, there are two inhibitor conditions for the firing of transition t_{C_q}, as follows:

$$x_C\left(p_{C_i}\right) \geq In_C\left(p_{C_i}, t_{C_q}\right) \tag{13}$$

and

$$x_C\left(p_{C_k}\right) \geq In_C\left(p_{C_k}, t_{C_q}\right). \tag{14}$$

After merging places p_{C_i} and p_{C_k}, a single inhibitor condition for the firing of t_{C_q} must be obtained. If $\alpha > 1$, then $x_C(p_{C_{i,k}}) = x_C(p_{C_k})$ and condition (14) leads to the following condition for the marking of the merged place $p_{C_{i,k}}$:

$$x_C\left(p_{C_{i,k}}\right) \geq In_C\left(p_{C_k}, t_{C_q}\right). \tag{15}$$

Since $x_C(p_{C_i}) = \alpha x_C(p_{C_k})$ for all reachable markings of the net, condition (13) yields

$$x_C\left(p_{C_{i,k}}\right) \geq \frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}. \tag{16}$$

Thus, from inequalities (15) and (16), it is easy to see that the new inhibitor arc $In'_C(p_{C_{i,k}}, t_{C_q})$ can be defined as

$$In'_C\left(p_{C_{i,k}}, t_{C_q}\right) = \min\left\{ceil\left(\frac{In_C\left(p_{C_i}, t_{C_q}\right)}{\alpha}\right), In_C\left(p_{C_k}, t_{C_q}\right)\right\}. \tag{17}$$

Using the same reasoning, it can be shown that if $\alpha \leq 1$, the new inhibitor arc can be defined as

$$In'_C\left(p_{C_{i,k}}, t_{C_q}\right) = \min\left\{In_C\left(p_{C_i}, t_{C_q}\right), ceil\left(\alpha In_C\left(p_{C_k}, t_{C_q}\right)\right)\right\}. \tag{18}$$

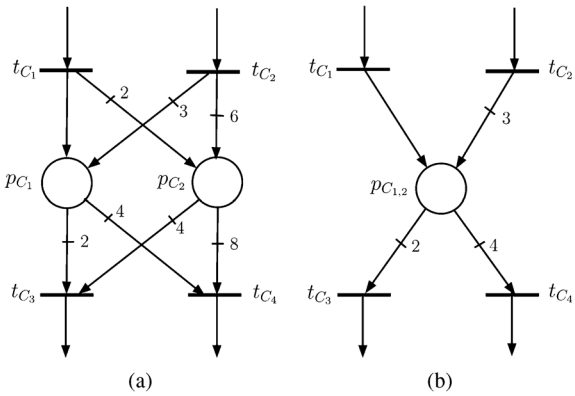

Fig. 7. (a) Part of a Petri net satisfying the conditions for the merging of parallel places and (b) simplification by merging the places.

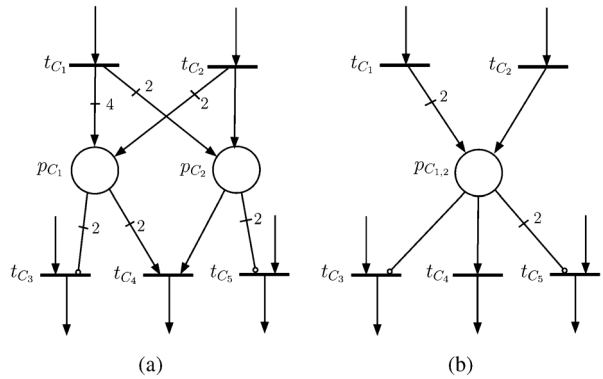

Fig. 8. (a) Part of an extended Petri net satisfying the conditions for the merging of places and (b) simplification by merging the places.

The merge of places $p_{C_i}, p_{C_k} \in P_C$ leads to a new CIPN N'_C obtained as follows.

Reduction Rule 3:

1) $T'_C = T_C$.
2) $P'_C = [P_C \setminus \{p_{C_i}, p_{C_k}\}] \cup \{p_{C_{i,k}}\}$.
3) $In'_C(p_{C_x}, t_{C_q}) =$
$$\begin{cases} In_C(p_{C_x}, t_{C_q}), \text{ if} p_{C_x} \in P'_C \setminus \{p_{C_{i,k}}\}. \\ In_C(p_{C_i}, t_{C_q}), \text{ if} p_{C_x} = p_{C_{i,k}}, In_C(p_{C_i}, t_{C_q}) > 0, \\ \qquad In_C(p_{C_k}, t_{C_q}) = 0, \text{ and } \alpha \leq 1. \\ ceil(In_C(p_{C_i}, t_{C_q})/\alpha), \text{ if } p_{C_x} = p_{C_{i,k}}, \\ \qquad In_C(p_{C_i}, t_{C_q}) > 0,\ In_C(p_{C_k}, t_{C_q}) = 0, \\ \qquad \text{and } \alpha > 1. \\ \min\{ceil(In_C(p_{C_i}, t_{C_q})/\alpha), In_C(p_{C_k}, t_{C_q})\} \\ \qquad \text{if } p_{C_x} = p_{C_{i,k}}, In_C(p_{C_i}, t_{C_q}) > 0, \\ \qquad In_C(p_{C_k}, t_{C_q}) > 0, \text{ and } \alpha > 1. \\ \min\{In_C(p_{C_i}, t_{C_q}), ceil(\alpha In_C(p_{C_k}, t_{C_q}))\} \\ \qquad \text{if } p_{C_x} = p_{C_{i,k}}, In_C(p_{C_i}, t_{C_q}) > 0, \\ \qquad In_C(p_{C_k}, t_{C_q}) > 0, \text{ and } \alpha \leq 1. \end{cases}$$
4) $Post'_C(t_{C_j}, p_{C_x}) =$
$$\begin{cases} \min\{Post_C(t_{C_j}, p_{C_i}), \\ \qquad Post_C(t_{C_j}, p_{C_k})\}, & \text{if } p_{C_x} = p_{C_{i,k}}, \\ Post_C(t_{C_j}, p_{C_x}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_{i,k}}\}. \end{cases}$$
5) $Pre'_C(p_{C_x}, t_{C_l}) =$
$$\begin{cases} \min\{Pre_C(p_{C_i}, t_{C_l}), \\ \qquad Pre_C(p_{C_k}, t_{C_l})\}, & \text{if } p_{C_x} = p_{C_{i,k}} \\ Pre_C(p_{C_x}, t_{C_l}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_{i,k}}\}. \end{cases}$$
6) If any of the places $p_{C_i}, p_{C_k} \in P_C$ is associated with actions, then these actions must be assigned to $p_{C_{i,k}} \in P'_C$.
7) $x'_{0,C}(p_{C_x}) =$
$$\begin{cases} \min\{x_{0,C}(p_{C_i}), x_{0,C}(p_{C_k})\}, & \text{if } p_{C_x} = p_{C_{i,k}} \\ x_{0,C}(p_{C_x}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_{i,k}}\}. \end{cases}$$

In Fig. 7(a) part of a Petri net with two places p_{C_1} and p_{C_2} that satisfy conditions C3.1–C3.4 are presented. Following Reduction Rule 3, we see that places p_{C_1} and p_{C_2} can be merged to form place $p_{C_{1,2}}$, as shown in Fig. 7(b). Notice the new arc weights that were obtained according to item 4) of Reduction Rule 3. In Fig. 8(a), part of an extended Petri net is shown to illustrate the merge of parallel places when there are inhibitor arcs. The merged place and the corresponding arcs obtained by following Reduction Rule 3 are shown in Fig. 8(b).

4) Merging Consecutive Places Connected Through λ-Transitions: λ transitions are usually created when DCIPNs are formed from ELPNs. In some cases, consecutive places connected through λ-transitions can be merged and the corresponding transition eliminated, reducing the CIPN.

The conditions for the merge of the input and output places of a λ-transition t_{C_j} can be divided in three cases: (i) there is no action associated with the input and output places of t_{C_j}; (ii) the input place of t_{C_j}, p_{C_i}, is a safe place with assigned action; and (iii) the input place p_{C_i} does not have assigned actions and at least one output place of t_{C_j}, $p_{C_o} \in O(t_{C_j})$, is a safe place with assigned action.

Let us first consider case (i). In this case, it is possible to reduce the CIPN if the following conditions are satisfied:

C4.1—$Pre_C(p_{C_i}, t_{C_j}) = 1$.
C4.2—The set of input places of t_{C_j} is a singleton, i.e., $I(t_{C_j}) = \{p_{C_i}\}$.
C4.3—$In_C(p_{C_x}, t_{C_j}) = 0$ for all places $p_{C_x} \in P_C$.

If conditions C4.1–C4.3 are satisfied, then when a transition $t_{C_q} \in I(p_{C_i})$ fires, an unstable marking is reached and transition t_{C_j} immediately fires leading to a new marking.

Consider, now, case (ii). In this case, some reduction in the CIPN is possible if C4.1–C4.3 are satisfied and, in addition, there is an output place $p_{C_o} \in O(t_{C_j})$ such that:

C4.4—p_{C_o} is safe.
C4.5—$x_{0,C}(p_{C_o}) = 0$.
C4.6—$I(p_{C_o}) = \{t_{C_j}\}$.

Condition C4.4 is necessary to guarantee that there will be a merged place $p_{C_{i,o}}$ to which the action associated with p_{C_i} can be assigned, and that $p_{C_{i,o}}$ will be a safe place for all reachable markings of the reduced Petri net. It is important to remark that p_{C_o} can also have an assigned action. Conditions C4.5 and C4.6 guarantee that the action assigned to p_{C_i} will be executed only if p_{C_i} has initially one token or an input transition of $p_{C_{i,o}}$ that is also an input transition of p_{C_i} fires.

Finally, in case (iii), the reduction is possible if C4.1–C4.3 are satisfied and

C4.7—p_{C_i} is a safe place.

Condition C4.7 is necessary since $p_{C_{i,o}}$ must also be a safe place.

The development above leads to the following reduced rules.

Reduction Rule 4:

1) Create the set of merged places $P_M = \{p_{C_{i,o}} : p_{C_i} \in I(t_{C_j}) \wedge p_{C_o} \in O(t_{C_j})\}$. Notice that since $I(t_{C_j})$ is a

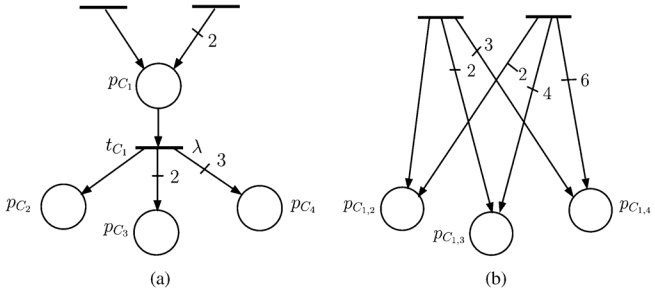

Fig. 9. (a) Consecutive places connected by a λ-transition and (b) reduction by merging the places.

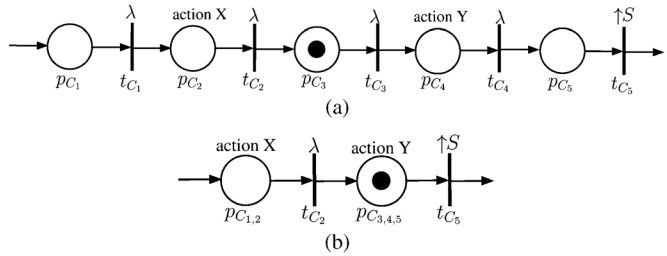

Fig. 10. (a) Consecutive places connected by λ-transitions and (b) reduction by merging the places.

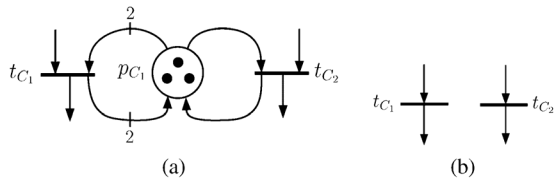

Fig. 11. (a) Redundant representation of resource sharing and (b) reduced Petri net obtained by eliminating the redundant place p_{C_1}.

singleton, the cardinality of P_M is equal to the cardinality of $O(t_{C_j})$.

2) $P'_C = [P_C \setminus (\{p_{C_i}\} \cup O(t_{C_j}))] \cup P_M$.

3) $T'_C = T_C \setminus \{t_{C_j}\}$.

4) $Pre'_C(p_{C_x}, t_{C_y}) =$
$$\begin{cases} Pre_C(p_{C_o}, t_{C_y}), & \text{if } p_{C_x} = p_{C_{i,o}} \in P_M, \\ & \text{and } t_{C_y} \in T'_C, \\ Pre_C(p_{C_x}, t_{C_y}), & \text{if } p_{C_x} \in P'_C \setminus P_M, \\ & \text{and } t_{C_y} \in T'_C. \end{cases}$$

5) $Post'_C(t_{C_y}, p_{C_x}) =$
$$\begin{cases} Post_C(t_{C_y}, p_{C_i}) \times Post_C(t_{C_j}, p_{C_o}), \\ \quad \text{if } p_{C_x} = p_{C_{i,o}} \in P_M, \text{ and } t_{C_y} \in I(p_{C_i}), \\ Post_C(t_{C_y}, p_{C_o}), \text{ if } p_{C_x} = p_{C_{i,o}} \in P_M, \\ \quad \text{and } t_{C_y} \in I(p_{C_o}). \\ Post_C(t_{C_y}, p_{C_x}), \text{ if } p_{C_x} \in P'_C \setminus P_M. \end{cases}$$

6) $In'_C(p_{C_x}, t_{C_y}) =$
$$\begin{cases} In_C(p_{C_o}, t_{C_y}), & \text{if } p_{C_x} = p_{C_{i,o}} \in P_M, \\ & \text{and } t_{C_y} \in T'_C, \\ In_C(p_{C_x}, t_{C_y}), & \text{if } p_{C_x} \in P'_C \setminus P_M, \\ & \text{and } t_{C_y} \in T'_C. \end{cases}$$

7) $x'_{0,C}(p_{C_{i,o}}) =$
$$\begin{cases} x_{0,C}(p_{C_o}) + Post_C(t_{C_j}, p_{C_o}) \\ \quad \times x_{0,C}(p_{C_i}), & \text{if } p_{C_x} = p_{C_{i,o}} \in P_M \\ x_{0,C}(p_{C_x}), & \text{if } p_{C_x} \in P'_C \setminus P_M \end{cases},$$

8) If place p_{C_i} is associated with an action, then this action must be associated with a place $p_{C_{i,o}} \in P_M$ such that p_{C_o} is a safe place. If p_{C_o} is associated with an action, this action must be assigned to place $p_{C_{i,o}} \in P_M$.

In Fig. 9(a), conditions C4.1–C4.3 for the elimination of transition t_{C_1} and the merge of input place p_{C_1} with the output places p_{C_2}, p_{C_3}, and p_{C_4} are satisfied. Applying Reduction rule 4, the reduced Petri net of Fig. 9(b) is obtained. Fig. 10(a) and (b) present a different example of the application of Reduction rule 4. Notice that places p_{C_2} and p_{C_3} have not been merged since p_{C_3} has initially one token and p_{C_2} has an assigned action, therefore, violating condition C4.5.

5) Eliminating Redundant Places: With the view to describing the states of the plant and controller in the closed-loop system, the ELPN can introduce false synchronizations, i.e., some places, describing states of the plant, can be redundant to the control of the firing of a transition.

A place $p_{C_i} \in P_C$ is said to be redundant with respect to its set of output transitions, if the following conditions are satisfied for all reachable markings of the net:

C5.1—p_{C_i} does not have assigned actions.

C5.2—The number of tokens in p_{C_i} is sufficient to fire all enabled output transitions of p_{C_i}.

C5.3—For each output transition of p_{C_i}, $t_{C_o} \in O(p_{C_i})$, there is a nonempty subset of input places, $P_{C_E} \subset I(t_{C_o}) \setminus \{p_{C_i}\}$.

C5.4—The enabling conditions of each output transition $t_{C_o} \in O(p_{C_i})$, associated with the markings of the places of P_{C_E}, are all satisfied only if the enabling condition related with the marking of p_{C_i} is also satisfied.

The search for redundant places can be carried out by using the reachability graph of the net. This identification can be carried out in three steps, as follows: (i) find synchronization structures in the net; (ii) for each place p_{C_i} involved in a synchronization, obtain its set of output transitions $O(p_{C_i})$ and verify if there is at least one input place different from p_{C_i} for each output transition $t_{C_o} \in O(p_{C_i})$; and (iii) from the reachability graph of the net, verify, for each one of the output transitions t_{C_o} of p_{C_i}, and for all reachable states of the Petri net, if the conditions for enabling t_{C_o}, provided by the markings of the places of P_{C_E}, are all satisfied and the condition obtained from the marking of p_{C_i} is not satisfied. In this case, p_{C_i} is not a redundant place since its marking is crucial to enable t_{C_o}; otherwise, p_{C_i} is a redundant place.

In some special cases, the identification of a redundant place p_{C_i} with respect to its set of output transitions is simple, since it requires the search for a specific structure in the net and its associated initial condition. In [35], a Reduction rule is presented for the elimination of a redundant representation of resource sharing, that is a special case of redundant place as defined in this paper. An example of redundant resource sharing is presented in Fig. 11.

Notice that if conditions C5.1–C5.4 are satisfied, then place p_{C_i} is not necessary to the control of the firing of any of its output transitions. Hence, place p_{C_i} and its associated arcs can be eliminated and the net reduced as follows.

Reduction Rule 5:

1) $P'_C = P_C \setminus \{p_{C_i}\}$.

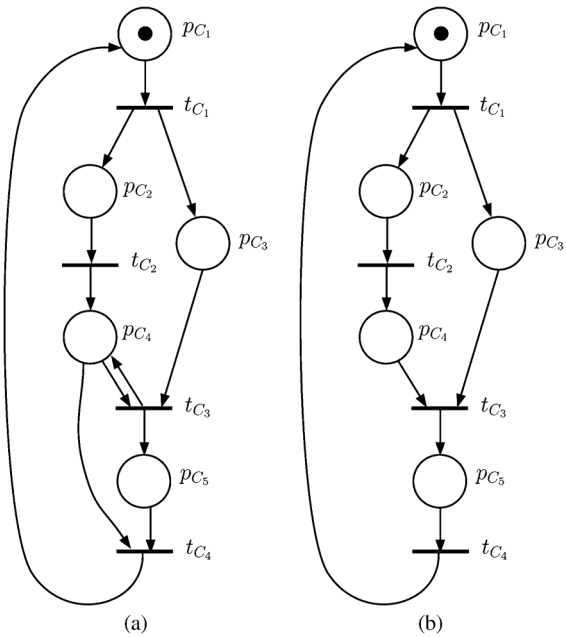

Fig. 12. (a) Example of a Petri net with redundant places and (b) reduced Petri net obtained by eliminating the redundant places.

2) $T'_C = T_C$.
3) $Pre'_C(p_{C_x}, t_{C_y}) =$
$\begin{cases} 0, & \text{if } p_{C_x} = p_{C_i}, \\ Pre_C(p_{C_x}, t_{C_y}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_i}\}. \end{cases}$
4) $Post'_C(t_{C_y}, p_{C_x}) =$
$\begin{cases} 0, & \text{if } p_{C_x} = p_{C_i}, \\ Post_C(t_{C_y}, p_{C_x}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_i}\}. \end{cases}$
5) $In'_C(p_{C_x}, t_{C_y}) =$
$\begin{cases} 0, & \text{if } p_{C_x} = p_{C_i}, \\ In_C(p_{C_x}, t_{C_y}), & \text{if } p_{C_x} \in P'_C \setminus \{p_{C_i}\}. \end{cases}$
6) $x'_{0,C}(p_{C_x}) = x_{0,C}(p_{C_x})$, for all $p_{C_x} \in P'_C$.

It is important to remark that in some cases, a place p_{C_i} can be redundant with respect to a transition t_{C_j} and not redundant for the firing of a different transition t_{C_l}. In these cases, depending on the net behavior described by the language generated by the system, this redundancy can also be eliminated. Consider, for instance, the Petri net shown in Fig. 12(a). Notice that place p_{C_5} can receive a token only if place p_{C_4} already has a token, and the firing of t_{C_4} removes the tokens of both places p_{C_5} and p_{C_4}. Thus, place p_{C_4} is redundant for the firing of transition t_{C_4}. However, p_{C_4} is not redundant with respect to transition t_{C_3}, since t_{C_3} can fire only after the firing sequence $t_{C_1}t_{C_2}$ has been executed. In this example the false synchronization of transition t_{C_4} can be eliminated by changing properly the structure of the net. This modification in the net structure must guarantee that the admissible transition firing sequences of the system are not changed. In Fig. 12(b), the reduced net without the false synchronization in transition t_{C_4} is shown. It can be seen that the admissible sequence of transitions has not been modified with this net structure.

The identification of a redundant place p_{C_i} with respect to an output transition $t_{C_o} \in O(p_{C_i})$ and the elimination of the redundancy by redesigning the Petri net can be very laborious and difficult to be automated. However, the analysis effort is rewarded with a Petri net with reduced number of places and transitions, which ultimately implies in a smaller programming code.

C. Example

In this Section, a plastic molding machine is used to illustrate the CIPN extraction from an ELPN. The molding machine consists of a hopper, an injection system composed by a reciprocating screw and barrel assembly, and a molding system, as shown in Fig. 13. The production of a part in a plastic injection molding machine consists, initially, in feeding a thermoplastic material in the form of small pellets to a hopper. This material is then gravity-fed from the hopper into the injection system, which is heated by electric heater bands. The reciprocating screw is used to compress, melt, and convey the material. After the approximation of the barrel to the molding system, the reciprocating screw compress the material against the inside diameter of the barrel, creating heat due to viscous friction. The friction and the compression are amplified by the progressive reduction in the diameter of the barrel. Thus, while the material is transported through the barrel, it melts. The heater bands outside the barrel help to maintain the material in the molten state. When the material in molten state reaches the end of the barrel, it is injected in the mold. The molding system shapes the plastic inside the cavity and ejects the molded part. After the injection, the part is cooled and solidified to the desired shape defined by the cavity. The solidified part is extracted by a hydraulic knock-out (ejector) system mounted on the stationary platen. The controlled variables in this process are the speed of the system components and the pressure of the fluid in the hydraulic valves responsible for the displacements.

The process can be divided into six operations: clamping of the mold, approximation of the injection system to the molding system, injection, return of the injection unit, opening of the mold, and extraction. Due to lack of space, we will consider only the mold opening process.

After the injection of the plastic material into the mold, the mold is opened and the extraction of the molded part is carried out at the same time as the moving platen returns to its initial position. Three different speeds, which depend on the position of the moving platen, must be set. These three positions are identified by position sensors S_{p_5}, S_{p_6}, and S_{p_7}.

After the end of the injection process, a time delay d_{27} must be set for the beginning of the opening process. Then, the moving platen starts to return, with speed $V_p = V_{p_4}$ and pressure $P_p = P_{p_4}$. When the moving platen reaches position S_{p_5}, the speed must change to $V_p = V_{p_5}$ and when the moving platen reaches position S_{p_6}, the speed must change to $V_p = V_{p_6}$. At the end of the displacement, at position S_{p_7}, the speed and pressure must both be set to zero.

Fig. 14 shows the ELPN of the opening process and Tables I and II present the meaning of each place and the label of each transition of the ELPN, respectively. Notice that, the ELPN has several places and transitions, since the ELPN is built in order to model the joint behavior of the plant and controller. Therefore, after each observation of an event in the system, the labeled Petri net must show the state of the whole closed loop system, which includes both the states of the plant and the controller. If the PLC

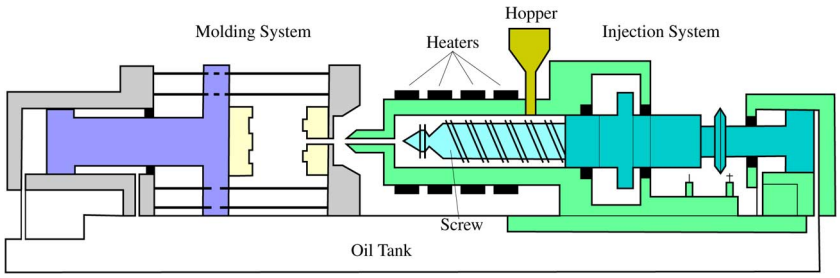

Fig. 13. Plastic injection molding machine scheme showing the three main subsystems. (Figure adapted from [38].)

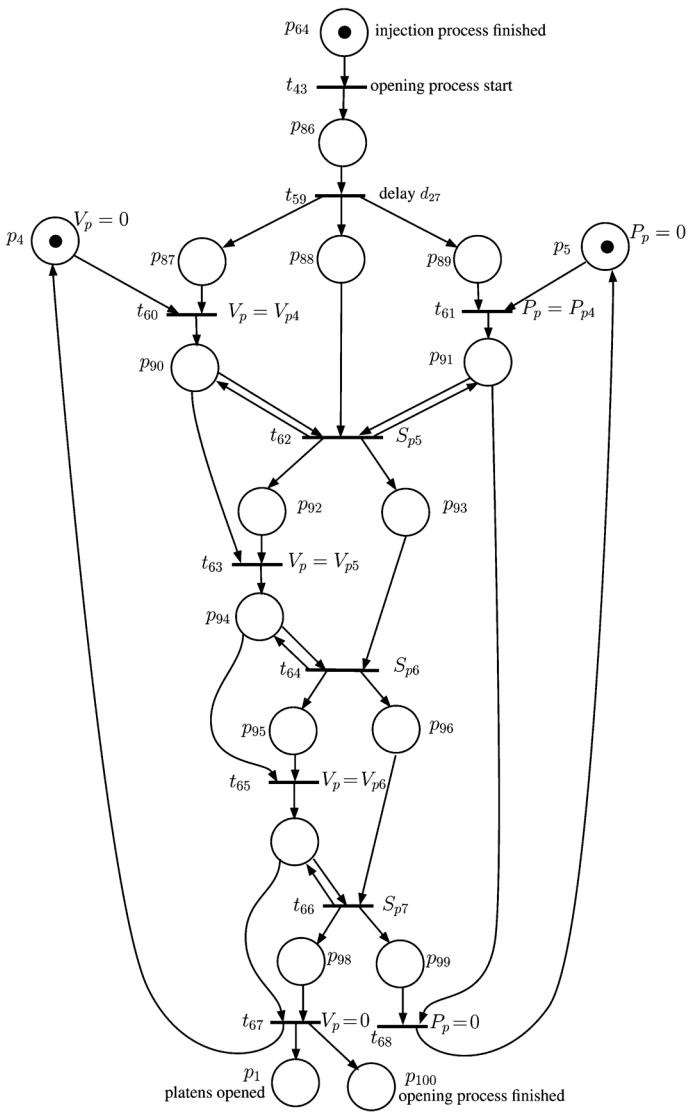

Fig. 14. Labeled Petri net of the opening process.

TABLE I
PLACE DESCRIPTIONS OF THE ELPN

Place	Description
p_4	$V_P = 0$
p_5	$P_P = 0$
p_{64}	Injection process finished
p_{86}	Opening process started
p_{87}	Speed change allowed
p_{88}	Platen between initial position and S_{P_5}
p_{89}	Pressure change allowed
p_{90}	$V_P = V_{P_4}$
p_{91}	$P_P = P_{P_4}$
p_{92}	Speed change allowed
p_{93}	Moving platen between S_{P_5} and S_{P_6}
p_{94}	$V_P = V_{P_5}$
p_{95}	Speed change allowed
p_{96}	Moving platen between S_{P_6} and S_{P_7}
p_{97}	$V_P = V_{P_6}$
p_{98}	Speed change allowed
p_{99}	Pressure change allowed
p_{100}	Opening process finished
p_1	Platens opened

TABLE II
TRANSITION LABELS OF THE ELPN

Transition	label
t_{43}	Opening process start
t_{59}	Time delay d_{27}
t_{60}	Speed change to V_{P4}
t_{61}	Pressure change to P_{P4}
t_{62}	Moving platen at position S_{P5}
t_{63}	Speed change to V_{P5}
t_{64}	Moving platen at position S_{P6}
t_{65}	Speed change to V_{P6}
t_{66}	Moving platen at position S_{P7}
t_{67}	Speed change to zero
t_{68}	Pressure change to zero

programming code were directly obtained from the ELPN, then it would have a complexity not compatible with the simplicity of the problem. It is, therefore, necessary to obtain from the ELPN a Petri net that models the controller behavior only, which, in our paper, is the RCIPN.

The first step to obtain the DEC model is to identify the events associated with sensor signals, timers, internal events, and actions. After that, the transitions labeled with actions are expanded as presented in Section IV-A. The resulting DCIPN is presented in Fig. 15.

The RCIPN can be found by applying the Reduction rules presented in Section IV-B to the DCIPN of Fig. 15. The steps for the computation of the RCIPN are as follows:

1) Notice that, places p_4 and p_5 are redundant with respect to their output transitions t_{60}^{in} and t_{61}^{in}, respectively. Thus, applying Reduction rule 5, both places and their related arcs can be eliminated. Reduction rule 5 can also be applied to modify the net structure in order to eliminate false synchronizations. It can be easily seen that places p_{90}, p_{94}, p_{97}, and p_{91} are redundant with respect to the output transitions t_{63}^{in}, t_{65}^{in}, t_{67}^{in}, and t_{68}^{in}, respectively. Thus, according to Reduction rule 5, the false synchronizations can be removed by eliminating the arcs (p_{90}, t_{63}^{in}), (t_{62}, p_{90}),

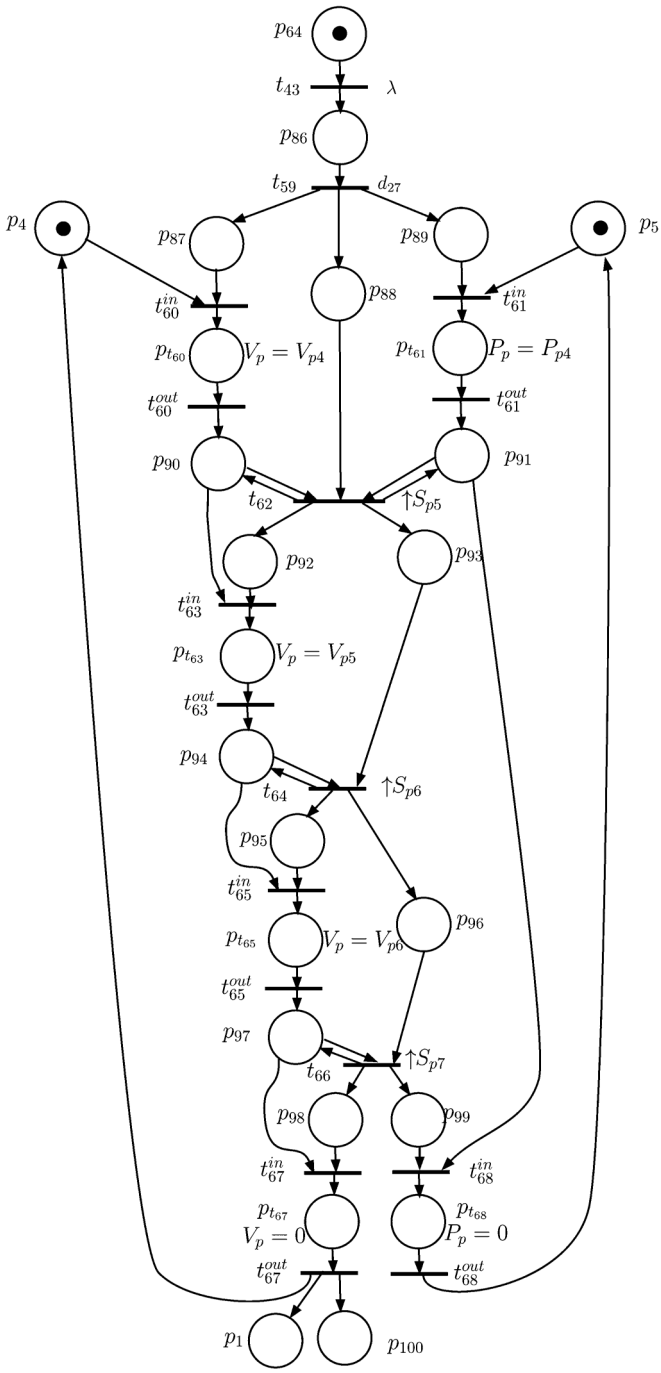

Fig. 15. DCIPN of the opening process.

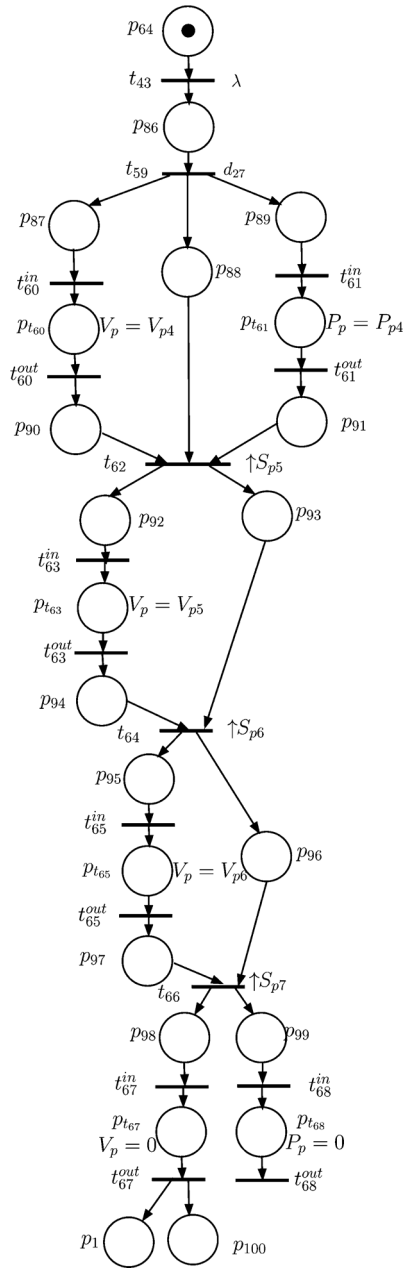

Fig. 16. CIPN obtained after the application of Reduction rule 5 to the CIPN of Fig. 15.

(p_{94}, t_{65}^{in}), (t_{64}, p_{94}), (p_{97}, t_{67}^{in}), (t_{66}, p_{97}), (p_{91}, t_{68}^{in}), and (t_{62}, p_{91}). The resulting Petri net is shown in Fig. 16.

2) The next step is the merge of places connected through λ-transitions. Applying Reduction rule 4, several places are merged leading to new places in the Petri net. For instance, place $p_{87,t_{60},90}$ is obtained after the merge of places p_{87}, $p_{t_{60}}$, p_{90} and place $p_{89,t_{61},91}$ is obtained after the merge of places p_{89}, $p_{t_{61}}$, p_{91}. The resulting Petri net is shown in Fig. 17.
3) Finally, Reduction rule 3 can be applied to merge parallel places. Place $p_{87,t_{60},90,89,t_{61},91}$ of the RCIPN is obtained after the merge of the parallel places $p_{87,t_{60},90}$ and $p_{89,t_{61},91}$. Notice that, in this case, the actions associated with both places are assigned to the merged place $p_{87,t_{60},90,89,t_{61},91}$. Places p_{88}, p_{93}, p_{96}, represent part of the states of the plant and are redundant. These places were also eliminated in the RCIPN by applying Reduction rule 5. Places $p_{64,86}$, $p_{98,t_{67},1}$, and $p_{98,t_{67},100}$ are used in the control of the other operations of the plastic injection molding machine and, therefore, remain unaltered. The resulting RCIPN is shown in Fig. 18.

It is important to stress that the number of places and transitions in the RCIPN is smaller than the number of places and transitions of the ELPN of Fig. 14, since several places were redundant to the control logic and could be eliminated from the Petri net.

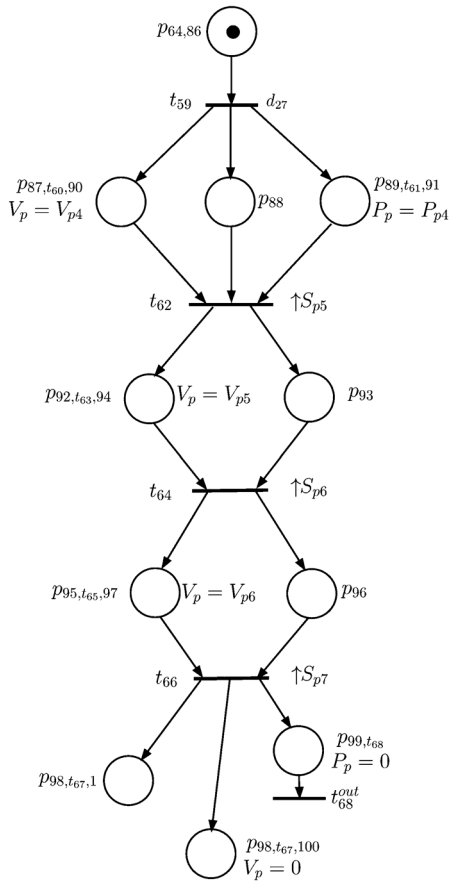

Fig. 17. CIPN obtained after the application of Reduction rule 4 to the CIPN of Fig. 16.

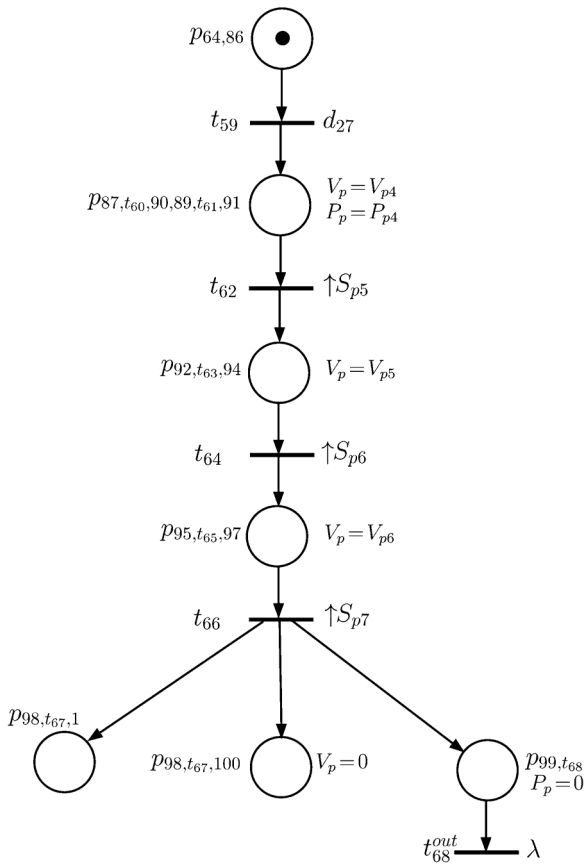

Fig. 18. Resulting RCIPN of the opening process after application of Reduction rules 3 and 5 to the CIPN of Fig. 17.

The ELPN model of all six processes of the plastic injection molding machine has 112 places and 77 transitions, while its corresponding RCIPN, obtained by following the transformation rules of Section IV, has only 78 places and 59 transitions.

Remark 1: It could be argued that it might be possible to design directly the RCIPN of the opening process shown in Fig. 18. This is indeed true if the chosen design methodology is the logic controller approach [7]. However, since in the logic controller design strategy, the designer directly designs a controller by defining its input-output behavior, without finding the model of the controlled system first, there is no monitoring of the dynamic evolution of the system as a whole, which could be a nontrivial task for more complex systems. On the other hand, in the controlled behavior approach, as we assume in the paper as the design methodology deployed, the first step is to obtain the controlled behavior of the closed-loop system and, in the sequel, to extract the controller model. In this regard, the proposed Reduction rules play a crucial role. It is worth remarking that the fact that the application of the Reduction rules leads to a model that could be a natural choice for a first DEC, within the logic controller approach, shows the effectiveness of the proposed Reduction rules.

V. Conversion of CIPNs Into Ladder Diagrams

A PLC operates executing scan cycles that consist of three main steps: (i) reading and storage of PLC inputs; (ii) execution of the user programming code; and (iii) output update. In general, input events are associated with the rising or the falling edges of sensor signals and the outputs are commands sent by the controller to the plant in response to changes in the values of the sensor signals.

We propose in this paper a new method for the conversion of CIPN into ladder diagrams that extends the method presented in [25]. The proposed scheme consists in dividing the ladder diagram in five modules, as follows:

- Module M1, associated with the identification of the occurrence of external events;
- Module M2, associated with the conditions for the firing of the transitions;
- Module M3, that describes the evolution of the tokens in the Petri net;
- Module M4, that represents the initialization of the Petri net, i.e., it defines the initial marking;
- Module M5, that defines which actions will be set in the current state of the system.

In the following subsections, we will present a detailed explanation of each one of the five modules and will present a running example to illustrate the conversion method proposed here. The Petri net graph for the example is depicted in Fig. 19, which was specially built to illustrate the construction of each one of the modules.

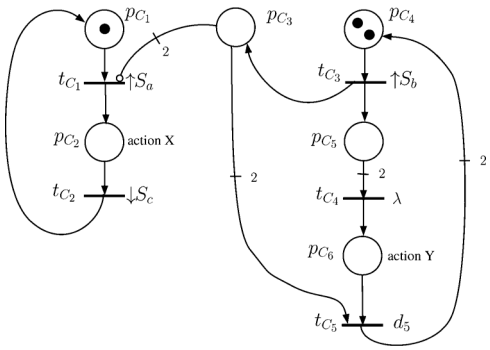

Fig. 19. CIPN used to illustrate the conversion technique presented in Section V.

A. The Module of External Events

External events are associated with the rising or the falling edge of sensor signals in the CIPN and can be detected by using a positive signal edge contact (P) or a negative signal edge contact (N), respectively. The P (N) contact is normally open and it closes, for just one scan cycle, when the Boolean condition in the same rung changes the logical value from zero to one (one to zero). The always occurring event is not represented in the module of external events since it is an internal event [3].

In the Petri net of Fig. 19, there are four events synchronizing the transitions: two events are associated with the rising edge of sensor signals, $\uparrow S_a$ and $\uparrow S_b$, one event is associated with the falling edge of a sensor signal, $\downarrow S_c$, and the last event is the always occurring event λ. Therefore, the module of external events for this CIPN must have three rungs, as shown in Fig. 20. The first and second rungs of the ladder diagram account for the external events associated with the rising edge of sensor signals $\uparrow S_a$ and $\uparrow S_b$, respectively. When, for instance, S_a changes its value from zero to one, the P contact closes for one scan cycle energizing the coil denoted as S_{ar}, that represents the rising edge of S_a, $\uparrow S_a$. The third rung of the ladder diagram of Fig. 20 accounts for the falling edge of the sensor signal S_c. When S_c changes its value from one to zero, the N contact closes energizing coil S_{cf} that corresponds to the falling edge of S_c, $\downarrow S_c$.

B. The Module of Firing Conditions

The module of firing conditions has $|T_C|$ rungs, where $|.|$ denotes cardinality, and each rung describes the conditions for the firing of a transition $t_{C_j} \in T_C$. Since the CIPN is an extended Petri net, then a transition t_{C_j} is enabled if and only if

$$x_C\left(p_{C_i}\right) \geq Pre_C\left(p_{C_i}, t_{C_j}\right), \forall p_{C_i} \in I\left(t_{C_j}\right) \qquad (19)$$

and

$$x_C\left(p_{C_i}\right) < In_C\left(p_{C_i}, t_{C_j}\right), \forall In_C\left(p_{C_i}, t_{C_j}\right) > 0. \qquad (20)$$

If transition $t_{C_j} \in T_C^0$, then t_{C_j} is fireable if conditions (19) and (20) are satisfied and the associated condition c_j is true, and it fires when the associated event e_j occurs. On the other hand, if $t_{C_j} \in T_C^D$ then t_{C_j} is fireable if conditions (19) and (20) are satisfied but it fires only after a delay time d_j.

Enabling conditions (19) and (20) can be easily expressed in the ladder diagram by using comparison instructions, which are connected in series with other elements that depend on whether

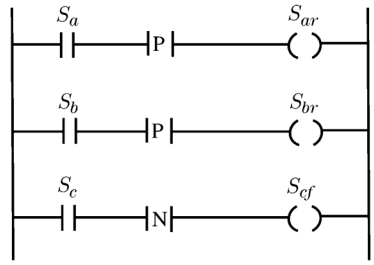

Fig. 20. Module of external events for the CIPN of Fig. 19.

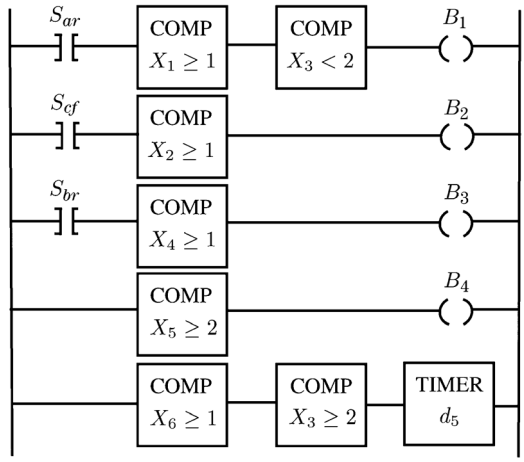

Fig. 21. Module of firing conditions for the CIPN of Fig. 19.

transition t_{C_j} is timed or not. If $t_{C_j} \in T_C^0$, the Boolean expression for the condition c_j is implemented with a simple association of NO and/or NC contacts. This association is connected in series with an NO contact representing the rising edge or the falling edge of the corresponding sensor signal that observes e_j. When all conditions for the firing of $t_{C_j} \in T_C^0$ are satisfied, a coil associated with the binary variable B_j is energized to represent that t_{C_j} is ready to fire. It is important to remark that if $c_j = 1$ or $e_j = \lambda$, then no contacts are added to represent the condition or the event. On the other hand, if $t_{C_j} \in T_C^D$, then a timer with preset value equal to the delay time d_j must be used. After the delay time, the timer energizes an output coil TDN_j indicating that the delay time has elapsed. In this paper it is considered that the timer resets its accumulated value if its corresponding rung is opened.

Fig. 21 shows the ladder diagram of the module of firing conditions of the CIPN of Fig. 19. Notice that transitions t_{C_j} for $j = 1, \ldots, 4$ are not timed but transition t_{C_5} is timed. Consider, for instance, the non-timed transition t_{C_1}. Notice that transition t_{C_1} is enabled if $x_C(p_{C_1}) \geq 1$ and $x_C(p_{C_3}) < 2$, and t_{C_1} fires when $\uparrow S_a$ occurs. The enabling conditions are represented in the first rung of Fig. 21 with two comparison instructions associated with the markings of places p_{C_1} and p_{C_3}, represented by the integer variables X_1 and X_3, respectively, and the occurrence of $\uparrow S_a$ is verified by using an NO contact associated with the binary variable S_{ar}. The conditions for the firing of the timed transition t_{C_5} are represented in the fifth rung of Fig. 21. Notice that, the timer is energized only after the enabling conditions $x_C(p_{C_3}) \geq 2$ and $x_C(p_{C_6}) \geq 1$ become true, which are represented in the ladder diagram as $X_3 \geq 2$ and $X_6 \geq 1$.

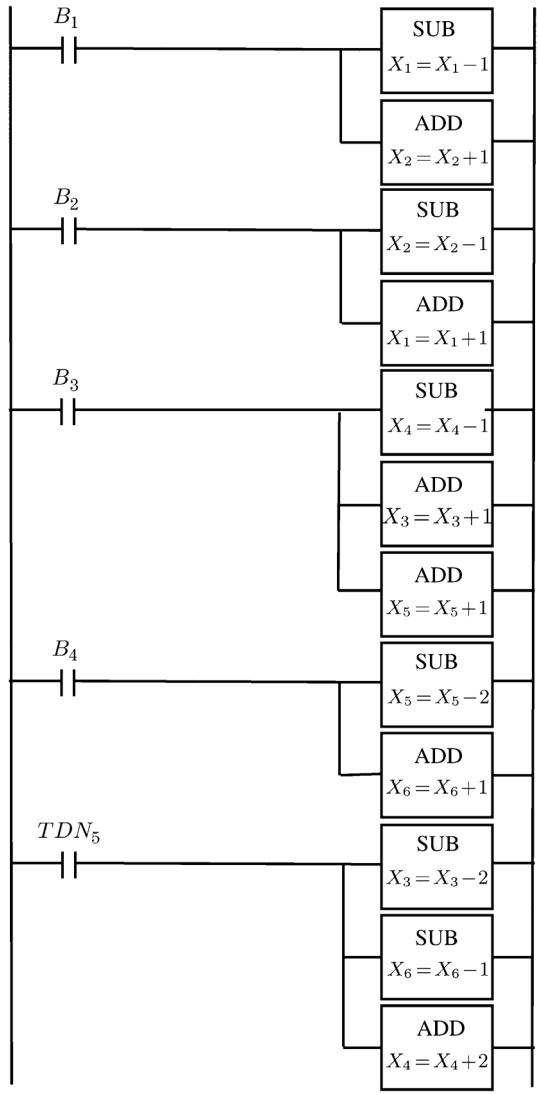

Fig. 22. Module of Petri net dynamics for the CIPN of Fig. 19.

C. The Module of Petri Net Dynamics

After a transition t_{C_j} fires, the number of tokens in the Petri net must be updated. This process is performed in the ladder diagram by the module of Petri net dynamics. This module has $|T_C|$ rungs. Each rung is associated with a transition $t_{C_j} \in T_C$ and expresses the changes in the place markings after the firing of t_{C_j}. An NO contact, associated with the binary variable B_j or with the output coil of the timer TDN_j, is used to represent that transition t_{C_j} is ready to fire. Math functions are used in series with the NO contact to represent the changes in the markings of the input and output places of t_{C_j}. The subtraction function (SUB) is used for the variables associated with input places p_{C_i} of t_{C_j}, and the weight $Pre_C(p_{C_i}, t_{C_j})$ is subtracted from the integer variable X_i that represents the number of tokens of p_{C_i}. The addition function (ADD) is used for the variables associated with output places p_{C_o} of t_{C_j}, and the weight $Post_C(t_{C_j}, p_{C_o})$ is added to the integer variable X_o that represents the number of tokens of the output place p_{C_o}.

Fig. 22 shows the ladder diagram of the module of dynamics of the Petri net depicted in Fig. 19. Each input place of a transition is associated with a SUB block and each output place is associated with an ADD block.

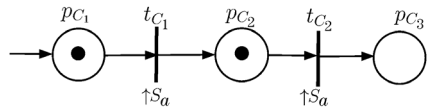

Fig. 23. Part of a CIPN with two consecutive transitions synchronized with the rising edge of the same sensor signal.

Remark 2: Notice that if a place p_{C_i} is safe then, in some cases, the math instructions ADD and SUB can be replaced with Set and Reset coils, respectively. However, in general, this procedure can lead to some unpredictable behavior. In order to illustrate this fact, consider the part of a Petri net depicted in Fig. 23. In this example, if the rungs are implemented in the order presented in Fig. 24(a), then the marking of p_{C_2} is equal to zero after the occurrence of the rising edge $\uparrow S_a$, since both transitions t_{C_1} and t_{C_2} are synchronized with the same external event. This incorrect behavior can be avoided by changing the order of the rungs in the Petri net dynamics module. However, defining the correct order can be difficult if the Petri net is complex. A simple way to overcome this problem is to use math instructions and consider integer variables instead of binary variables even in the representation of the markings of safe places. In the ladder diagram of Fig. 24(b), the marking of p_{C_2}, represented by variable X_2, is clearly equal to one after the occurrence of $\uparrow S_a$. □

D. The Initialization Module

The initialization module contains just one rung formed with an NC contact associated with an internal binary variable B_0 that, in the first scan cycle, logically energizes MOVE blocks associated with places that have tokens in the initial marking. After the first scan cycle, the NC contact is opened. It is worth remarking that there is no need to set the value zero to the variables associated with places without any initial markings since the variables are automatically initialized with zero.

Fig. 25 depicts the initialization module for the CIPN of Fig. 19 for the initial marking $\underline{x} = [1\ 0\ 0\ 2\ 0\ 0]^T$. The MOVE blocks are used in the initialization module of Fig. 25 to define the values of the integer variables X_1 and X_4, that represent the initial markings of places p_{C_1} and p_{C_4}, respectively.

E. The Module of Actions

The number of rungs in the module of actions is equal to the number of places with assigned actions in the CIPN. Since, in the CIPN defined in this paper, only impulse actions are considered, the actions must be executed only when the marking of a safe place changes from zero to one. In order to implement this behavior, a comparison instruction in series with a P contact is used to verify if the marking of a place that has an assigned action changes its logical value from zero to one. If this is the case, then an output coil is logically energized and the impulse action is executed.

The module of actions for the example of Fig. 19 is shown in Fig. 26. Notice that the ladder diagram of the module of actions

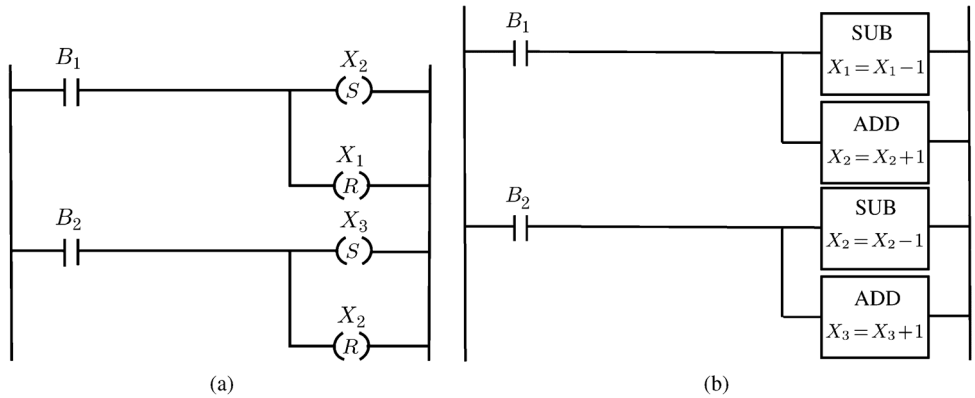

Fig. 24. (a) Incorrect module of Petri net dynamics for the CIPN of Fig. 23 using Set and Reset coils and binary variables to represent the marking of the safe places and (b) the correct module of Petri net dynamics for the CIPN of Fig. 23 using math instructions and integer variables to represent the marking of the safe places.

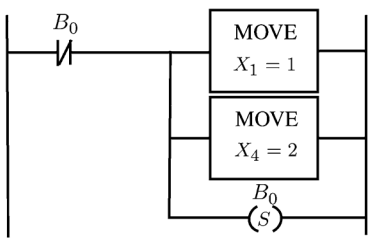

Fig. 25. Initialization module for the CIPN of Fig. 19.

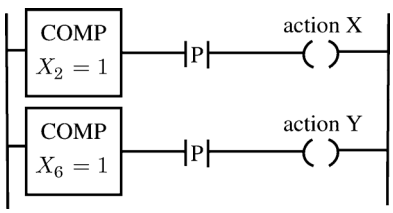

Fig. 26. Module of actions for the CIPN of Fig. 19.

has two rungs only, since the CIPN has only two places with assigned actions.

Remark 3: It is important to remark that only impulse actions are allowed in the CIPN used in this paper. Therefore, we assume here that either the plant has a device that observes the impulse action sent by the controller and executes an operation, or SET and RESET coils must be used associated with the controller outputs.

F. Organization of the Ladder Diagram

The five modules must be implemented in the same order of presentation in this paper, namely: (i) the module of external events; (ii) the module of firing conditions; (iii) the module of Petri net dynamics; (iv) the initialization module; and (v) the module of actions.

The order of the modules of the ladder code is important to avoid the avalanche effect and also to guarantee that actions defined in the initial marking are executed. The avalanche effect is avoided because the conditions for the firing of all transitions are verified first in the module of firing conditions, and only after that, the evolution of the tokens are carried out in the module of Petri net dynamics. This implementation scheme guarantees that each marking of the CIPN (even unstable markings) remains unchanged for at least one scan cycle in its ladder implementation. Therefore, only enabled transitions can fire when the associated event occurs. Another benefit of the organization of the structure of the modules proposed in this paper is that it allows the actions assigned to safe places of unstable markings of the CIPN to be executed correctly.

It is important to notice that the actions assigned to safe places that have a token in the initial marking must be executed. In order to guarantee this behavior, the initialization module is implemented after the modules of firing conditions and Petri net dynamics. Therefore, if a safe place with assigned actions is marked with a token in the initialization module, the assigned actions are executed as described by the module of actions.

Finally, it is worth remarking that, although the method proposed here, in general, generates a larger ladder code than other methods proposed in the literature, it guarantees that the intended behavior of the CIPN will be executed by its ladder implementation.

G. Size of the Ladder Diagram

Assuming that there are l distinct external events associated with the rising edge or the falling edge of sensor signals, then the maximum number of rungs in the ladder diagram obtained from the method is $(l+2|T_C|+1+|P_C|)$. Although the number of rungs could be smaller, the ladder diagram proposed in this work allows a complete visualization of the Petri net structure, and mimics its behavior. Thus, any modification in the DEC described by the CIPN can be easily implemented in the existing ladder diagram.

H. Example

We will now apply the methodology proposed here to the opening process of the plastic molding machine whose RCIPN is shown in Fig. 18. In order to make the notation simpler, we will rename the places and transitions of the RCIPN, as follows: $p_{C_1} = p_{64,86}$, $p_{C_2} = p_{87,t_{60},90,89,t_{61},91}$, $p_{C_3} = p_{92,t_{63},94}$, $p_{C_4} = p_{95,t_{65},97}$, $p_{C_5} = p_{98,t_{67},1}$, $p_{C_6} = p_{98,t_{67},100}$, $p_{C_7} = p_{99,t_{68}}$, $t_{C_1} = t_{59}$, $t_{C_2} = t_{62}$, $t_{C_3} = t_{64}$, $t_{C_4} = t_{66}$, $t_{C_5} = t_{68}^{out}$. The corresponding ladder diagram obtained by applying the conversion rule proposed in Section V is shown in Fig. 27. Notice that the modules are incomplete since the other operations

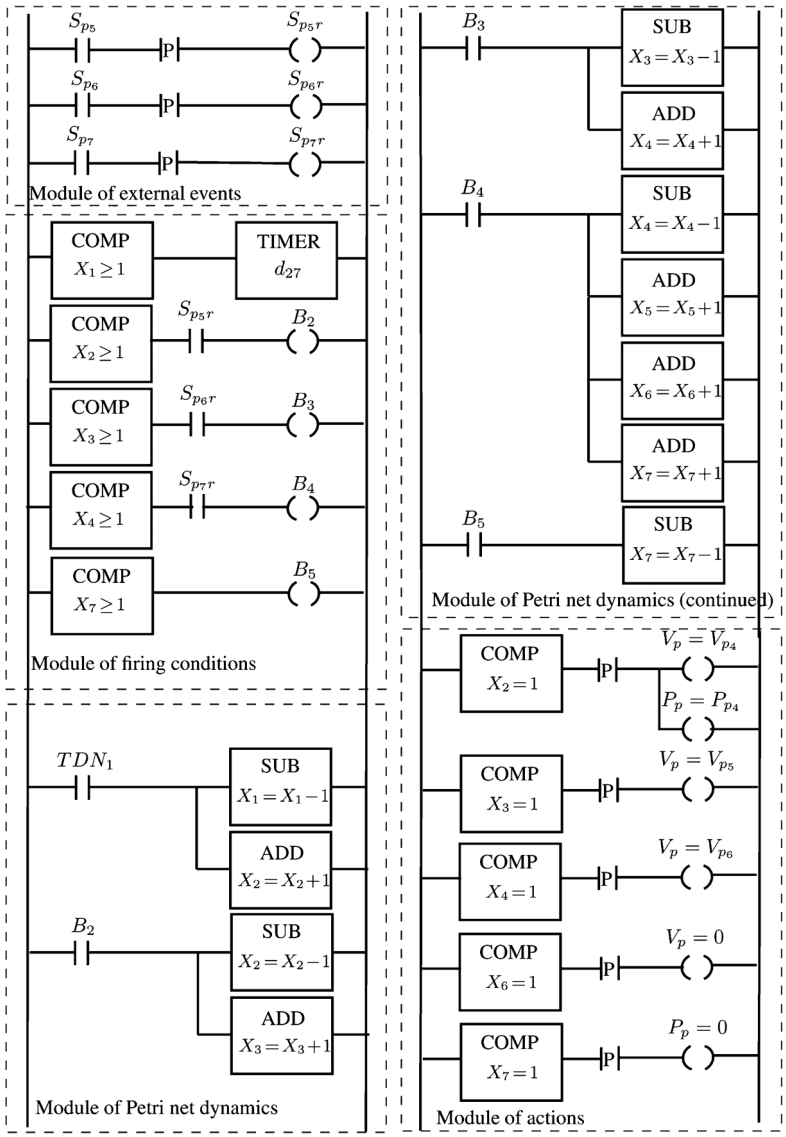

Fig. 27. Ladder diagram of the RCIPN of Fig. 18.

of the plastic molding machine must also be implemented to guarantee the correct execution of all parts of the system. It is also important to remark that there are no rungs in the initialization module related with the opening process since place p_{C_1} receives a token only when the injection process is finished. Therefore, all places of the CIPN of the opening process have zero initial marking.

VI. Conclusion

We presented in this paper a two-step approach to the implementation of DECs for closed-loop systems modeled with ELPNs using ladder diagrams, which guarantees that the controlled system has the desired closed-loop behavior described by the ELPN. The proposed transformation rules are straightforward to apply and the method presented here is expected to benefit practitioners and to allow the theory used to design DEC to be applied in the design of practical systems.

Acknowledgment

The authors would like to thank the anonymous reviewers for their comments and suggestions which helped improve the presentation and readability of this paper.

References

[1] J. L. Peterson, *Petri net theory and the modeling of systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 1981.
[2] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
[3] R. David and H. Alla, *Discrete, Continuous and Hybrid Petri Nets*. New York, NY, USA: Springer, 2005.
[4] H. Hu, M. Zhou, and Z. Li, "Supervisor optimization for deadlock resolution in automated manufacturing systems with Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 4, pp. 794–804, Oct. 2011.
[5] F. Moscato, V. Vittorini, F. Amato, A. Mazzeo, and N. Mazzocca, "Solution workflows for model-based analysis of complex systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 83–95, Jan. 2012.
[6] Y. Du, X. Li, and P. Xiong, "A Petri net approach to mediation-aided composition of web services," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 429–435, Apr. 2012.

[7] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 7, pp. 151–190, 1997.
[8] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Opt.*, vol. 25, pp. 206–230, 1987.
[9] M. V. Iordache and P. J. Antsaklis, *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. Boston, USA: Birkhauser, 2006.
[10] H. Hu, M. C. Zhou, Z. Li, and Y. Tang, "An optimization approach to improved Petri net controller design for automated manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 772–782, Jul. 2013.
[11] M. C. Zhou, F. DiCesare, and D. L. Rudolph, "Design and implementation of a Petri net based supervisor for a flexible manufacturing system," *Automatica*, vol. 28, pp. 1199–1208, 1992.
[12] R. David, "Grafcet: A powerful tool for specification of logic controllers," *IEEE Trans. Control Syst. Technol.*, vol. 3, pp. 253–268, Sep. 1995.
[13] M. C. Zhou and E. Twiss, "Design of industrial automated systems via relay ladder logic programming and Petri nets," *IEEE Trans. Syst., Man, Cybern., Part C: Appl. Revi.*, vol. 28, no. 1, pp. 137–150, Feb. 1998.
[14] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 8, no. 3, pp. 350–361, Jun. 1992.
[15] M. D. Jeng and F. DiCesare, "A review of synthesis techniques for Petri nets with applications to automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 301–312, Jan./Feb. 1993.
[16] I. Suzuki and T. Murata, "A method for stepwise refinement and abstraction of Petri nets," *J. Comput. Syst. Sci.*, vol. 27, pp. 51–76, 1983.
[17] R. David and H. Alla, "Petri nets for modeling of dynamic systems—A survey," *Automatica*, vol. 30, pp. 175–202, 1995.
[18] M. Uzam and A. H. Jones, "Discrete event control system design using automation Petri nets and their Ladder diagram implementation," *Int. J. Adv. Manuf. Technol.*, vol. 14, pp. 716–728, 1998.
[19] K. Venkatesh, M. Zhou, and R. J. Caudill, "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. Ind. Electron.*, vol. 41, pp. 611–619, Dec. 1994.
[20] S. S. Peng and M. C. Zhou, "Ladder diagram and Petri-net-based discrete-event control design methods," *IEEE Trans. Syst., Man, Cybern. —Part C: Appl. Rev.*, vol. 34, pp. 523–531, Nov. 2004.
[21] M. Fabian and A. Hellgren, "PLC-based implementation of supervisory control for discrete event systems," in *Proc. 37th IEEE Conf. Decision and Control*, 1998, pp. 3305–3310.
[22] A. Hellgren, M. Fabian, and B. Lennartson, "On the execution of sequential function charts," *Control Eng. Practice*, vol. 13, pp. 1283–1293, 2005.
[23] M. Uzam, A. H. Jones, and N. Ajlouni, "Conversion of Petri nets controllers for manufacturing systems into Ladder logic diagrams," in *Proc. IEEE Conf. Emerging Technol. Factory Autom.*, 1996, pp. 649–655.
[24] G. B. Lee, H. Zandong, and J. S. lee, "Automatic generation of Ladder diagram with control Petri net," *J. Intell. Manuf.*, vol. 15, pp. 245–252, 2004.
[25] M. V. Moreira, D. S. Botelho, and J. C. Basilio, "Ladder diagram implementation of control interpreted Petri nets: A state equation approach," in *Proc. 4th IFAC Workshop Discrete-Event Syst. Design*, 2009, pp. 75–81.
[26] I. Jimenez, E. lopez, and A. Ramirez, "Synthesis of Ladder diagrams from Petri nets controller models," in *Proc. IEEE Int. Symp. Intell. Control*, 2001, pp. 225–230.
[27] M. Uzam, "A general technique for the PLC-based implementation of RW supervisors with time delay functions," *Int. J. Adv. Manuf. Technol.*, vol. 62, pp. 687–704, 2012.
[28] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Opt.*, vol. 25, pp. 206–230, 1987.
[29] F. Charbonnier, H. Alla, and R. David, "The supervised control of discrete-event systems," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 2, pp. 175–187, 1999.
[30] F. Basile and P. Chiacchio, "On the implementation of supervised control of discrete event systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 4, pp. 725–739, Jul. 2007.
[31] K. John and M. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems*. New York, NY, USA: Springer, 2005.
[32] F. Basile, P. Chiacchio, and D. Gerbasio, "On the implementation of industrial automation systems based on PLC," *IEEE Trans. Autom. Sci. Eng.*, 2013, 10.1109/TASE.2012.2226578, to be published.
[33] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Events Systems*, 2nd ed. New York, NY, USA: Springer, 2008.
[34] K. H. Lee and J. Favrel, "Hierarchical reduction method for analysis and decomposition of Petri nets," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, pp. 272–280, Mar.-Apr. 1985.
[35] K. H. Lee, J. Favrel, and P. Baptiste, "Generalized Petri net reduction method," *IEEE Trans. Syst., Man, Cybern.*, vol. 17, pp. 297–303, Mar. 1987.
[36] K. H. Lee and J. Favrel, "Corrections to generalized Petri net reduction method," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1328–1329, Sep./Oct. 1989.
[37] G. Berthelot, "Checking properties of nets using transformations," *Lecture notes in Computer Science*, vol. 222, pp. 19–40, 1985.
[38] I. C. F. S. Telles, "A Petri net model for the automatic injection system of a plastic injection machine," (in Portuguese) M.S. thesis, COPPE—Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2007.

Marcos Vicente Moreira was born on May, 11, 1976 in Rio de Janeiro, Brazil. He received the electrical engineer degree, the M.Sc. degree and the D.Sc. degree in control from the Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 2000, 2002, and 2006, respectively.

Since 2007, he has been an Associate Professor with the Department of Electrical Engineering, Federal University of Rio de Janeiro. His main interests are multivariable control, robust control, discrete-event systems, and the development of control laboratory techniques.

João Carlos Basilio (M'12) was born on March 15, 1962 in Juiz de Fora, Brazil. He received the electrical engineering degree from the Federal University of Juiz de Fora, Juiz de Fora, Brazil, in 1986, the M.Sc. degree in control from the Military Institute of Engineering, Rio de Janeiro, Brazil, in 1989, and the Ph.D. degree in control from Oxford University, Oxford, U.K., in 1995.

He began his career in 1990 as an Assistant Lecturer at the Department of Electrical Engineering, Federal University of Rio de Janeiro, and, since 2007, has been a Senior Associate Professor of Control with the Department of Electrical Engineering. He served as the Academic Chair for the graduation course in Control and Automation from January 2005 to December 2006, and as the Chair for the Electrical Engineering Postgraduation Program from January, 2008 to February 2009, and, since June 2012, he has been the Chair for the Electrical Engineering Undergraduate Department. From September 2007 to December 2008, he spent a sabbatical leave at the University of Michigan, Ann Arbor, MI, USA. He is currently interested in discrete-event systems and in the development of control and automation laboratories and new teaching techniques.

Dr. Basilio is the recipient of the Correia Lima Medal.