Tracking Control of Electro-pneumatic Systems based on Petri Nets

Carlos Renato Vázquez[1], José Antonio Gómez-Castellanos[2] and Antonio Ramírez-Treviño[3]
[1]*Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias, Av. Ramón Corona 2514, 45019, Zapopan, Mexico*
[2]*UTZMG, Carr. Santa Cruz-San Isidro km. 45, 45640, Tlajomulco de Zuñiga, Mexico*
[3]*CINVESTAV Unidad Guadalajara, Av. del Bosque 1145, 45019, Zapopan, Mexico*

Keywords: Discrete Event Systems, Petri Nets, Automation.

Abstract: This work introduces and solves a tracking control problem for electro-pneumatic systems (*ENS*) modeled by interpreted Petri nets (*IPN*). The aim of this work is to maintain the simplicity of the specifications given by practitioners in the field of *ENS*'s and formalize the synthesis of a controller to ensure properties such as controllability, liveness and boundedness. In order to achieve this goal, this work presents the *IPN* models for *ENS* elements. The synchronous product of these modules yields in the plan model. Afterwards the synthesis of the controller is presented as an algorithm that provides both an *IPN* model of the closed-loop system and an *IPN* model of the controller, which can be translated to a Ladder Diagram for its implementation on a PLC device. The method is applied to a small *ENS* to show its efficacy.

1 INTRODUCTION

Petri nets (*PN*) are a mathematical formalism useful for modelling and analyzing discrete event systems (*DES*), such as manufacturing systems, automation systems, industrial robotics, among others. The study of *PN*'s has been particularly intensive for the synthesis of controllers and supervisors. In the literature, different supervision and control methods have been reported for *DES*'s based on Petri nets. The most studied control paradigms are:

- Supervisory-control (Ramadge and Wonham, 1987; Holloway et al., 1997; Iordache and Antsaklis, 2005). In this paradigm, the system behavior must be confined into the specification behavior (both given as languages) by means of an agent named supervisor that disabled controllable events. The synthesis of the controller consists in the computation of the *supreme controllable* language inside the specification, next, a *DES* that generates such language is obtained and used as the controller.
- Generalized mutual exclusions (Giua et al., 1992; Basile et al., 2013). In this technique, places (named monitors) and arcs are added to the *PN*, constraining the weighted sum of tokens inside certain places. In this way, the behavior of the resulting system avoids unsafe states or deadlock states; unsafe states are frequently either states in which two activities occur simultaneously and in deadlock states none transition is enabled.
- Liveness (Chen et al., 2011; Li et al., 2012). There exist several works that address the problem of controlling the system in order to guarantee its liveness. Liveness is an important property according to which, the firing of any transition in the future evolution is possible from any reachable state. In the literature, different analysis and controller synthesis methods have been proposed in order to guarantee liveness. Since the verification of liveness is a NP problem in the number of nodes, the studies frequently focus on net subclasses, such as *S*3*PR* (Ezpeleta, 1995), by using mathematical programming (Li and Liu, 2007; Chao, 2009) or structural analysis (Li and Zhou, 2008; S. Wang et al., 2012) to identify siphons that can lose tokens, and then adding monitor places to avoid that these siphons lose their tokens.

Despite the amount of works reported in the literature regarding control techniques in *PN*'s, there is a lack of theoretical developments and techniques for the synthesis of controllers for different control objectives. For instance, in the automation of industrial processes, the requirements are sequences of sensors signals, for which actuators must be executed in certain order; in the control of flexible manufacturing systems, the requirements are processed products, for which cer-

Vázquez, C., Gómez-Castellanos, J. and Ramírez-Treviño, A.
Tracking Control of Electro-pneumatic Systems based on Petri Nets.
DOI: 10.5220/0006861403440354
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 1*, pages 344-354
ISBN: 978-989-758-321-6

tain events must be executed, such as assemblies and machine loads; in the supervision of rail transport systems, the requirements are sequences of vehicle positions, for which the movement of the vehicles is enabled or disabled in a safe manner. In these applications, the system is required to be controlled in such a way that its output be equal to a reference signal. The control paradigms for *DES*'s mentioned above do not address this problem. In fact, the design of controllers for these applications is frequently based on heuristic rules developed by practitioners, without following any standard procedure that guarantee the safe operation of the closed-loop system.

For these problems, the regulation control framework was introduced and studied in (Ramírez-Prado et al., 2000; Santoyo et al., 2001; Sánchez-Blanco et al., 2004; Campos-Rodríguez et al., 2004). In these, the specification and the system to be controlled (the Plant) are interpreted Petri nets (*IPN*), in which some transitions are enabled or disable by means of the application of certain input symbols, and some places have symbols that are observable by external agents. The objective is to design a controller that, for every firing in the specification, executes a sequence of transitions in the Plant so the output symbols of the specification and the Plant become equal. In (Santoyo et al., 2001), the control method was illustrated in an automation problem, including an algorithm for translating the synthesized controller into a ladder diagram. Finally, in (Campos-Rodríguez et al., 2004) an extension of the control method was made in order to consider partial observations.

In this work, the control of electro-pneumatic systems (*ENS*) is considered by using the *IPN* paradigm. From a practitioner point of view, the specification of an *ENS* is given as a sequence of signals provided by the actuators' limit position sensors, such sequences are triggered by signal from switches or proximity sensors that are detecting the presence of parts or machine conditions. Nevertheless, in this kind of specifications, not all *ENS* signals appear, or even worst, some uncontrollable events may affect the plant, producing sensor signals that not mentioned at the specification. Hence the control techniques reported in the literature cannot handle these specifications. The approach herein presented, named *Tracking Control*, handles these specifications and synthesizes controllers capable to drive the *ENS* behavior according to the specification, guaranteeing closed-loop properties such as boundedness and liveness.

To avoid overwhelming practitioners with the formal modeling of a plant, this work presents an *IPN* model for each *ENS* component. Since all the transitions are differently labeled, then the synchronous product of these modules is merely the disjoint collection of the presented modules, simplifying the building of the plant model. The specification is given as a simple set of sequences of *IPN*'s, where places have assigned plant output symbols or are unlabeled. This specification definition follows the idea of specifications given by practitioners. The controller synthesis is presented as an algorithm that can be easily followed, providing both an *IPN* model of the closed-loop system and an *IPN* model of the controller, which can be translated to a Ladder Diagram for its implementation on a PLC device.

This paper is organized as follows: in Section *II* an overview of *IPN*'s is presented. In Section *III*, *IPN* models for typical electron-pneumatic components, and specifications, are introduced. In Section *IV*, the control synthesis algorithm is introduced, and some properties of the closed-loop system are presented. In Section *V*, the introduced concepts are illustrated through a case study. Conclusions and future work are presented in Section *VI*.

2 BASIC CONCEPTS

In this section, the *PN* and *IPN* definitions and some basic concepts are recalled (for more details see (David and Alla, 2010)).

2.1 Petri Nets

Definition 1. *A Petri net (PN) structure* $\mathcal{N}$ *is a bipartite digraph represented by the 4-tuple* $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$ *where* $P = \{p_1, p_2, ..., p_n\}$ *is the finite set of places,* $T = \{t_1, t_2, ..., t_m\}$ *is the finite set of transitions,* **Pre** *and* **Post** *are* $|P| \times |T|$ *matrices representing the weighted (nonnegative integer number) arcs going from places to transitions and from transitions to places, respectively. The incidence matrix of* $\mathcal{N}$ *is defined as* $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$.

Let $x \in P \cup T$ be a node of $\mathcal{N}$. The input set of x, denoted by ${}^\bullet x$, is defined as ${}^\bullet x = \{x_i \in P \cup T |$ there exists an arc from x_i to $x\}$. Similarly, the output set of a node x, denoted by $x^\bullet$, is defined as $x^\bullet = \{x_i \in P \cup T |$ there exists an arc from x to $x_i\}$.

A *PN* is a *state machine* if each transition has only one input and one output place, i.e., $\forall t \in T$ $|{}^\bullet t| = |t^\bullet| = 1$ and it is strongly connected.

Definition 2. *A PN system is the pair* $\langle \mathcal{N}, \mathbf{M}_0 \rangle$, *where* $\mathcal{N}$ *is a PN structure. The* marking *function* $M : P \to \mathbb{Z}^+$ *is a mapping from each place to the nonnegative integers representing the number of tokens residing inside each place. The marking of a PN is*

expressed as a column vector $\mathbf{M}$ *of length* $|P|$. $\mathbf{M}_0$ *is the initial marking distribution.*
The marking distribution evolves according to the firing of transitions. A transition t_j *is enabled at marking* $\mathbf{M}_k$ *iff* $\forall p_i \in {}^\bullet t_j$, $\mathbf{M}_k(p_i) \geq \mathbf{Pre}(p_i,t_j)$, *this is denoted as* $\mathbf{M}_k \xrightarrow{t_j}$. *A transition* t_j *can fire if it is enabled. The firing of an enabled transition* t_j *reaches a new marking* $\mathbf{M}_{k+1}$ *that can be computed with the so-called PN* fundamental equation

$$\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{C}\mathbf{v}_k$$

where $\mathbf{v}_k(i) = 0$ *for* $i \neq j$ *and* $\mathbf{v}_k(j) = 1$. *This is denoted as* $\mathbf{M}_k \xrightarrow{t_j} \mathbf{M}_{k+1}$.

Graphically, places are represented by circles, transitions by rectangles, arcs by arrows, and tokens are represented as dots or positive integer numbers inside places.

Definition 3. *A sequence of transitions* $\sigma = t_i t_j, ..., t_k$ *of a PN system* $\langle \mathcal{N}, \mathbf{M}_0 \rangle$ *such that* $\mathbf{M}_0 \xrightarrow{t_i} \mathbf{M}_1 \xrightarrow{t_j}, ..., \mathbf{M}_w \xrightarrow{t_k}$ *is said to be fireable or it is said that* σ *is a firing transition sequence.*

The marking $\mathbf{M}'$ reached after the firing of σ at a marking $\mathbf{M}$ can be computed by

$$\mathbf{M}' = \mathbf{M} + \mathbf{C}\vec{\sigma}$$

where $\vec{\sigma}$ is a vector, named Parikh vector, defined as a column vector of size $|T|$ such that $\vec{\sigma}(j) = k$ if t_j is fired k times in the sequence σ. This is denoted as $\mathbf{M} \xrightarrow{\sigma} \mathbf{M}'$, $\mathbf{M}'$ is said to be *reachable* from $\mathbf{M}$.

The reachability set of a *PN* is the set of all the reachable markings from $\mathbf{M}_0$, and it is denoted as $R(\mathcal{N}, \mathbf{M}_0)$.

A *PN* system is said to be *bounded* if there exists a finite number k such that, for all the reachable markings, each place has at most k tokens, i.e., $\forall \mathbf{M} \in R(\mathcal{N}, \mathbf{M}_0)$ $\forall p \in P$ it holds $\mathbf{M}(p) \leq k$. A *PN* system is said to be *safe* if it is bounded with $k = 1$.

A *PN* system is said to be *live* if for any transition $t_j \in T$ and any reachable marking $\mathbf{M} \in R(\mathcal{N}, \mathbf{M}_0)$ there exists a fireable sequence σ such that $\mathbf{M} \xrightarrow{\sigma} \mathbf{M}'$ and t_j is enabled at $\mathbf{M}'$.

2.2 Interpreted Petri Nets

In this work, the *Interpreted Petri net* model will be used, which is an extension to *PN*'s allowing to represent input and output symbols (Ramírez-Trevino et al., 2003).

Definition 4. *An Interpreted Petri net (IPN) system is a 6-tuple* $Q = \langle \mathcal{N}, \mathbf{M}_0, \Sigma_I, \Sigma_O, \lambda, \varphi \rangle$ *where:*

- $\langle \mathcal{N}, \mathbf{M}_0 \rangle$ *is a Petri net system;*
- Σ_I *is the input alphabet of the Petri net system, where each element of the set* Σ_I *is an input symbol;*
- $\lambda : T \to \Sigma_I \cup \{\varepsilon\}$ *is the labeling function of transitions with the restriction that nondeterministic inputs are not allowed, i.e.,* $\forall t_j, t_k \in T, j \neq k$, *if* $\mathbf{Pre}(p_i,t_j) = \mathbf{Pre}(p_i,t_k) \neq 0$ *and both* $\lambda(t_j), \lambda(t_k) \neq \varepsilon$, *then* $\lambda(t_j) \neq \lambda(t_k)$. *Here,* ε *represents a system's internal event;*
- Σ_O *is the output alphabet of the Petri net system, where each element of the set* Σ_O *is an output symbol;*
- $\varphi : P \to \Sigma_O \cup \{\varepsilon\}$ *is an output function that associates places to output symbols.* ε *represents a symbol that is not available to an external observer.*

The function φ can be represented by a $|\Sigma_O| \times |P|$ matrix φ, in which $\varphi(i,j) = 1$ if the place p_j is associated to the i-th output symbol and $\varphi(i,j) = 0$ otherwise.

Here it is assumed that each place generates at most one output symbol. A place $p \in P$ is said to be measurable if $\varphi(p) \neq \varepsilon$, otherwise, it is nonmeasurable. A transition t is said to be controllable if $\lambda(t) \neq \varepsilon$, otherwise it is uncontrollable.

The evolution of an *IPN* is similar to that of the *PN* system with the addition that a symbol $a \in \Sigma_I$ is *indicated* if it is activated by an external device (for instance a controller or a user). The following aspects are also considered for the transitions firing.

- If $\lambda(t_j) = a_i \neq \varepsilon$ is indicated and t_j is enabled then t_j *must* fire. If t_j is enabled by the marking, but the symbol $\lambda(t_j)$ is not indicated, then t_j cannot fire. If $\lambda(t_j) = \varepsilon$ and t_j is enabled, then t_j *can* fire at any moment.
- At any reachable marking $\mathbf{M}_k$, an external observer reads the symbols associated to the marked places.

In this work, it will be assumed that the *IPN*'s are event-detectable (Ramírez-Trevino et al., 2003), a property that is recalled as follows:

Definition 5. *An IPN* $\langle \mathcal{N}, \mathbf{M}_0, \Sigma_I, \Sigma_O, \lambda, \varphi \rangle$ *is said to be event-detectable if the firing of any* $t_i \in T$ *can be detected by a change on the output symbols and can be distinguished from the firing of other transitions, i.e.,* $\forall t_i \in T$

- $\varphi C(\bullet, t_i) \neq \mathbf{0}$
- $\forall t_j \in T \setminus \{t_i\}$, $\lambda(t_i) \neq \lambda(t_j)$ *or* $\varphi\mathbf{C}(\bullet, i) \neq \varphi\mathbf{C}(\bullet, j)$.

IPN models for complex systems can be built from *IPN* submodels of *independent* components

(i.e., the submodels do not share places, transitions or symbols). This is performed by the synchronous product defined as follows:

Definition 6. *Let* $Q^1 = \langle \mathcal{N}^1, \mathbf{M}_0^1, \Sigma_I^1, \Sigma_O^1, \lambda^1, \varphi^1 \rangle$ *and* $Q^2 = \langle \mathcal{N}^2, \mathbf{M}_0^2, \Sigma_I^2, \Sigma_O^2, \lambda^2, \varphi^2 \rangle$ *be two IPN models, where* $\mathcal{N}^1 = \langle P^1, T^1, \mathbf{Pre}^1, \mathbf{Post}^1 \rangle$ *and* $\mathcal{N}^2 = \langle P^2, T^2, \mathbf{Pre}^2, \mathbf{Post}^2 \rangle$. *The synchronous product results in an IPN* $Q^3 \langle \mathcal{N}^3, \mathbf{M}_0^3, \Sigma_I^3, \Sigma_O^3, \lambda^3, \varphi^3 \rangle$, *where where* $\mathcal{N}^3 = \langle P^3, T^3, \mathbf{Pre}^3, \mathbf{Post}^3 \rangle$, *denoted as* $Q^3 = Q^1 || Q^2$. *The IPN* Q^3 *is computed as follows:*

- *The net structure is computed as follows:* $P^3 = P^1 \cup P^2$ *and* $T^3 = T^1 \cup T^2$, *enumerating first the nodes in* $\mathcal{N}^1$. *Next,* $\mathbf{Pre}^3 = diag(\mathbf{Pre}^1, \mathbf{Pre}^2)$ *and* $\mathbf{Post}^3 = diag(\mathbf{Post}^1, \mathbf{Post}^2)$, *where* $diag(\bullet, \bullet)$ *is a matrix built with the argument matrices as diagonal blocks, and other entries are null.*
- *The initial marking is* $\mathbf{M}_0^3 = [(\mathbf{M}_0^1)^T, (\mathbf{M}_0^2)^T]^T$.
- *The input and output alphabets are* $\Sigma_I^3 = \Sigma_I^1 \cup \Sigma_I^2$ *and* $\Sigma_O^3 = \Sigma_O^1 \cup \Sigma_O^2$, *respectively.*
- *The input function is defined as:* $\forall t \in T^3$ *set* $\lambda^3(t) = \lambda^i(t)$, *where* t *is a node of* T^i *with* $i \in \{1,2\}$. *Similarly, the output function is defined as:* $\forall p \in P^3$ *set* $\varphi^3(p) = \varphi^i(p)$, *where* p *is a node of* P^i *with* $i \in \{1,2\}$.

This synchronous product is compatible with the previous reported in the literature since the *IPN* modules used in next sections are label disjoint (i.e. two different modules do not share input symbols).

3 IPN PLANT AND SPECIFICATION MODELS

In the following subsection, a library of *IPN* models for the most frequently used components in *ENS*'s is proposed; it includes push buttons, selectors, proximity sensors, electro-pneumatic valves and actuators. In the next subsection , the plant model is built using a bottom-up approach. It is very simple and appealing from a practitioner point of view. Later, in Subsection 3.3, the specification model is defined.

3.1 Electro-pneumatic Component Models

Figure 1 shows *IPN* models for different kind of switches. In these, output symbols are defined in the places that represent a state in which the switch conducts current. Figure 1.(a) represents a normally open (NO) switch, whereas Figure 1.(b) represents a normally closed (NC) switch. These switches can be either push-buttons or switches (in such case, t_0 represents the event of "push" and t_1 the event of "release"), or lock-buttons (in such case, t_0 represents the event of "push for the first time" and t_1 the event of "push for the second time"). Figure 1.(c) represents a single-pole two-throw switch, in one state the current is conducted to throw *A*, in the other state the current is conducted to throw *B*. Finally, Figure 1.(d) represents a selector between three throws, *A*, *B* or *C*.

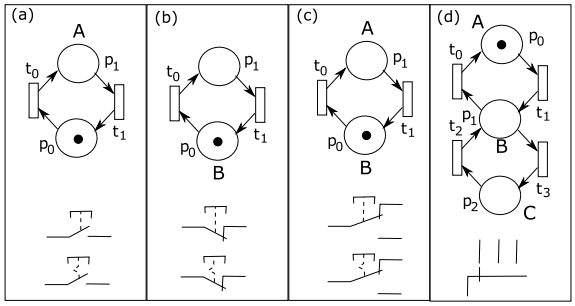

Figure 1: *IPN* models for different kinds of switches.

Figure 2 shows the electro-pneumatic symbols for different kind of proximity sensors (DIN standard, as used in electro-pneumatic diagrams) and its *IPN* model. All of them are proximity sensors commonly used in *ENS*'s to detect a limit position of an actuator, the presence of a part or a machine condition. These sensors are the pressure sensor (Figure 2.(a)), the capacitive sensor (Figure 2.(b)), the inductive sensor (Figure 2.(c)), the magnetic sensor (Figure 2.(d)), and the optical sensor (Figure 2.(e)). Frequently, these sensors provide two output signals (in Figure 2 only one terminal is drawn), one NO and one NC to detect the presence and absence of parts, respectively. For that reason, the *IPN* model includes two output symbols, *B* for NC and *A* for NO. Transition t_0 represents the event "a part is detected" and t_1 represents the event "a part is not detected". Reed magnetic sensors (not shown) are used to detect limit positions of pneumatic actuators, these behave as NO switches, thus, the model of Figure 1.(a) should be used for these sensors.

Frequently, pneumatic actuators and valves are used in usual assemblies, the most common are shown in Figure 3 and Figure 4. Figure 3 shows a vacuum actuator assembly, consisting of a 3-ways 2-positions valve, a Venturi nozzle to produce vacuum, a suction cup and a vacuum sensor, which provides an activation signal when vacuum is detected (when a part is grasped). The *IPN* model of the vacuum assembly represents the valve (nodes p_3, p_4, t_3 and t_4) and the actuator (nodes p_0, p_1, p_2, t_0, t_1 and t_2). Place p_0 represents the state in which vacuum is not activated, thus the sensor provides a signal *B*; place p_1 repre-

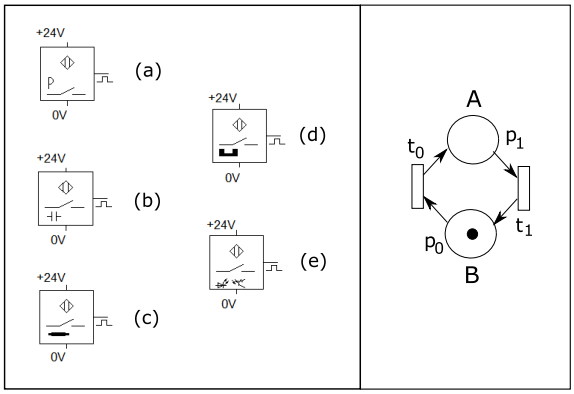

Figure 2: *IPN* models for different proximity sensors.

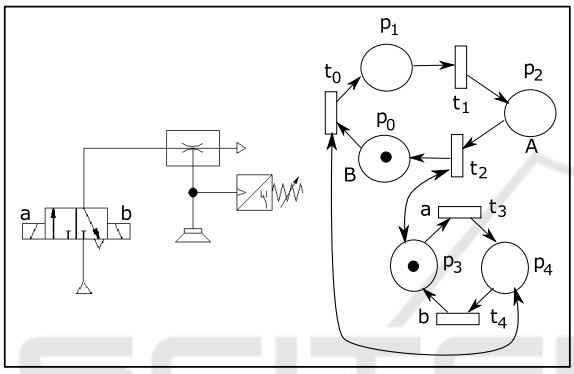

Figure 3: *IPN* model for vacuum assembly.

sents a transition state, and place p_2 represents the state in which a part is grasped, thus the vacuum sensor provides the output signal A. The activation of the valve solenoids are the only controllable events, represented by symbols a and b at transitions t_3 and t_4, respectively.

Figure 4 represents the most typical valve-actuator assemblies: double acting actuator controlled by a 5-ways 2-positions valve (a), spring return actuator controlled by a 3-ways 2-positions spring return valve (b), and a rotary actuator controlled by a 5-ways 2-positions valve (c). Pneumatic-grippers (not shown) are usually driven by double acting actuators controlled by 5-ways 2-positions valves, thus they are similar to Figure 4.(a). The same *IPN* model is valid for all the assemblies. In this, the valve is represented by nodes p_3, p_4, t_4 and t_5, and the actuator is represented by nodes p_0, p_1, p_2, t_0, t_1, t_2 and t_3. In the assemblies, sensors are located to detect the limit positions, providing the output signals represented by A and B, for the leftmost and rightmost positions, respectively. On the other hand, the activation of the valve solenoids are the only controllable events, represented by symbols a and b (to move to the left and right, respectively) at transitions t_4 and t_5, respectively. For the case of the spring return valve at Figure 4.(b), the symbol a is

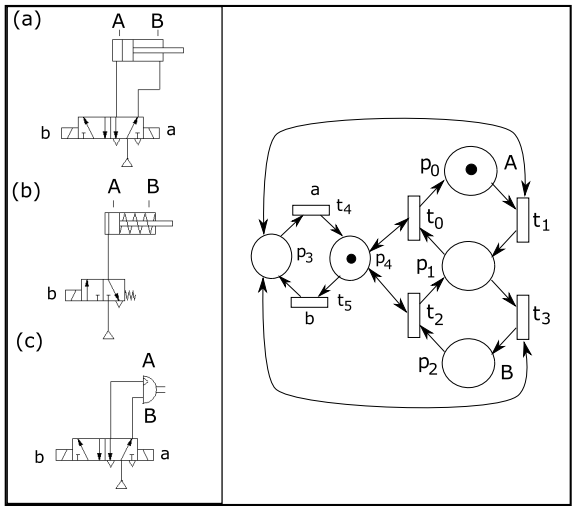

Figure 4: *IPN* models for electro-pneumatic assemblies.

defined as the logical negation of b (i.e., the absence of event b).

In addition to the introduced *IPN* models, the following control function must be stated. It indicates the controllable transition that is needed to turn on a particular sensor (i.e. to reach a marking where a measurable place is marked). This is formalized in the sequel:

Definition 7. *The function $t_{\rightarrow p} : P \rightarrow T$, which indicates the controllable transition (if it exists) whose firing leads to the marking of p, is defined as follows:* $\forall p \in P$,

$$t_{\rightarrow p}(p) = \begin{cases} t & \text{if } \varphi(p) \neq \varepsilon \text{ and there exists a directed path from a controllable transition } t \text{ to } p, \text{ which does not contain any other controllable transition or measured place. If more than one of such paths exists, then any of them can be used,} \\ t' & \text{where } t' \in^{\bullet} p, \text{ if either } \varphi(p) = \varepsilon \text{ or there does not exist a path from a controllable transition to } p \text{ not containing other measured places.} \end{cases}$$

The models of figs. 1-2 do not contain controllable transitions, thus $t_{\rightarrow p}$ is defined as in the second statement of previous definition. For the models of figs. 3 and 4, $t_{\rightarrow p}$ is defined as follows:

Model in Figure3	Model in Figure4
$t_{\rightarrow p}(p_0) = t_4$	$t_{\rightarrow p}(p_0) = t_4$
$t_{\rightarrow p}(p_1) = t_0$	$t_{\rightarrow p}(p_1) = t_1$
$t_{\rightarrow p}(p_2) = t_3$	$t_{\rightarrow p}(p_2) = t_5$
$t_{\rightarrow p}(p_3) = t_4$	$t_{\rightarrow p}(p_3) = t_5$
$t_{\rightarrow p}(p_4) = t_3$	$t_{\rightarrow p}(p_4) = t_4$

3.2 Building the Plant Model

Following the Control Theory terminology, the system to be controlled is named the *Plant*. A plant *IPN* model for an *ENS* can be built by using the synchronous product on the *IPN* models of the components. Since the labels of the transitions are module disjoint, then the plant is a disjoint collection of component submodels (Figure 1-4). In other words, the plant model is a collection of disjoint *ENS IPN* modules.

Definition 8. *The plant is a safe event-detectable IPN* $Q^p = \langle \mathcal{N}^p, \mathbf{M}_0^p, \Sigma_I^p, \Sigma_O^p, \lambda^p, \varphi^p \rangle$, *where* $\mathcal{N}^p = \langle P^p, T^p, \mathbf{Pre}^p, \mathbf{Post}^p \rangle$, *that models the DES to be controlled. For an ENS with components* $\{c_1, .., c_n\}$, *its plant IPN model can be obtained as follows:*

- *For each component* c_i, *define its IPN model* $Q^{c_i} = \langle \mathcal{N}^{c_i}, \mathbf{M}_0^{c_i}, \Sigma_I^{c_i}, \Sigma_O^{c_i}, \lambda^{c_i}, \varphi^{c_i} \rangle$, *as shown in the figs. 1-4, where* $\mathcal{N}^{c_i} = \langle P^{c_i}, T^{c_i}, \mathbf{Pre}^{c_i}, \mathbf{Post}^{c_i} \rangle$, *and define its function* $t^{c_i}_{\to p}$. *Different input and output alphabets must be defined for different components, i.e.,* $\Sigma_I^{c_i} \cap \Sigma_I^{c_j} = \emptyset$ *and* $\Sigma_O^{c_i} \cap \Sigma_O^{c_j} = \emptyset$ *if* $j \neq i$.
- *The plant's IPN model is computed as*

$$Q^p = Q^{c_1} || ... || Q^{c_n}$$

- *The function* $t^p_{\to p}$ *of the plant is computed as:* $\forall p \in P^p$ *set* $t^p_{\to p}(p) = t^{c_i}_{\to p}(p)$, *where p is a node of* P^{c_i}.

Moreover, $\Sigma_O^{act} \subseteq \Sigma_O^p$ *is defined as the subset of output symbols related to actuators (symbols from models of Figure 3-4).*

3.3 Specification Model

In *ENS*'s, specifications are a collection of plant output signals sequences. In them, neither the occurrences of internal events nor the reachability of silent states are specified.

Each specification sequence must include a strict sequences of actuators' output symbols that are required to occur in the plant (sensor or switch plant output symbols are allowed to occur in any order). In these sequences, a place p must be defined for each instance of a required plant output symbol o; this place must have associated the symbol o. In addition, specification sequences may have extra places with associated symbols from sensors and switches, defining guards for the occurrence of sequences, which can be implemented as either a place in the sequence or as self-loop places.

In this work, the specification is represented as an *IPN* describing output sequences and/or selections between output sequences. This is formalized as follows:

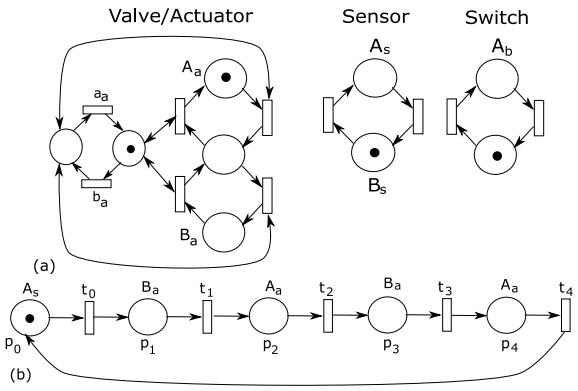

Figure 5: *IPN* models of plant (a) and specification (b) of example 1.

Definition 9. *An specification is a safe and live state machine IPN* $Q^s = \langle \mathcal{N}^s, \mathbf{M}_0^s, \Sigma_I^s, \Sigma_O^s, \lambda^s, \varphi^s \rangle$, *with additional marked self-loop places (a self-loop is a place p with* $Pre(p,t) = Post(p,t) = 1$ *for a particular transition t). All the transitions are controllable. The output alphabet of the specification is equal to the output alphabet of the plant, i.e.,* $\Sigma_O^s = \Sigma_O^p$.

Example 1. *For instance, consider an ENS consisting of an assembly valve/double acting actuator (the one shown in Figure 4.a), a proximity sensor (shown in Figure 2) and a push button (shown in Figure 1.(a)). An specification for this system indicates that when the proximity sensor detects a part the actuator must complete two operation cycles (extending/returning).*
The plant's model is shown in Figure 5.(a), it is the collection of the IPN models of each ENS component, as expected. The specification is depicted in Figure 5.(b), notice that it is a sequence starting by symbol A_s *and continuing with the two working cycles of the actuator, as the required specification of the example indicates.*
Since the specification does not mention the plant output symbols B_s *and* A_b, *then they are not relevant for the specification and can occur in any order during the plant's operation. Moreover the sensor symbol* A_s *could occur in any order during the plant's operation, however, the symbol* A_s *is required to be present for starting the sequence.*
If the pressing of the push button is additionally required to start the sequence, a self-loop place with one token can be added to t_0 *with the symbol* A_b, *associated to the button's closed position (i.e., a place p connected with input an output arcs to* t_0, *having one token and the symbol* A_b*).*

4 CONTROL SYNTHESIS ALGORITHM

The goal of the control of *ENS*'s is to drive the plant in such a way that a required sequence of actuators' movements is achieved, by activating the corresponding valves, in response to the activation of sensors and switches. At the *IPN* level this means a proper synchronization of the plant and the specification. The following algorithm provides a formal solution for the tracking control problem whenever the plant and the specification are modelled according to Definitions 8 and 9. In the algorithm it is used the property $\varphi(p) \neq \varepsilon$ iff p is measured, similarly, $\lambda(t) \neq \varepsilon$ iff t is controllable.

Algorithm 4.1: Calculation of the closed-loop *IPN* and the controller.

Input *IPN* models of the plant Q^p and the specification Q^s, and the function $t_{\to p}$.
Output *IPN* models of the closed-loop system Q^{cl} and the controller Q^c.

% First, relabel the specification's places as follows:
Let $\Sigma_O^s = \{o_1, ..., o_m\}$ be the output alphabet of Q^s. Define a new output alphabet for Q^s as $\Sigma_O^{s\prime} = \{o_1', .., o_m'\}$ and a bijective function $\Pi : \Sigma_O^s \to \Sigma_O^{s\prime}$ such that $\Pi(o_i) = o_i'$.
Define a new output function $\varphi^{s\prime}$ for the specification as follows: $\forall p \in P^s$,

$$\varphi^{s\prime}(p) = \begin{cases} \Pi(\varphi^s(p)) & \text{if } \varphi^s(p) \neq \varepsilon \\ \varepsilon & \text{otherwise} \end{cases}$$

% Next, define mirror places in the specification associated to places sharing the same output symbol as follows:
Initialize $P^{s\prime} = P^s$, $\mathbf{M}_0^{s\prime} = \mathbf{M}_0^s$, $\mathbf{Pre}^{s\prime} = \mathbf{Pre}^s$ and $\mathbf{Post}^{s\prime} = \mathbf{Post}^s$.
for every $p_i \in P^s$ such that $\varphi^{s\prime}(p_i) \neq \varepsilon$ **do**
  if $\exists p_j \in P^s$ s.t. $\varphi^{s\prime}(p_i) = \varphi^{s\prime}(p_j)$ **then**
    Define the set of specification places with the same output symbol as $[p_i] = \{p_j \in P^s | \varphi^{s\prime}(p_i) = \varphi^{s\prime}(p_j)\}$. Add a place p_i' to $P^{s\prime}$. Connect p_i' to other nodes in such a way that, $\forall t_j \in T^s$, $\mathbf{Pre}^{s\prime}(p_i', t_j) = \sum_{p_s \in [p_i]} \mathbf{Pre}^{s\prime}(p_s, t_j)$ and $\mathbf{Post}^{s\prime}(p_i', t_j) = \sum_{p_s \in [p_i]} \mathbf{Post}^{s\prime}(p_s, t_j)$.
    Define the initial marking of p_i' as $\mathbf{M}_0^{s\prime}(p_i') = \sum_{p_s \in [p_i]} \mathbf{M}_0^s(p_s)$. Therefore, p_i' is marked iff any place $p \in [p_i]$ is marked.
    Define the output symbol $\varphi^{s\prime}(p_i') = \varphi^{s\prime}(p_i)$. Next, eliminate the output symbols from any place in $[p_i]$.
  end if
end for
Initialize the closed-loop system as $Q^{cl} = Q^p || Q^{s\prime}$, where $Q^{s\prime} = \langle \mathcal{N}^{s\prime}, \mathbf{M}_0^{s\prime}, \Sigma_I^s, \Sigma_O^{s\prime}, \lambda^s, \varphi^{s\prime} \rangle$ and $\mathcal{N}^{s\prime} = \langle P^{s\prime}, T^s, \mathbf{Pre}^{s\prime}, \mathbf{Post}^{s\prime} \rangle$.
% Next, add bidirectional arcs to Q^{cl}, between specification's places and plant's controllable transitions, and between plant's measured places and specification's transitions, as follows:
for every $p_i \in P^s$ such that $\varphi^{s\prime}(p_i) \neq \varepsilon$ **do**
  Let $p_p \in P^p$ be such that $\varphi^p(p_p) = \varphi^{s\prime}(p_i)$. Let $t_j = \mathrm{t}_{\to p}(p_p)$.
  if $\lambda^p(t_j) \neq \varepsilon$ **then**
    Define a bidirectional arc between p_i and t_j, i.e., set $\mathbf{Pre}^{cl}(p_i, t_j) = \mathbf{Post}^{cl}(p_i, t_j) = 1$.
  end if
  for every transition $t_u \in p_i^\bullet$ **do**
    Define a bidirectional arc between p_p and t_u, i.e., set $\mathbf{Pre}^{cl}(p_p, t_u) = \mathbf{Post}^{cl}(p_p, t_u) = 1$.
  end for
end for
The resulting *IPN* Q^{cl} represents the closed-loop system.
Initialize the *IPN* of the controller as $Q^c = Q^{cl}$.
Eliminate from Q^c the plant's nodes that are only connected to other plant's nodes, i.e., eliminate places $p \in P^p$ s.t. $\mathbf{Pre}^c(p,t) = \mathbf{Post}^c(p,t) = 0$ $\forall t \in T^s$, and eliminate transitions $t \in T^p$ s.t. $\mathbf{Pre}^c(p,t) = \mathbf{Post}^c(p,t) = 0$ $\forall p \in P^{s\prime}$.
The resulting *IPN* Q^c represents the controller.

Remark 1. *Once the controller Q^c is computed by the Algorithm 4.1, it can be translated to a Ladder Diagram as in (Santoyo et al., 2001) for its implementation in a PLC.*

Example 2. *Consider the plant of Figure 5.(a) and the specification of Figure 5.(b). By applying the Algorithm 4.1, the closed-loop model depicted in Figure 6 is obtained. In this, the plant and specification are drawn in solid line, whereas the nodes and arcs added by the Algorithm 4.1 are drawn in dashed line. In particular, steps 3-4 relabel the specification symbols, from A_s, A_a and B_a to A_s', A_a' and B_a', respectively. Notice that there are two places with the symbols A_a' and B_a', then steps 5-12 define the new places drawn in dashed line with their input and output arcs, one for the symbol A_a' and another for the symbol B_a'. The symbols in brackets $\{\bullet\}$ represent symbols that should be removed at step 9, but they are kept in Figure 6 for this explanation. Notice that the place with symbol A_a' (resp. B_a') is marked iff any of the places with symbol $\{A_a'\}$ (resp. $\{B_a'\}$) is marked, that is why they are called mirror places. Steps 15-18 add bidirectional arcs from the places with symbols A_a' and B_a' to the plant's controllable transitions with*

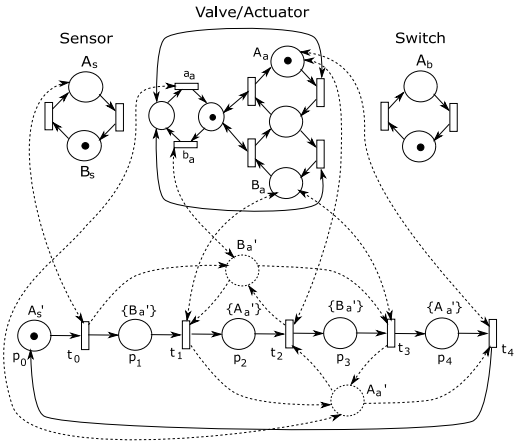

Figure 6: *IPN* closed-loop system of example 2.

symbols a_a and b_a, respectively. Steps 19-21 add bidirectional arcs from the plant's measured places with symbols A_a and B_a to the output transitions of the new places with symbols A_a' and B_a', respectively. A simulation of the closed-loop system can show that the Plant evolves following the specification, i.e., it performs a double cycle after the sensor detects a part. Moreover, notice that the sensor and switch can change their states during the actuator's movement, in accordance to the explanation of example 1.

The controller Q^c, can be obtained from the closed-loop model by eliminating the plant's nodes that are only connected to other plant's nodes. The resulting IPN can be translated to the Ladder Diagram shown in Figure 7, by following the procedure explained in (Santoyo et al., 2001), in which an internal coil is defined for each non-plant place (a coil is ON iff the corresponding place has a token), a network is defined for the places' initial condition which is executed after pressing a reset push-button, a network is added for each specification transition (Pre arcs are translated to NO switches and Reset *coils, Post arcs are translated to* Set *coils), and other networks are added for defining the PLC inputs and outputs.*

4.1 Properties of the Closed-Loop System

The Algorithm 4.1 ensures that the closed-loop system exhibits important good properties. This is formally introduced in the following proposition.

Proposition 1. *Consider a plant IPN model Q^p and an specification IPN model Q^s, as in Definitions 8 and 9. Consider the IPN closed-loop model $Q^{cl} = \langle \mathcal{N}^{cl}, \mathbf{M}_0^{cl}, \Sigma_I^{cl}, \Sigma_O^{cl}, \lambda^{cl}, \varphi^{cl} \rangle$ as computed by the Algorithm 4.1. The following statements hold:*

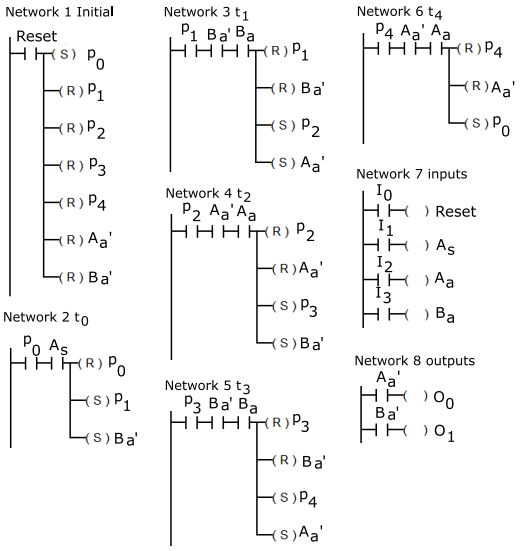

Figure 7: Ladder Diagram for the controller Q^c of example 2. The inputs I_0, I_1, I_2 and I_3 of the PLC are connected to a NO reset push-button, the NO terminal of the proximity sensor A_s, the actuator's limit sensor A_a and the actuator's limit sensor B_a, respectively. The outputs O_0 and O_1 of the PLC are connected to the electro-valve's coils corresponding to a_a and b_a, respectively.

1. *The Algorithm 4.1 is well defined , i.e., all the required information is available and the resulting models Q^{cl} and Q^c are IPN's.*
2. *Q^{cl} is bounded.*
3. *The plant is connected to the specification only through measured sensors and controllable transitions. Formally, for any $p \in P^p$ and $t \in T^{cl} \setminus T^p$, if $\varphi^p(p) = \varepsilon$ then $\mathbf{Pre}^{cl}(p,t) = \mathbf{Post}^{cl}(p,t) = 0$, else $\mathbf{Pre}^{cl}(p,t) = \mathbf{Post}^{cl}(p,t)$. For any $t \in T^p$ and $p \in P^{cl} \setminus P^p$, if $\lambda^p(t) = \varepsilon$ then $\mathbf{Pre}^{cl}(p,t) = \mathbf{Post}^{cl}(p,t) = 0$, else $\mathbf{Pre}^{cl}(p,t) = \mathbf{Post}^{cl}(p,t)$.*
4. *The plant is only allowed to evolve in order to reach the specification symbol. Formally, for any reachable marking $M_i \in R(\mathcal{N}^{cl}, \mathbf{M}_0^{cl})$, if there exists $t_i \in T^s$ such that $M_i \xrightarrow{t_i} M_j$, where $M_i(p_j) = 0$ and $M_j(p_j) > 0$ for a place $p_j \in P^s$ with $\varphi^s(p_j) \in \Sigma_O^{act}$, then there exists a fireable sequence $\sigma_i = t_i^1 t_i^2 ... t_i^r$, describing a marking trajectory $M_j \xrightarrow{t_i^1} M_j^1 \xrightarrow{t_i^2} M_j^2 ... \xrightarrow{t_i^r} M_j^r$, such that:*
   (a) *$t_i^1, t_i^2, ..., t_i^r \in T^p$,*
   (b) *$M_j^r(p_s) > 0$, where $p_s \in P^p$ is such that $\varphi^p(p_s) = \varphi^s(p_j)$,*
   (c) *$\varphi^{act} M_j \geq \varphi^{act} M_j^1 \geq ... \geq \varphi^{act} M_j^{r-1}$, where φ^{act} is the restriction of φ^{cl} to the rows associated to the symbols in Σ_O^{act},*
   (d) *any other sequence $\sigma_i' = t_i'^1, ..., t_i'^u$, describing a*

trajectory $M_j \xrightarrow{t_i'^1} M_j'^1 ... \xrightarrow{t_i'^u} M_j'^u$, *s.t.* $M_j(p_s) = M_j'^1(p_s) = ... = M_j'^u(p_s) = 0$, $t_i'^1,...,t_i'^u \in T^p$ *and* $\lambda^p(t_i^1)...\lambda^p(t_i^r) = \lambda^p(t_i'^1)...\lambda^p(t_i'^u)$, *fulfills* $\varphi^{act} M_j \geq \varphi^{act} M_j'^1 \geq ... \geq \varphi^{act} M_j'^u$.

5. *The specification cannot evolve until the plant reaches the specification symbol. Formally, for any* $p_s \in P^s$ *such that* $\varphi^s(p_s) \in \Sigma_O^p$, *let* $p_i \in P^p$ *be s.t.* $\varphi^s(p_s) = \varphi^p(p_i)$, *then* $\mathbf{Pre}^{cl}(p_i, t) \neq 0$ *for any* $t \in p_s^\bullet \cap T^s$.
6. *The PN system* $\langle \mathcal{N}^{cl}, \mathbf{M}_0^{cl} \rangle$ *is live.*

The second statement of Proposition 1 is required for the controller to be realized with a finite number of resources. The third statement stands for the realization of the controller: the only arcs allowed between the plant and the rest of nodes are bidirectional arcs between measured plant's places and transitions not in the plant, and between controllable plant's transitions and places not in the plant.

The fourth and the fifth statements of Proposition 1 stand for the fulfillment of the specification. The third statement requires that if a transition t_i in the specification is enabled, whose firing generates a symbol $\varphi^s(p_j)$ that belongs to the actuators' alphabet Σ_O^{act}, then there is a fireable sequence σ_i involving only plant's transitions (condition a)) that leads to a marking in the plant that generates the same output symbol $\varphi^s(p_j)$ (condition b)). During the execution of σ_i, output symbols from actuators can disappear, but the generation of new actuators' symbols is not allowed (excepting $\varphi^s(p_j)$, condition c)). Condition d) is a controllability condition, i.e., if another sequence σ_i' is fireable from the same marking M_j by indicating the same input symbols sequence $\lambda^p(t_i^1)...\lambda^p(t_i^r)$, then σ_i' does not produce new output actuator symbols. The fifth statement requires that the specification cannot evolve until the plant has reached the same output.

Remark 2. *The tracking controller is neither a particular case of the supervisory-control theory nor of the regulation control framework for a given specification, because in our setting the output language of the controlled system is not included in the specification's output language. It might be possible to modify the specification in order to make the output tracking controller fit in the supervisory-control or regulation frameworks, however, such adaptation would not be trivial and would not fulfilled Definition 9.*

5 CASE STUDY

Consider the *ENS* depicted in Figure 8 composed of one pneumatic arm, two stamping machines, one part dispenser, a conveyor and a start lock-button. The *ENS* required functionality is the following. The pneumatic arm retrieves a part from the dispenser and loads it in the stamping machine M_1 or M_2, depending on which is idle. Every time that a machine is loaded, it stamps and forms the part. Whenever a machine M_1 or M_2 finishes its work, the pneumatic arm unloads it placing the finished part on the conveyor. The cycle is repeated every time that the dispenser has a part. The dispenser has a proximity sensor to indicate that it holds a part (position $As1$) or it is empty (position $Bs1$); machines M_1 and M_2 have proximity sensors to indicate that the machine is idle without part (positions B_{s2} and $Bs3$ respectively) or they finished a part that must be unloaded (positions $As2$ and $As3$ respectively).

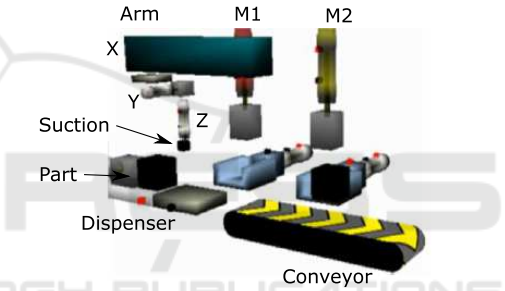

Figure 8: *ENS* for case study.

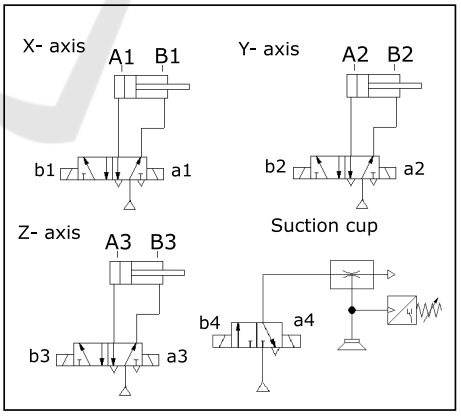

Figure 9: Actuators for the electro-pneumatic arm.

In this section, the controller for the pneumatic arm will be synthesized. Its electro-pneumatic diagram is shown in Figure 9. Every axis, X, Y, Z of the pneumatic arm has an electro-pneumatic assembly. The specification for the arm is composed of four sequences, representing when the pneumatic arm: it loads machine M_1; it loads M_2; it unloads M_1; and it

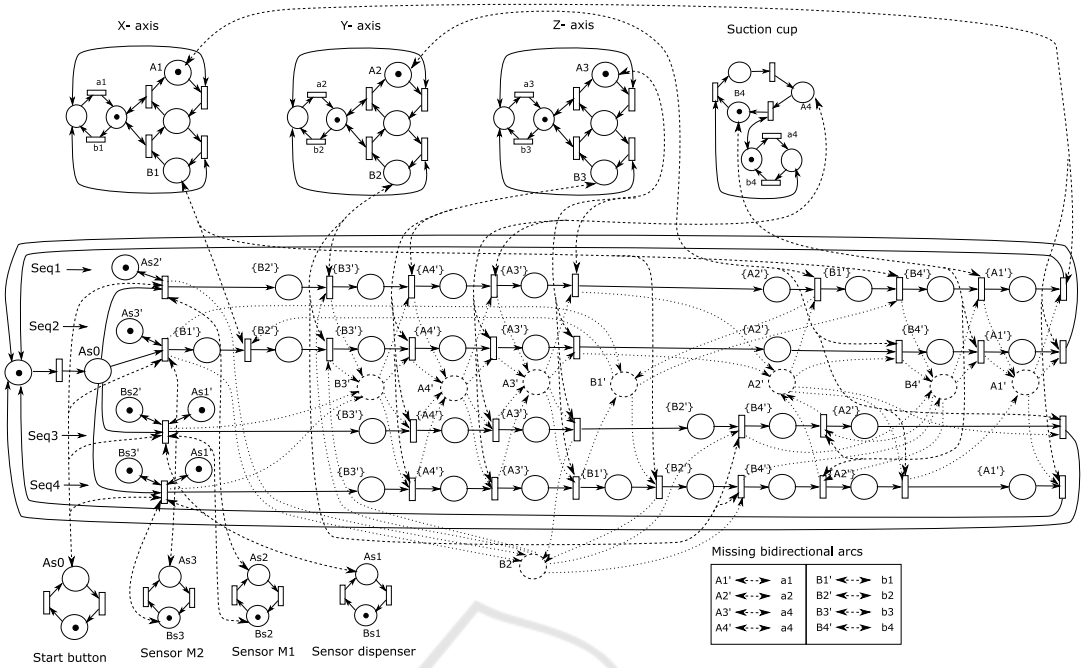

Figure 10: Closed-loop system Q^{cl} of the electro-pneumatic arm.

loads M_2. Qualitatively, every sequence is split into five subsections: the arm moves to the picking position; the effector (suction cup) turns on; the arm moves to the placing position; the effector turns off; and the arm returns to its home position. The initial state of the arm is its home position.

The specification sequences are the following:

- *unloading M_1 and placing the part on the conveyor*
  $Seq_1 = B_2B_3A_4A_3A_2B_1B_4A_1$
- *unloading M_2 and placing the part on the conveyor*
  $Seq_2 = B_1B_2B_3A_4A_3A_2B_4A_1$
- *retrieving a part form the dispenser to load M_1*
  $Seq_3 = B_3A_4A_3B_2B_4A_2$
- *retrieving a part form the dispenser to load M_2*
  $Seq_4 = B_3A_4A_3B_1B_2B_4A_2A_1$

The controller is designed according to the Algorithm 4.1 presented in subsection 4. Figure 10 presents the closed loop behavior. The specification is represented by the nodes in the central area with solid line, the four sequences are indicated. According to the proposed controller design methodology, some guards are added to transitions in sequences: the first transition of Seq_1 must be guarded by $As2$ (i.e. M_1 finished its work), thus the self-loop place $As2'$ is added to this transition; the first transition of Seq_2 must be guarded by $As3$ (i.e., M_2 finished its work), thus the self-loop place $As3'$ is added to this transition; the first transition of Seq_3 must be guarded by $Bs2$ (i.e., M_1 is idle), thus the self-loop place $Bs2'$ is added to this transition; and the first transition of Seq_4 must be guarded by $Bs3$ (i.e., M_4 is idle), thus the self-loop place $Bs3'$ is added to this transition. In addition, the last two sequences are guarded also by $As1$, requiring that a part is available in the dispenser. The starting of the four sequences is guarded by $As0$, the signal of the start lock-button.

In Figure 10 dashed circles with their input and output arcs are the places added at steps 5-12 of the Algorithm 4.1 for mirror places with the same symbols (symbols in brackets $\{\bullet\}$ represent original symbols that are removed by step 9). Bidirectional thick dashed arcs represent synchronizations between plant's output places and specification's transitions, obtained at steps 19-21 of the Algorithm. For clarity of presentation, the bidirectional arcs between specification's output places and controllable plant's transitions are not drawn (computed as steps 15-18), however, those arcs are indicated in the table in Figure 10.

The corresponding controller can be computed by eliminating from Q^{cl} the nodes of the plant that are not connected to the specification (i.e., all the plant's nodes without symbols). Finally, the resulting controller can be translated to a Ladder Diagram for its implementation in a PLC as explained in (Santoyo et al., 2001).

6 CONCLUSIONS

In this work, a tracking control problem for *ENS*'s modeled by *IPN*'s was addressed. The synthesized controller enforces the plant to track actuator's output symbol sequences indicated by the specification. The *IPN* plant model is formed as a collection of individual *ENS* models herein presented, thus its construction is a straightforward process. The specification indicates the actuator's output symbol sequence, just as the practitioners do, hence it is represented as a simple *IPN* sequence. The proposed controller is capable of handling output symbols of proximity sensors and switches as guards to trigger sequences of actuators. Moreover, the closed-loop behavior exhibits important properties, such as liveness and boundedness. The proposed approach was illustrated through an application example.

As a future work, the synthesis will be extended to specifications involving concurrency. Moreover, decentralized approaches will be investigated.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Conacyt Program for Education, project number 288470.

REFERENCES

Basile, F., Cordone, R., and Piroddi, L. (2013). Integrated design of optimal supervisors for the enforcement of static and behavioral specifications in Petri net models. *Automatica*, 49(11):3432–3439.

Campos-Rodríguez, R., Ramírez-Trevino, A., and López-Mellado, E. (2004). Regulation control of partially observed discrete event systems. In *IEEE Int. Conf. on Systems, Man and Cybernetics*.

Chao, D. (2009). Direct minimal empty siphon computation using mip. *The International Journal of Advance Manufacturing Technology*, 45(3–4):397–405.

Chen, Y. F., Li, Z. W., Khalgui, M., and Mosbahi, O. (2011). Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 8(2):374–393.

David, R. and Alla, H. (2010). *Discrete, Continuous, and Hybrid Petri nets*. Springer.

Ezpeleta, J. (1995). A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11(2):173–184.

Giua, A., DiCesare, F., and Silva, M. (1992). Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *IEEE Int. Conf. on Systems, Man and Cybernetics*.

Holloway, L., Krongh, B., and Giua, A. (1997). A survey of petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 9(2):151–190.

Iordache, M. and Antsaklis, P. (2005). A survey on the supervision of Petri nets. In *Int. Conf. on Petri nets*.

Li, Z. and Liu, D. (2007). A correct minimal siphons extraction algorithm from a maximal unmarked siphon of a Petri net. *International Journal of Production Research*, 45(9):2163–2167.

Li, Z., Wu, N., and Zhou, M. (2012). Deadlock control for automated manufacturing systems based on Petri nets. a literature review. *IEEE Transactions on Systems, Man and Cybernetics- Part C: Applications and Reviews*, 42(4):437–462.

Li, Z. and Zhou, M. (2008). On siphon computation for deadlock control in a class of Petri nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(3):667–679.

Ramadge, R. and Wonham, W. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230.

Ramírez-Prado, G., Santoyo, A., Ramírez-Trevino, A., and Begovich, O. (2000). Regulation problem in discrete event systems using interpreted petri nets. In *IEEE Int. Conf. on Systems, Man and Cybernetics*.

Ramírez-Trevino, A., Rivera-Rangel, I., and López-Mellado, E. (2003). Observability of discrete event systems modeled by interpreted Petri nets. *IEEE Transactions on Robotics and Automation*, 19(4):557–565.

S. Wang, C. W., Zhou, M., and Li, Z. (2012). A method to compute strict minimal siphons in a class of Petri nets based on loop resource subsets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(1):226–237.

Sánchez-Blanco, F., Ramírez-Trevino, A., and Santoyo, A. (2004). Regulation control in interpreted Petri nets using trace equivalence. In *IEEE Int. Conf. on Systems, Man and Cybernetics*.

Santoyo, A., Jiménez-Ochoa, I., and Ramírez-Trevino, A. (2001). A complete cycle for controller design in discrete event systems. In *IEEE Int. Conf. on Systems, Man and Cybernetics*.