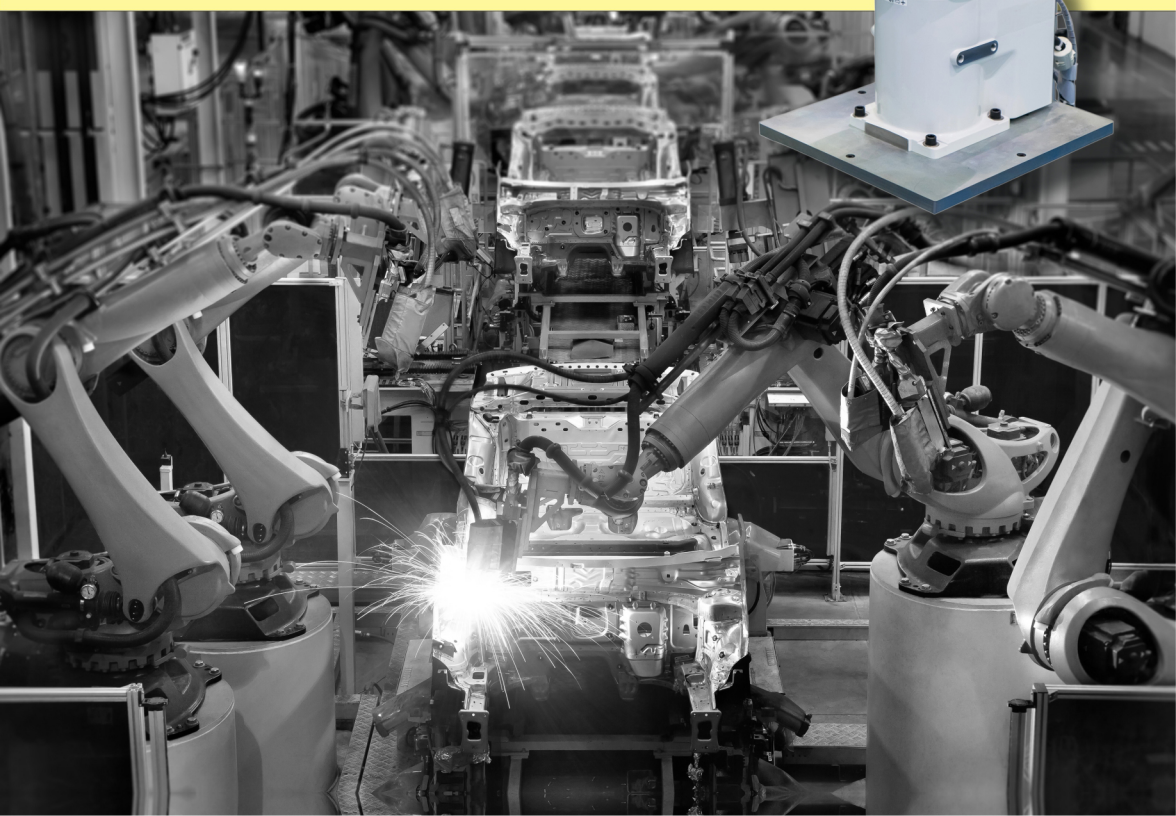
Sergey Y. Yurish, Editor

IFSA

Advances in Robotics and Automatic Control: Reviews

2



Advances in Robotics: Reviews

Book Series, Volume 2

S. Y. Yurish
Editor

Advances in Robotics: Reviews

Book Series, Volume 2

International Frequency Sensor Association Publishing

S. Y. Yurish, *Editor*
Advances in Robotics: Reviews, Book Series, Vol. 2

Published by IFSA Publishing, S. L., 2021

E-mail (for print book orders and customer service enquires):
ifsa.books@sensorsportal.com

Visit our Home Page on http://www.sensorsportal.com

ISBN: 978-84-09-25863-5
e-ISBN: 978-84-09-25862-8
BN-20210211-XX
BIC: TJFM1

Contents

Preface

After successful publication of '*Advances in Robotics and Automatic Control: Reviews*', Open Access Book Series, Vol. 1 in 2018 and feedback from our authors and readers, we have decided to publish the second volume of this Book Series in 2021.

The 2nd volume of '*Advances in Robotics and Automatic Control: Reviews*' Book Series contains 5 chapters written by 12 authors from 5 countries: Australia, Egypt, Mexico, Russia and Spain. But it is not a set of reviews. As usually, each chapter contains the extended state-of-the-art followed by new, unpublished before, obtained results.

Chapter 1 addresses the regulation problem in large-scale industrial automation systems from both, centralized and distributed perspectives. In the centralized case, the specification is given as a single Finite State Machine PN (FSM) that produces the required output sequences. In the distributed case the specification is defined as a set of local specifications and a global specification to organize the local specification evolutions.

Chapter 2 describes the modern engineering methods for linear systems with parametric noises and nonlinear control stochastic systems (CStS) are presented. Two types of differential and discrete CStS are considered: linear (for linear functional spaces) and circular (including multichannel case). Modification of normal approximation method (NAM) and statistical linearization methods (SLM) are given and illustrated by test examples. Short survey of SLM for rational, irrational, fraction-rational, integral, Bessel and Jacobi nonlinearities is included. Specific methods for nonlinear CStS (with unsolved derivatives and singular are developed and illustrated.

Chapter 3 contains a review of the state of the art, control algorithms and applications concerning solar energy plants (thermal and solar cooling) is presented. Two Model Predictive Control (MPC) algorithms developed for the solar cooling plant located at the Escuela Superior de Ingenieros at the Universidad de Sevilla are described. These strategies show that the application of advanced control techniques to solar power plants can improve the operation of this kind of plants.

Chapter 4 outlines that the decision problems under partial observation serve a rich framework for planning and control of operations in diverse

applications. In this area, approximate point-based algorithms have emerged improving the scalability of existing numerical schemes to a key level where they can be broadly used in robotics. The author elaborates on some core principles of point-based methods and present an approximate solution technique, which utilizes linear state dynamics and convexity.

Chapter 5 contains the mathematical analysis of the minimum jerk trajectory (MJT) generation. In this study, the position and the velocity of the minimum jerk trajectory as a function of time is presented in two cases: the first one is the unconstrained point-to-point movements of the human hand, whereas the second case is the curved point-to-point movements of the human hand. Simulation study is carried out with some examples and MATLAB is used for this simulation. The results prove that the minimum jerk trajectory is the smoothest possible movement of the human hand.

I hope that readers will enjoy this new volume and that can be a valuable tool for those who are involved in research and development of different robots and automation systems.

I am looking for any advices, comments, suggestions and notices from the readers to make the next volumes of becoming popular *'Advances in Robotics and Automatic Control: Reviews'* Book Series.

Sergey Y. Yurish

Book Series Editor Barcelona, Spain

Contributors

Eduardo F. Camacho
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, Camino de los Descubrimientos s/n, 41092 Sevilla, Spain

Juan M. Escaño
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, Camino de los Descubrimientos s/n, 41092 Sevilla, Spain

Antonio J. Gallego
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, Camino de los Descubrimientos s/n, 41092 Sevilla, Spain

Daniel Gevara-Lozano
CINVESTAV Unidad Guadalajara, Av. del Bosque 1145, CP 45019, Zapopan, Jalisco, Mexico, E-mail: dguevara@gdl.cinvestav.mx

Juri Hinz
University of Technology Sydney, School of Mathematical and Physical Sciences, PO Box 123 Broadway, 2007 Sydney, Australia

Antonio Ramírez-Treviño
CINVESTAV Unidad Guadalajara, Av. del Bosque 1145, CP 45019, Zapopan, Jalisco, Mexico, E-mail: jruiz@gdl.cinvestav.mx

Durvvin Rozo-Ibañez
CINVESTAV Unidad Guadalajara, Av. del Bosque 1145, CP 45019, Zapopan, Jalisco, Mexico, E-mail: darozo@gdl.cinvestav.mx

Javier Ruiz-León
CINVESTAV Unidad Guadalajara, Av. del Bosque 1145, CP 45019, Zapopan, Jalisco, Mexico

Adolfo J. Sánchez
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla, Camino de los Descubrimientos s/n, 41092 Sevilla, Spain

Abdel-Nasser Sharkawy
Mechatronics Engineering, Mechanical Engineering Department, Faculty of Engineering, South Valley University, Qena 83523, Egypt, E-mail: eng.andelnassersharkawy@gmail.com

Igor Sinitsyn
Federal Research Center 'Computer Sciences and Control', Russian Academy of Sciences, 44-2 Vavilov Str., Moscow 119333, Russia, E-mail: sinitsin@dol.ru

Carlos Renato Vázquez
Tecnologico de Monterrey, Av. Ramón Corona 2514, CP 45201, Zapopan, Jalisco, Mexico, E-mail: cr.vazquez@tec.mx

Chapter 1

Regulation Control of Discrete Event Systems for Industrial Automation

Daniel Guevara-Lozano, Durvvin Rozo-Ibañez,
Carlos Renato Vázquez, Javier Ruiz-León
and Antonio Ramírez-Treviño

1.1. Introduction

Modern industrial automation systems are becoming larger and more complex in order to be able to meet the actual requirements of consumers, such as high quality and customizable products. As long as the expectations on automation systems increase, the paradigms and tools for the analysis and design of these systems are also evolving. Now the concepts of additive manufacturing, collaborative robotics, and production based on product life-cycle are commonly introduced in these systems. Design methodologies using formal tools are also evolving to provide efficient solutions to model, analyze and control large-scale automation systems. In particular, methodologies for the analysis of Discrete Event Systems (*DES*) based on finite state automata (*FSA*) and Petri nets (*PN*) are being rapidly adapted to this new scenario.

Focusing on the methodologies for the design and synthesis of controllers for *DES*, the supervisory control theory (*SCT*) [1] has been widely used to confine the *DES* behavior into a safe one, first using *FSA* and then extending the theory to use PN [2-4]. Nowadays, this theory considers many extensions, such as control under partial observation, supremum, infimum and maximally permissive controllers, among many others. A fine updated survey of *SCT* is found in [5].

Daniel Guevara-Lozano
CINVESTAV Unidad Guadalajara, Zapopan, Jalisco, Mexico

Another interesting control methodology for *PN*, reported in the literature, is based on adding Generalized Mutual Exclusions [6-7]. This control method consists in adding monitor places to the system in order to constraint the *PN* behavior, avoiding the simultaneous execution of certain operations. This elegant idea allows to avoid undesirable states, enforce required properties, or limiting the SED behavior into the required one. This technique has been extensively studied for avoiding deadlocks in particular classes of *PN* modelling manufacturing systems [8-9]. Different authors have addressed the control problem for large-scale systems by adopting decentralized and distributed strategies, extending the *SCT* based on *FSA* (see, for instance [10]).

All the aforementioned control approaches consider safety specifications, i.e., the control objective is to confine the system to evolve according to safe trajectories. However, in industrial automation we require to enforce the system to perform predefined tasks, described as sequences of actuators' (or machines) activations. The *regulation control* framework has been introduced and studied for *DES* in [11-17] to address this control objective. In this framework, both the *specification* and the system to be controlled, named the *plant*, are modelled by interpreted Petri nets (*IPN*), which is an extension of *PN* in which controllable events (representing actuators) are labelled with input symbols and measured places (representing sensors activated at local states) are labelled with output symbols. The controller is an agent that forces the occurrence of controllable events in the plant in order to produce the output signals indicated by the specification. In detail, for every event-occurrence in the specification, the controller forces the occurrence of a sequence of controllable events in the plant so the current plant and specification states generate the same output signals, i.e., the plant output is tracking the specification output.

This chapter addresses the regulation problem in large-scale industrial automation systems from both, centralized and distributed perspectives. In the centralized case, the specification is given as a single Finite State Machine *PN* (*FSM*) that produces the required output sequences. In the distributed case the specification is defined as a set of local specifications and a global specification to organize the local specification evolutions. The centralized approach is useful in small and medium size systems, nevertheless, in large size systems the distributed approach must be chosen. Depending on the system structure, three different implementations are possible. In the first one the local controllers work without knowledge of other controllers. The system structure is enough

to guarantee the adequate closed-loop evolution. In the second one, messages are passed among local controllers in order to enforce closed-loop properties. The third implementation includes a high-level controller, named coordinator that controls the local controllers to guarantee the required global behaviour.

This chapter is organized as follows: basic concepts on Interpreted Petri nets are recalled in Section 1.2. The regulation control framework is described in Section 1.3. Section 1.4 presents modelling methodologies for electro-pneumatic systems and continuous processes. Section 1.5 presents a methodology for defining specifications. Section 1.6 presents an algorithm for the controller synthesis. Decentralized and distributed control strategies are presented in Section 1.7. Finally, conclusions and future work are presented in Section 1.8.

1.2. Interpreted Petri Nets

In the regulation control framework, an extension of Petri nets ([18]) called *Interpreted Petri nets* ([19]) is used, this extension allows to represent input and output symbols in transitions and places, respectively.

1.2.1. Petri Nets

Definition 1. A Petri net (*PN*) structure is a bipartite digraph represented by the 4-tuple $G = \langle P,T,I,O \rangle$, where $P = \{p_1,p_2,...,p_n\}$ is the finite set of places, $T = \{t_1,t_2,...,t_o\}$ is the finite set of transitions, $I:P\times T \rightarrow Z_{\geq 0}$ is a function representing the weighted arcs connecting places to transitions, $O:P\times T \rightarrow Z_{\geq 0}$ is a function representing the weighted arcs connecting transitions to places. $Z_{\geq 0}$ represents the non-negative integers.

Pictorially, places are represented by circles, transitions by rectangles, and arcs by arrows. Functions I and O are characterized by $|P|\times|T|$ matrices ***Pre*** and ***Post***, respectively, in such a way that $\boldsymbol{Pre}[i,j] = I(p_i,t_j)$ and $\boldsymbol{Post}[i,j] = O(p_i,t_j)$. The *incidence matrix* of a *PN* is a $|P|\times|T|$ matrix $\boldsymbol{C}$ defined such that $\boldsymbol{C}[i,j] = \boldsymbol{Post}[i,j]-\boldsymbol{Pre}[i,j]$. The symbol $\bullet t_j$ (resp. $\bullet p_i$ denotes the set of all places p_i (resp. transitions t_j) such that $I(p_i,t_j) \neq 0$ (resp. $O(p_i,t_j) \neq 0$). Similarly, $t_j\bullet$ (resp. $p_i\bullet$) denotes the set of all places p_i (resp. transitions t_j) such that $O(p_i,t_j) \neq 0$ (resp. $I(p_i,t_j) \neq 0$).

A *PN* is said to be a *state machine* if $|\bullet t_j| = |t_j\bullet| = 1, \forall t_j \in T$.

Definition 2. Given a *PN* structure, the *marking* distribution is defined as a function $M:P \rightarrow Z_{\geq 0}$, where $M(p_i)$ represents the number of tokens residing inside the place p_i (tokens are depicted as dots). The marking distribution is expressed as a column vector $\boldsymbol{M}$ of len3gth $|P|$, where $\boldsymbol{M}[i] = M(p_i), \forall p_i \in P$. A *PN system* is the duple $\langle G, \boldsymbol{M}_0 \rangle$, where G is a *PN* structure and $\boldsymbol{M}_0$ the initial marking distribution. The marking distribution evolves according to the firing of transitions. A transition t_j is *enabled* at marking $\boldsymbol{M}_k$ if $\forall p_i \in \bullet t_j$, $\boldsymbol{M}_k(p_i) \geq I(p_i, t_j)$, this is denoted as $\boldsymbol{M}_k[t_j\rangle$. A transition t_j can fire if it is enabled. The firing of an enabled transition t_j leads to a new marking $\boldsymbol{M}_{k+1}$ that can be computed by

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k + \boldsymbol{C}\boldsymbol{e}_j,$$

where $\boldsymbol{e}_j[i] = 0$ for $i \neq j$ and $\boldsymbol{e}_j[j] = 1$. This is denoted as $\boldsymbol{M}_k[t_j\rangle \boldsymbol{M}_{k+1}$.

Given a *PN* system $\langle G, \boldsymbol{M}_0 \rangle$, a sequence of transitions $\sigma = t_1 t_2 \ldots t_k$ is said to be a firing sequence (or fireable) from $\boldsymbol{M}_0$ if there exist markings $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_{k-1}$ such that $\boldsymbol{M}_0[t_1\rangle \boldsymbol{M}_1, \boldsymbol{M}_1[t_2\rangle \boldsymbol{M}_2, \ldots, \boldsymbol{M}_{k-1}[t_k\rangle$.

The marking $\boldsymbol{M}'$ reached after the firing of a fireable σ from a marking $\boldsymbol{M}$ can be computed by the so-called *PN fundamental equation*

$$\boldsymbol{M}' = \boldsymbol{M} + \boldsymbol{C}\vec{\sigma},$$

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 $\boldsymbol{M}[\sigma\rangle \boldsymbol{M}'$, moreover, $\boldsymbol{M}'$ is said to be *reachable* from $\boldsymbol{M}$. A *PN* system is said to be *safe* if, for every reachable marking, there is at most one token in each place. The reachability set of a *PN* is the set of all the reachable markings from $\boldsymbol{M}_0$, and it is denoted as $R(G,\boldsymbol{M}_0)$. A *PN* is said to be *deadlock-free* if $\forall \boldsymbol{M} \in R(G,\boldsymbol{M}_0)\ \exists t \in T$ such that $\mathbf{M}[\mathrm{t}\rangle$.

1.2.2. Interpreted Petri Nets

Definition 3. An Interpreted Petri net (*IPN*) system is a 6-tuple $Q = \langle G, \boldsymbol{M}_0, \Sigma_I, \Sigma_O, \lambda, \varphi \rangle$, where $\langle G, \boldsymbol{M}_0 \rangle$ is a *PN* system, and:

- Σ_I is the input alphabet, where each element of the set Σ_I is an input symbol;

- Σ_O is the output alphabet, where each element of the set Σ_O is an output symbol;
- $\lambda: T \rightarrow 2^{\Sigma_I}$ is the labeling function of transitions (a single transition can be associated with more than one symbol from the input alphabet Σ_I);
- $\varphi: P \rightarrow 2^{\Sigma_O}$ is the labeling function of places (a single place can be associated with more than one output symbol).

If $\lambda(t) \neq \emptyset$, the transition t is said to be *controllable,* otherwise it is *uncontrollable.* A place $p \in P$ is said to be measurable if $\varphi(p) \neq \emptyset$, otherwise, it is non-measurable.

The evolution of an *IPN* is similar to that of the *PN* system with the addition of the following rules:

1) A symbol $a \in \Sigma_I$ is said to be *indicated* if either it is activated by an external device (for instance a controller or a user) or $\exists p \in P$ such that $a \in \varphi(p)$ and $M(p) > 0$ (notice that Σ_I may include symbols of Σ_O);
2) If $\lambda(t_j) = a$ is indicated and t_j is enabled then t_j *must* fire;
3) If $\lambda(t_j) = a$ and t_j is enabled, but the symbol $\lambda(t_j)$ is not indicated, then t_j cannot fire;
4) If $\lambda(t_j) = \emptyset$ and t_j is enabled, then t_j *can* fire at any moment;
5) At any reachable marking $\boldsymbol{M}_k$, an external observer reads the symbols $\varphi(p_i)$ if $M(p_i) > 0$.

The reachability set of an *IPN Q* will be denoted as $R(Q, \boldsymbol{M}_0)$, in order to make explicit that in these models the reachability depends not only on the *PN* structure and the initial marking, but also on the input and output labelling functions.

An *IPN* is said to be *event-detectable* if $\forall t_i, t_j \in T$ it holds either they are associated to different input symbols ($\lambda(t_i) \neq \lambda(t_j)$) or their firings produce different marking changes ($\mathbf{Ce}_i \neq \mathbf{Ce}_j$).

Example 1. Fig. 1.1 shows an *IPN Q* where $P = \{p1, p2, p3, p4\}$, $T = \{t1, t2, t3\}$, $\Sigma_I = \{a,b\}$, $\Sigma_O = \{A,B,C,b\}$. The incidence matrix of *Q* is given by

$$\boldsymbol{C} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{bmatrix}$$

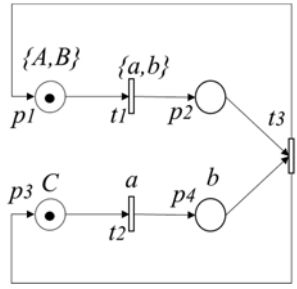


Fig. 1.1. *IPN* system described in Example 1.

The initial marking is $\boldsymbol{M}_0 = [1, 0, 1, 0]^T$, describing that p_1 and p_3 have a token but p_2 and p_4 have not. Thus, at the initial marking, the *IPN* indicates the output signals $\{A,B,C\}$. Now, at the initial marking, both t_1 and t_2 are enabled by the marking, however, none of them can fire since both have input symbols that must be indicated (they are controllable). In particular, t_1 requires that both symbols a and b be indicated. If an external agent indicates the symbol a, only t_2 is fired. The firing of t_2 transfers a token from p_3 to p_4. The new marking can be computed as $\boldsymbol{M}_1 = \boldsymbol{M}_0 + \boldsymbol{C}\boldsymbol{e}_2 = [1,0,0,1]^T$, where $\boldsymbol{e}_2 = [0,1,0]^T$ (i.e., the elementary vector associated to t_2), thus p_1 and p_4 are the marked places. At this new marking, the symbols $\{A,B,b\}$ are indicated. Then, t_1 is fired if an external agent indicates the symbol a again (since b is indicated by the current marking). The firing of t_1 transfers a token from p_1 to p_2. At this new marking, t_3 is enabled since its input places have tokens and it is uncontrollable (it is not labelled), thus it can fire at any time, leading the system to the initial marking. In this way, the sequence $\sigma = t_2t_1t_3$ is a sequence fireable from the initial marking. As an example, $\sigma' = t_1t_2t_1$ is not fireable from the initial marking, since t_1 cannot fire twice without firing t_3.

The function φ can be represented by a $|\Sigma_O| \times |P|$ matrix $\boldsymbol{\varphi}$. Enumerating the output symbols as $\Sigma_O = \{a_1, a_2, \ldots, a_{|\Sigma_O|}\}$, the *output matrix* $\boldsymbol{\varphi}$ is defined such that $\boldsymbol{\varphi}[i,j] = 1$ if $a \in \varphi(p_j)$and $\boldsymbol{\varphi}[i,j] = 0$ if $a \notin \varphi(p_j)$. In this way, each row of $\boldsymbol{\varphi}$ represents a symbol from Σ_O. The output vector

of the *IPN* at the marking $\boldsymbol{M}$ is defined as $\boldsymbol{y} = \boldsymbol{\varphi M}$, where $\boldsymbol{y}[i] > 0$ if a_i is indicated by $\boldsymbol{M}$.

Example 2. Consider again the *IPN* depicted in Fig. 1.1, if we order the output symbols as $\Sigma_O = \{A,B,C,b\}$, then the output matrix is defined as

$$\boldsymbol{\varphi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, the first row of $\boldsymbol{\varphi}$ means that the symbol A is indicated by the marking of p_1, the second row means that B is also indicated by the marking of p_1, the third and fourth rows mean that C is indicated by the marking of p_3 and symbol b is indicated by the marking of p_4, respectively. If the sequence $\sigma = t_2t_1t_3$ is fired from the initial marking, the sequence of output symbols is $\{A,B,C\}$, $\{A,B,b\}$, b, $\{A,B,C\}$ or $[1,1,1,0]^T$, $[1,1,0,1]^T$, $[0,0,0,1]^T$, $[1,1,1,0]^T$ as output vectors.

1.3. Regulation Control Framework

The regulation problem for *IPN* was introduced in [11-13], and has been recently addressed in [14-15] as a tool for designing controllers for industrial automation. In the tracking-control paradigm, the system to be controlled, named the *Plant*, and the desired behavior, named the *Specification*, are both modelled as *IPN*. The goal is to compute a controller that drives the Plant in such a way that its output becomes equal to the Specification's output, i.e., the Plant's output tracks the Specification's output. In the forthcoming sections, methodologies for the Plant modelling and Specification definition will be recalled from [14-17]. Before introducing such methodologies, let us describe some concepts of the regulation problem.

Definition 4. The *Plant* is a safe event-detectable *IPN* $Q = (G, \boldsymbol{M}_0, \Sigma_I, \Sigma_O, \lambda_I, \varphi)$ that models the system to be controlled, where $G = (P,T,I,O)$. It is assumed that two plant places cannot generate the same output symbol (i.e. a sensor cannot be turned on by two different variables), moreover, the input and output plant alphabets are disjoint (i.e., $\Sigma_I \cap \Sigma_O = \emptyset$).

Definition 5. Given a plant as defined in Definition 4, a *Specification* is a deadlock-free state machine $IPN\ Q^S = (G^S, \boldsymbol{M}_0{}^S, \Sigma_I{}^S, \Sigma_O{}^S, \lambda^S, \varphi^S)$. The output alphabet fulfils $\Sigma_O{}^S \cap \Sigma_I = \emptyset$. The specification input alphabet contains symbols from the plant's alphabets and other external symbols, whose set is denoted as $\Sigma_{ext} = \Sigma_I{}^S \setminus \{\Sigma_I \cup \Sigma_O\}$.

The regulation problem can be formulated as the synthesis of a controller function H that provides the input symbols that have to be indicated to the plant in order to produce an output equal to that of the specification.

Definition 6. Let $Q = (G, \boldsymbol{M}_0, \Sigma_I, \Sigma_O, \lambda_I, \varphi)$ and $Q^S = (G^S, \boldsymbol{M}_0{}^S, \Sigma_I{}^S, \Sigma_O{}^S, \lambda^S, \varphi^S)$ be the *IPN* modeling the plant and specification, respectively. The regulation problem consists in the computation of a *controller* function

$$H: R(Q, \boldsymbol{M}_0) \times R(Q^S, \boldsymbol{M}_0{}^S) \times T^S \rightarrow \Sigma_I,$$

such that $\forall \boldsymbol{M}_i \in R(Q, \boldsymbol{M}_0)$, $\forall \boldsymbol{M}_j{}^S \in R(Q^S, \boldsymbol{M}_0{}^S)$, the indication of the input symbols $H(\boldsymbol{M}_i, \boldsymbol{M}_j{}^S, t_k{}^S)$, where $t_k{}^S$ is the specification transition previously fired, will eventually lead the plant to a marking $\boldsymbol{M}_j$ such that $\varphi(\boldsymbol{M}_j) = \varphi^S(\boldsymbol{M}_j{}^S)$.

For practical reasons, the definition of a controller as the function H is inconvenient, since it must be defined for every combination of the plant and the specification reachable markings. Instead, a regulation controller can be conveniently realized as an *IPN* in such a way that, when synchronized with the plant, will produce the symbols that the plant transitions require to eventually reach a marking that produces the specified output (i.e., the symbols that H would provide).

Example 3. Consider the *IPN* model of Fig. 1.2 representing the plant (composed of the actuator and the sensor) and the specification. The plant is an assembly of a double-effect pneumatic actuator with its two-position solenoid valve and a proximity sensor. The pneumatic actuator can be retracted (when signal *XA* is generated, i.e. there is a token in p_0) or extended (when signal *XB* is generated, i.e. there is a token in p_2). Indicating the input signal *Xa* (representing the activation of a valve's solenoid) fires t_4 when it is enabled, adding one token in p_4 and allowing the firing of t_2 and t_0 when they are enabled, retracting the actuator; similarly, signal *Xb* (representing the activation of the other valve's solenoid) leads to an extension of the actuator. The sensor is turned on (p_6 is marked) if a work-piece is detected, and turned off (p_5 is marked) if it is not the case. According to the specification, it is required

that when the proximity sensor detects a work-piece (the signal *On* is indicated by p_6), transition t_1^S is fired starting two cycles of extending-retracting the actuator.

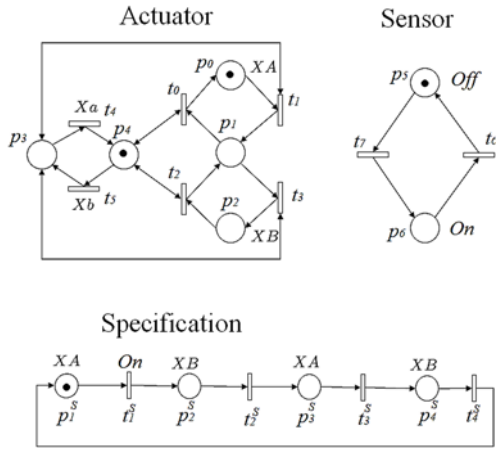


Fig. 1.2. *IPN* models of the plant and the specification of Example 1.

Fig. 1.3 shows an *IPN* that represents the closed-loop behavior of the plant with a controller that imposes the required specification. Notice that, in the closed-loop behavior, the controller indicates symbols of controllable transitions in the plant (*Xa, Xb*); and the plant indicates symbols that allow the firing of enabled transitions in the controller (*XA, XB, On*). In the current example, the places p_0 and p_6 of the plant are marked indicating signals *{On, XA}*. Hence transition t_8 in the controller is fired, moving the token from p_7 to p_8, and the symbol *Xb* is indicated by the controller. This symbol results in the firing of transition t_5 in the plant leading to the extension of the pneumatic-actuator (the token at p_0 will move to p_2), marking place p_2 in the plant and indicating signal *XB*, allowing the controller to evolve. The closed-loop evolution continues generating the two retracting-extending cycles in the actuator.

1.4. Plant Construction

By adopting a Bottom-Up modelling methodology, a plant *IPN* model for an automation system can be built by using the synchronous and permissive products of small *IPN* models defined for each of the components [20]. This strategy is adapted to electro-pneumatic systems

in [14] and to industrial processes represented by Piping and Instrumentation Diagrams (*P&ID*) in [16]. In [20] a library of the different actuators and sensors is created, and the synchronous and permissive products are implemented by the firing rules and an adequate labeling of places and transitions. In [16] the process diagrams are translated into tables capturing the system elements and relations among them. Tables are used to remove errors and unnecessary information. Latter, these tables are translated to *IPN*. Next sections describe these modelling methodologies.

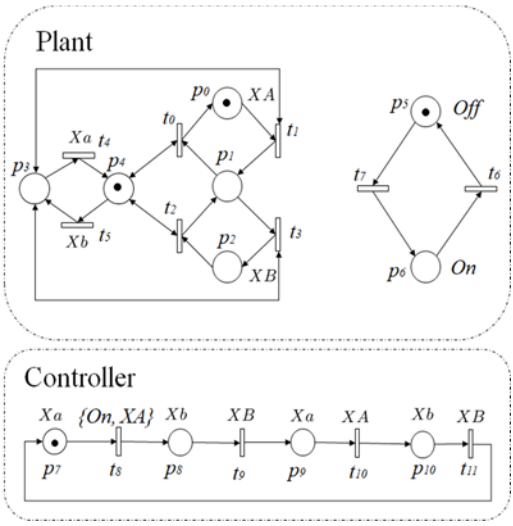


Fig. 1.3. *IPN* describing the closed-loop of Example 1.

1.4.1. Electro-pneumatic Systems

Electro-pneumatic systems are common components in industrial automation. For this reason, a library of *IPN* models was proposed in [14] for the most frequently used components in electro-pneumatic systems, including push buttons, selectors proximity sensors, electro-pneumatic valves and actuators. These component models are presented in Figs. 1.4-1.7.

Fig. 1.4 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, for instance, 1.4 (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*. Reed magnetic sensors (not shown) are used to detect limit positions of pneumatic actuators, these behave as NO switches, thus, the model of Fig. 1.4 (a) can also be used for these sensors.

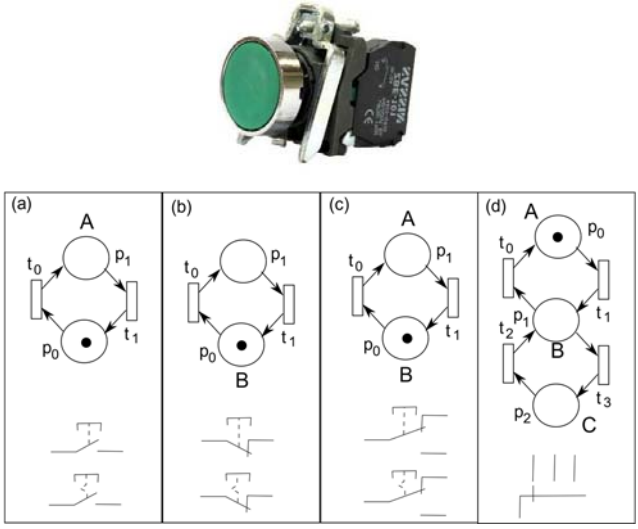


Fig. 1.4. *IPN* models for different kinds of switches.

Fig. 1.5 shows the electro-pneumatic symbols for different kind of proximity sensors, commonly used to detect the presence of a part or a limit position of an actuator, and its *IPN* model. These sensors provide two output signals, one NO (signal *A*) and one NC (signal *B*) to detect the presence and absence of parts, respectively.

Pneumatic actuators and valves are frequently used in usual assemblies. Fig. 1.6 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). Place p_0 represents the state in which vacuum is not activated, thus the sensor provides a signal *B*; place p_1 represents 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.

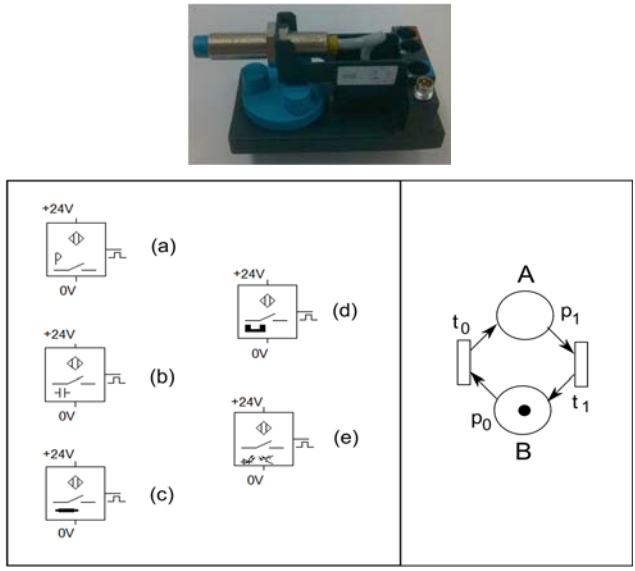


Fig. 1.5. Sensor symbols and their *IPN* model for a pressure sensor (a); capacitive sensor (b); inductive sensor (c); magnetic sensor (d), and optical sensor (e).

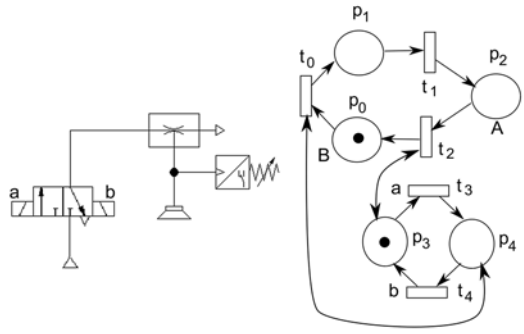


Fig. 1.6. *IPN* model for a vacuum assembly.

Fig. 1.7 represents the most typical valve-actuator assemblies. Pneumatic-grippers (not shown) are usually driven by double acting actuators controlled by 5-ways 2-positions valves, thus they are similar to Fig. 1.7(a). The same *IPN* model is valid for all the assemblies. In these, 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 Fig. 1.7 (b), the symbol *a* is defined as the logical negation of *b* (i.e., the absence of event *b*).

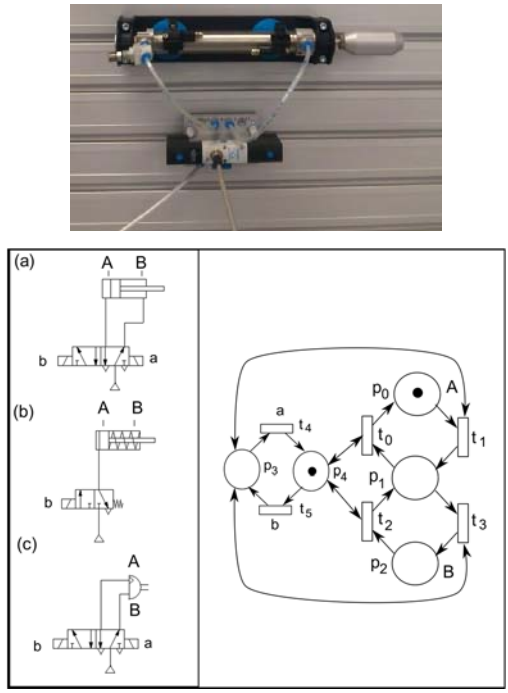


Fig. 1.7. *IPN* models for electro-pneumatic assemblies: double acting actuator controlled by a 5-ways 2-positions valve (a); spring return actuator controlled by a 3-ways 2-positions spring return valve (b), and a rotary actuator controlled by a 5-ways 2-positions valve (c).

In this way, electro-pneumatic systems can be modelled by taking the *IPN* models of Figs. 1.4-1.7.

Example 4. Fig. 1.8 shows a manufacturing cell at the Automation Lab of the Tecnologico de Monterrey that is used for testing controllers. Fig. 1.9 shows a diagram and the *IPN* model of the manufacturing cell.

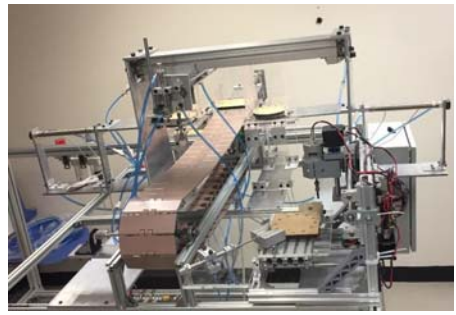


Fig. 1.8. Manufacturing cell for supervisor synthesis tests at the Automation Lab.

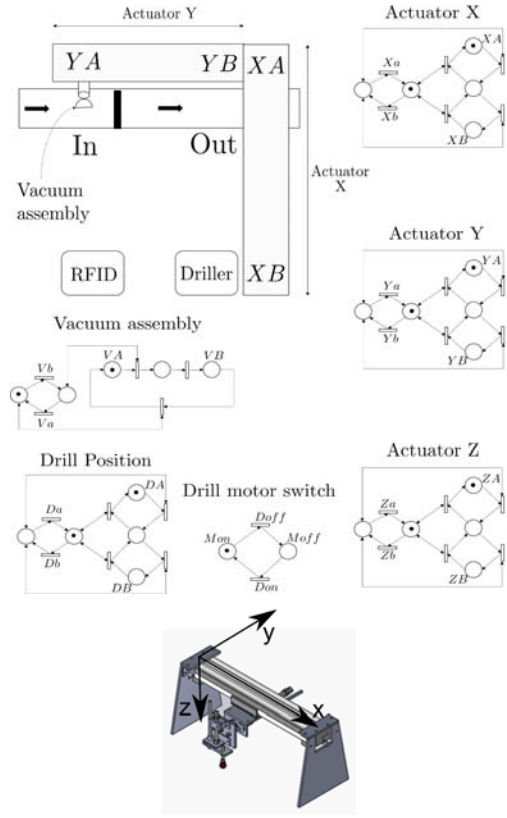


Fig. 1.9. *IPN* model of a manufacturing cell consisting of a pneumatic arm with 3 degrees of freedom, a CNC-Driller station, a RFID-tag colocation station and a conveyor. The 3D model of the pneumatic arm with its axis is shown at the right.

The cell consists of a conveyor, divided in two sections (*In* and *Out*) by a physical barrier, a pneumatic arm with three linear pneumatic actuators (*X*, *Y* and *Z*) and a vacuum assembly (*V*) with a suction cup, and two stations, one that places a RFID tag to work-pieces and a CNC driller. The arm can reach 8 different positions that are the combinations of the linear actuators' extreme positions. The suction cup has two states, turned off (*VA*) and on (*VB*). To take a work-piece from the conveyor's *In* position, the arm's actuators must reach the positions *XA, YA, ZB* and then the cup is turned on. Similarly, to leave a work-piece in the conveyor's *Out* position, the arm's actuators must reach *XA, YB, ZB* and then the suction cup is turned off. To take or leave a part at the RFID and the Driller stations, the arm's actuators must reach *XB, YA, ZB* and *XB, YB, ZB*, respectively. The actuator *Z* must be at *ZA* position (the suction cup is elevated) when the arm moves between these positions. The *Driller* station consists of an actuator (*D*) with two positions, retracted (*DA*) and extended (*DB*), that secures the work-piece to be drilled, and a switch (*M*) that turns off (M_{off}) and on (M_{on}) the drill.

There are sensors (not modelled) providing the signals *SIn*, *tagIn*, *SRFID*, *tagRFID* and *SDrill* that indicate if there is a work-piece at the *In* position, if the work-piece at the *In* position is tagged, if there is a work-piece in the RFID station, if the work-piece in station RFID is tagged and if there is a work-piece in the Driller station, respectively. These sensors also provide the corresponding negation signals, denoted with either an over-line or a symbol ¬.

1.4.2. Industrial Processes with Discrete Actuators

Previous subsection shows how to model pneumatic systems, now this section focuses on modeling systems described by *Piping and Instrumentation Diagrams* (*P&ID*), *Process narratives* (*ProN*), and *Operation narratives* (*OpN*). Through this section, we recall the modelling methodology introduced in [16] for these systems.

A *P&ID* is a graphic representation illustrating the interconnection of the equipment, control *elements* (sensors, actuators, converters, timers, etc.) used in the control of a process [21], as well as the variables that must be controlled. This diagram uses a set of symbols and nomenclatures for elements and variables identification based on the standards issued by the Instrumentation Systems and Automation Society (*ISA*).

The variables appearing in a *P&ID* are named *controlled variables*, the value of each variable is named the *variable state*, and the set of possible values of a variable is named the *variable range*. In a similar way, the *actuator state* is the condition of the actuator and the *actuator range* is the set of possible states of the actuator. Both the controlled variables and the actuators are referred as the process *elements*. The elements controlling, acting or sensing a controlled variable form a *control loop* in the *P&ID*; control loops are gathered to form *sub-processes*. In this work, we consider processes in which the actuators range and the variables range are discrete.

The *ProN* is a description of the process functionality, provided in natural language. This narrative describes the possible evolution of each controlled variable as function of both the variables states and the actuators states. Variables and actuators interact in two different ways. First, the simultaneous occurrence of variables and actuators state changes is represented by a synchronous relation [22-23]. Second, the permissive relation represents the enabling of a variable state change when some actuator is at a particular state [22]. These relations are implied in the process narrative.

Example 5. Consider the system represented in the *P&ID* of Fig. 1.10. The *ProN* and *OpN* are described as following.

Process narrative. The system has two sub-processes. The first one is the bottle transfer (*C-1*) that is composed of three position transmitters (*ZT-1*, *ZT-2*, and *ZT-3*), a motor (*ZZ-103*), and a robot arm (*ZN-103*) with its gripper (*ZZC-103*). Transmitters are used to detect the relevant bottle sites arrival (position-1), filling (position-2), and unloading (position-3), defining controlled variables BP_1*-103*, BP_2*-103* and BP_3*-103* respectively, where each can be either at a *presence* state (BP_1^p, BP_2^p, and BP_3^p) or at an *absence* state (BP_1^a, BP_2^a, and BP_3^a). If the motor *ZZ-103* is turned *on* the conveyor belt conveys bottles, then bottles are transferred from the position-1 to the position-2, and from the position-2 to the position-3 (a physical barrier is at the end of the conveyor); when the motor *ZZ-103* is turned *off* the conveyor stops. The variable of the gripper *ZZC-103* can be *opened* or *closed*. The robot arm is programmed to reach positions *Home* (R_H), RP_1 (over *ZT-3*) and RP_2 (over the exit). The second sub-process (*T-1*) is the bottle filling, it consists of a level transmitter (*LT-104*) and a level valve (*LV-104*). The bottle-level variable (*LT-104*) can be *empty* (BL^e) or *full* (BL^f). If the valve *LV-104* is

opened, then the state of *BL-104* changes from BL^e to BL^f. The complete process is controlled by the controller *YIC-1*.

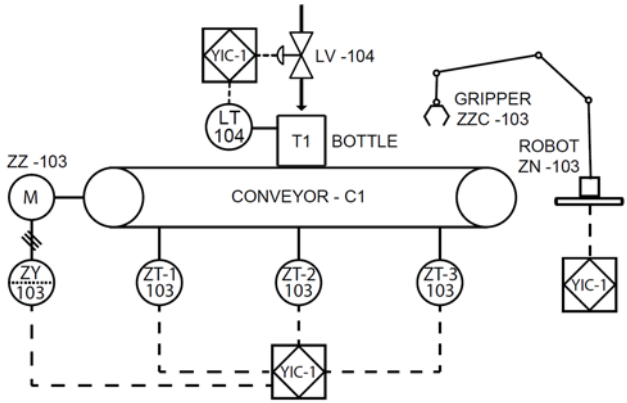


Fig. 1.10. P&ID of the bottle filling system of Example 4.

In the modelling methodology proposed in [16], the *P&ID* and *ProN* information is summarized in four tables, named the *elements description* table, the *elements-behavior* table, the *permissive relation* table *and the synchronous relation* table.

The *elements description* table lists the controlled variables and actuators with their characteristics: range, initial condition, labels, etc. The first column of this table is filled with the element's name; the second column with the variable/actuator range; and the third column with the variable/actuator initial state. The different values in the Range column are referred as the states of the element. The last two columns correspond to the element location in the process. The *element description* table for Example 5 is depicted in Table 1.1.

Every *element behavior* table captures the behavior of an element mentioned in the *P&ID*. In each table, every row and column represent a state of the element from the Range column of the Element Description table. Given two different states, state-1 and state-2, a check-mark is drawn in the cell at row state-1 and column state-2 if the element evolves from state-1 to state-2 without visiting another state.

The *elements' behavior* tables of the industrial process described for the Example 5 are shown in Table 1.2.

Table 1.1. *Elements description* table for Example 5.

Controlled variables & Actuators	**Range**	**Initial state**
ZZ-103 (Motor)	*off, on*	off
ZN-103 (Robot)	*home* (R_H), RP_1, RP_2	RH
ZZC-103 (Gripper)	opened (G_o), closed (G_c)	Go
LV-104 (Valve)	opened, closed	closed
BP_1*-103* (Bottle-Position 1)	absence (BP_1^a), presence (BP_1^p)	absence
BP_2*-103* (Bottle-Position 2)	absence (BP_2^a), presence (BP_2^p)	absence
BP_3*-103* (Bottle-Position 3)	absence (BP_3^a), presence (BP_3^p)	absence
BL-104 (Bottle-level)	empty (BL^e), full (BL^f)	empty

Table 1.2. *Elements' behavior* tables for Example 5.

ZZ-103	***off***	***on***
Off		X
On	X	

BP_1-103	***BP_1^a***	***BP_1^p***
BP_1^a		X
BP_1^p	X	

BP_3-103	***BP_3^a***	***BP_3^p***
BP_3^a		X
BP_3^p	X	

BP_2-103	***BP_2^a***	***BP_2^p***
BP_2^a		X
BP_2^p	X	

ZZC-103	***opened***	***Closed***
opened		X
closed	X	

BL-104	***BL^e***	***BL^f***
BL^e		X
BL^f	X	

ZN-103	***RP_1***	***Home***	***RP_2***
RP_1		X	
Home	X		X
RP_2		X	

LV-104	***opened***	***closed***
Opened		X
Closed	X	

The interaction between controlled variables and actuators is captured in the *permissive* and *synchronous relation* tables. The *permissive relation* table (Table 1.3) represents the fact that certain variable changes occur only if certain actuators are at some particular states. In the *permissive relation* table, a row is defined for each actuator from the *P&ID*. Furthermore, each actuator row is split into sub-rows, one per each state in the actuator's range. On the other hand, a column is defined for each

controlled variable. If the evolution of a controlled variable V_i from state S_1 to state S_2 requires that actuator A_j be in state R, then write "S_1 to S_2" in the cell associated to the actuator A_j, at sub-row representing state R and column representing V_j.

Table 1.3. *Permissive relation* table for Example 5.

		Controlled variables			
		BP1-103	***BP2-103***	***BP3-103***	***BL-104***
Actuator	**Actuator state**	**Events**			
ZZ-103	*off*				
	on	BP_1^p to BP_1^a	BP_2^a to BP_2^p BP_2^p to BP_2^a	BP_2^a to BP_2^p	
ZN-103	RP_1			BP_3^p to BP_3^a	
	home				
	RP_2				
ZCC-103	*closed*			BP_3^p to BP_3^a	
	opened				
LV-104	*closed*				
	opened				BL^e to BL^f

The *synchronous relation* table indicates when two or more controlled variables evolve simultaneously. In the synchronous relation table, rows and columns are associated to controlled variables. Furthermore, rows and columns are split into sub-rows and sub-columns, respectively, representing the events of each controlled variable (i.e. the change from one state to another). The cell defined by a sub-row $event_i$ and a sub-column $event_j$ is marked with an "*X*" if these events occur simultaneously. The synchronous relation table for the bottle filling system is shown in Table 1.4.

The methodology continues by representing the relation between actuators and variables of the process in the Process Graph(P_G) defined below.

Definition 7. A process graph is the triplet $P_G = \{N,B,D\}$, where $N = \{n_1,...,n_q\}$ is a finite set of vertices, and n_i represents the *i-th* element of the *P&ID*; $B \cup D \subset N \times N$ is a set of arcs, where B represents the permissive relation, and D represents the synchronous relation. If arc $(n_i,n_j) \in B$, then n_i is an actuator, n_j is a variable, and there exists a no null cell at row n_i and column n_j in the permissive relation table. If arc $(n_i,n_j) \in D$, then n_i,n_j are both variables and there is a marked cell at row

n_i and column n_j in the *variable relation* table. Arcs defined from B are represented with solid lines and arcs defined from D are represented with dashed lines.

Fig. 1.11 represents the process graph of the Example 5.

Table 1.4. *Synchronous relation* table for Example 5.

Variables and Events		***BP1-103***		***BP2-103***		***BP3-103***		***BL-104***	
		BP_1^a to BP_1^p	BP_1^p to BP_1^a	BP_2^a to BP_2^p	BP_2^p to BP_2^a	BP_3^a to BP_3^p	BP_3^p to BP_3^a	BL^e to BL^f	BL^f to BL^e
BP_1-103	BP_1^a to BP_1^p								
	BP_1^p to BP_1^a			X					
BP_2-103	BP_2^a to BP_2^p		X						
	BP_2^p to BP_2^a					X			X
BP_3-103	BP_3^a to BP_3^p				X				X
	BP_3^p to BP_3^a								
BL-104	BL^e to BL^f								
	BL^f to BL^e				X	X			

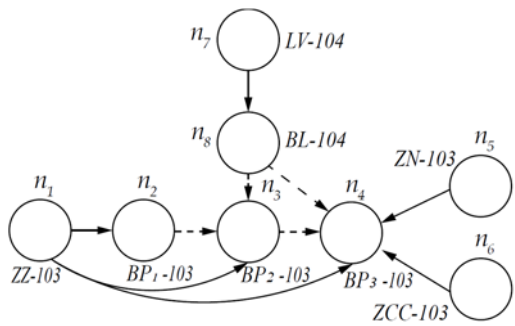


Fig. 1.11. Process graph of the bottle filling process.

The next step in the methodology translates the P_G into an *IPN*, every vertex of P_G is translated into an *IPN* module and the resulting *IPN* modules are connected using the synchronous and permissive relations.

IPN module. The *i-th* module $Mod_i = (G^i, M_0^i, \Sigma_I^i, \Sigma_O^i, \lambda^i, \varphi^i)$, where $G_i = (P_i,T_i,I_i,O_i)$, is built for the n_i vertex of P_G, based on the element behavior table associated to the *i-th* element. In detail, $P_i = \{p_1,...,p_r\}$, where each place is defined for each row of the *behavior* table. Moreover, if there is a check mark in the cell defined by the row r_j and column c_k of the *behavior* table, then a transition t_{jk} is defined and arcs from p_j to t_{jk}, and from t_{jk} to p_k are defined as well. In this way, there are as many transitions as check marks in the *behavior* table. The initial marking of Mod_i is given by $M_0^i(p_k) = 1$ if p_k is the initial condition of e_i and $M_0^i(p_k) = 0$ otherwise.

Places are labelled according to the element description table, i.e., $\varphi^i(p_j) = state_k$ if the place p_j of the *i-th* element is associated to the state $state_k$. Moreover, distinct labels are associated to the transitions of the actuator elements (all the transitions in the actuators are controllable). The input and output alphabets are defined as the union of the labels associated to all the transitions and places in the module, respectively.

Module interconnection. The place p_h of the module Mod_i is connected to the transition t_{kl} of Mod_j, with a self-loop, if the event "p_k to p_l" is written in the cell defined by the sub-row r_h of e_i and the column e_j of the *permissive relation* table. In addition, transitions t_{ab}, t_{cd} of modules Mod_i, Mod_j, respectively, are merged into a single one if there exists a mark in the cell defined by the sub-row "*a* to *b*" and the sub-column "*c* to *d* ", respectively, in the *synchronous relation* table. Notice that by construction, the *IPN* model is consistent and conservative.

Example 6. Consider the system illustrated in Fig. 1.10, where each element is represented by a *PN module* in Fig. 1.12. In this figure, the modules Mod_1 - Mod_8 represent the *PN* modules of the motor, position-1 transmitter, position-2 transmitter, position-3 transmitter, Robot, gripper, valve and bottle, respectively.

For example, in order to build the *PN* module Mod_5 (Robot), the range is defined as $(RP_1, home, RP_2)$, and its initial condition as *home*, according to the *Elements description* table. Therefore, the places of Mod_5 are p_1^5, p_2^5, and p_3^5, and a mark is included in the place p_2^5, respectively. The transitions $t_{1,2}^5$, $t_{2,3}^5$, $t_{3,2}^5$ and $t_{2,1}^5$ and their directed arcs are included in Mod_5, since the states of the robot can change in one step from p_1^5 to p_2^5, from p_2^5 to p_3^5, from p_3^5 to p_2^5, and from p_2^5 to p_1^5, as indicated in the behavior table *ZN-103*.

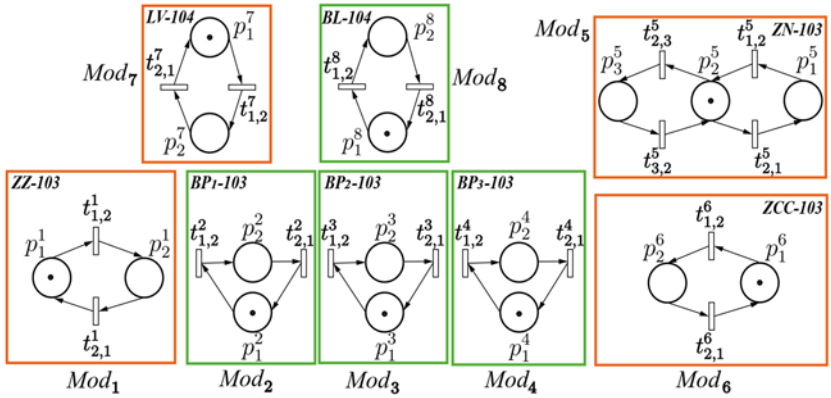


Fig. 1.12. Process PN modules of the bottle filling process.

The complete process *IPN* model is illustrated in Fig. 1.13, where the synchronous and permissive relations between *PN* modules are presented. For instance, the transitions $t_{2,1}^{2}$ and $t_{1,2}^{3}$ in Fig. 1.12 are merged into a transition t^{M_2, M_3}, since there exists a dashed arc from n_2 to n_3 in P_G (Fig. 1.11); moreover, two-way arcs are included from the place p_2^1 in Mod_1 to transitions $t^{M_{2,3}}$, $t^{M_{3,8,4}}$, and $t_{2,1}^{4}$, since there exists a relation between the actuator state *on* in *ZZ-103* and the variable events BP_1^p to BP_2^a and BP_2^a to BP_3^p in *BP1-103*, *BP2-103*, and *BP3-103* presented in the permissive relation table.

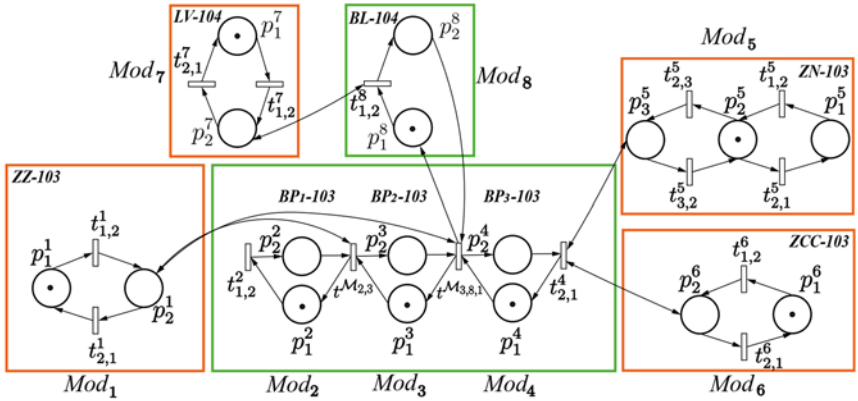


Fig. 1.13. Process *IPN* model of the bottle filling process.

1.5. Specification Construction

As stated in previous sections, the regulation control framework requires a specification modelled as a safe state machine *IPN* that describes desired output sequences in the plant. However, the specification design can be an issue since it can become rather complex for big systems, and its construction depends on the designer skill and knowledge of this control framework. In order to reduce this problem, in this section the design of a specification is presented in a way that it can be automatically computed from minimum information provided by a practitioner, who may not have knowledge in the modelling and control of DES. This methodology for building specifications was introduced in [17].

The methodology is obtained by defining sets of tasks and the devices that can perform each set, next the required plant conditions to execute each task are declared. Then, the practitioner fills a couple of tables with the necessary information required for the specification construction. The information in these tables is finally used to automatically generate an *IPN* specification.

1.5.1. Tasks and Work Stations

The design of specifications proposed in [17] is based on the idea that industrial automation systems can be split in a set of interacting work stations that are composed of the actuators capable of performing sets of different tasks. In the first step of the design, the practitioner identifies the required *tasks* to be performed by the plant, and for each task, the practitioner identifies the sequence of signals that the plant must produce in order to perform it. The set of actuators that can produce those signal sequences is identified as a *work station*. Tasks may include signals sequences that appear several times in one or more tasks, called *subtasks*. The formal definitions of these concepts are depicted below.

Definition 8. A *subtask* $subtask_a = (s^{\alpha_1},...,s^{\alpha_n})$ is an ordered set of operations $s^{\alpha},..., s^{\alpha_n} \in 2^{\Sigma_O}$, each one defined as a subset of actuators' output signals. The operations alphabet of a *subtask* is defined as $\Sigma_{sub}{}^{a} = \{ s^{\alpha_1},...,s^{\alpha_n} \}$. The set of all the subtasks of the plant is denoted as $\tau' = \{subtask_1,..., subtask_m\}$, and the alphabet of τ' is denoted as $\Sigma_{sub} = \{sub_1,..., sub_m\}$, where each $sub_i \in \Sigma_{sub}$ is defined as a new symbol associated with a subtask $subtask_i \in \tau'$.

Definition 9. A task $task_a = (s^{a_1},...,s^{a_n})$ is an ordered set of operations $s^{a_1},..., s^{a_n} \in 2^{\Sigma_O} \cup \Sigma_{sub}$, each one defined as a subset of actuators' output signals or subtasks symbols. The operations alphabet of a *task* is defined as $\Sigma_{task}{}^a = \{ s^{a_1},...,s^{a_n} \}$. The set $\tau = \{task_1,..., task_m\}$ is the set of all the tasks of the plant. A *task* represents a required sequence of actuators signals that the controlled plant must accomplish.

Definition 10. Consider the plant as a collection of *IPN* models of the devices set $D = \{d_1,...,d_n\}$ as described in [14], where $\Sigma_I{}^{d_k}$ and $\Sigma_O{}^{d_k}$ are the input and output alphabet of the k-th device, respectively. In practice, it holds that $\Sigma_O{}^{d_k} \cap \Sigma_O{}^{d_l} = \emptyset$, $\Sigma_I{}^{d_k} \cap \Sigma_I{}^{d_l} = \emptyset$, for all $k \neq l$. Given a set of tasks $\tau_i \subseteq \tau$, the work station of τ_i is defined as the smallest subset of devices $w_i \subseteq D$ such that $\Sigma^{\tau_i} \subseteq 2^{\Sigma_o{}^{wi}}$, where $\Sigma^{\tau_i} = \{s \mid s \in task, task \in \tau_i\}$ denotes the set of operations involved in τ_i and $\Sigma_o{}^{w_i}$ denotes the set of output symbols of all the devices in w_i.

It is assumed that the plant is split in a set of workstations $W = \{w_1,..., w_u\}$ covering all the tasks. Moreover, it is assumed that $\forall task_a, task_b \in \tau_i$, $s^{a_1} = s^{b_1}$, i.e., all the tasks of a work station have the same initial operation.

It is also possible to indicate when a task or subtask of a work station finish its sequence in order to coordinate with the tasks and subtasks of a different work station. For instance, if a task that manufactures a product is completed then the task that packages that finished product can begin its sequence, this way the first task can keep manufacturing another product at the same time the second task is packaging the finished product, i.e. work stations can work concurrently. In this method, the auxiliary signals that tell if a task or subtask is completed are defined as *flag signals*.

Definition 11. A flag signal in a $task_a$ is a duple $(s^{a_l}, flag_o{}^a)$, where $s_1{}^a$ is an operation of $task_a$ and $flag_o{}^a \in \Sigma_{flag}$, where Σ_{flag} is a set of symbols of virtual sensors fulfilling $\Sigma_{flag} \subseteq \Sigma_O{}^S \setminus \Sigma_O$. Flag signals can also be defined in subtasks in a similar way.

Along with flag signals, the activation of certain sensor signals can be used as conditions for the tasks or subtasks to perform their sequences, these conditions are named *event conditions* and are formally defined as follows.

Definition 12. An event condition of a $task_a$ is a triplet $(s_1^a, s_{(l+1)}^a, cond_f^a)$, where s_1^a and $s_{(l+1)}^a$ are operations of the $task_a$, and $cond_f^a$ is a logical expression which variables are elements of $\Sigma_{flag} \cup \Sigma_O$ and only logical operators $\wedge$ and $\vee$ are allowed. Here, $cond_f^a$ represents the necessary conditions of sensors in the plant and virtual sensors for the $task_a$ to advance from the operation s_l^a to the operation $s_{(l+1)}^a$. Event conditions can also be defined in subtasks in a similar way.

For practicality, the practitioner can summarize the information of each work station and its tasks in a *Tasks* table as shown in Table 1.5. A row in the Table 1.5 must be filled for each required task as follows:

- The first column describes the task's work station;
- The second column describes the name given to the task;
- The third column describes the operations sequence $(s_1^a, \ldots, s_r^a)$ that defines the task;
- The fourth column describes all the event conditions of the task's operations;
- and the fifth column describes the flag signals.

If there are subtasks a table for them must be filled in the same way as the Table 1.5.

Table 1.5. Summarized information for specification definition.

Work Station	Task Name	Operations Sequence	Event Conditions	Flag Signals
w_1	$task_1$	$(s_1^1, s_2^1, \ldots, s_r^1)$	$(s_l^1, s_m^1, cond_l^1), \ldots,$ $(s_p^1, s_q^1, cond_p^1)$	$(s_j^1, flag_l^1), \ldots,$ $(s_k^1, flag_o^1)$
⋮	⋮	⋮	⋮	⋮
w_u	$task_n$	$(s_1^n, s_2^n, \ldots, s_r^n)$	$(s_l^n, s_m^n, cond_l^n), \ldots,$ $(s_p^n, s_q^n, cond_p^n)$	$(s_j^n, flag_l^n), \ldots,$ $(s_k^n, flag_o^n)$

1.5.2. *IPN* Specification

Once the tables are filled their information is translated into an *IPN* model. For each work station, the following Algorithm 1 generates a state machine *IPN*, where the transitions are labelled based on the event

conditions and the places are labelled based on the operations alphabet and flag signals.

Algorithm 1. Computation of the *IPN* specification.

Initialize $P^S = \emptyset$, $T^S = \emptyset$, $Pre^S = \emptyset$, $Post^S = \emptyset$

FOR each work station DO

% A state machine is generated for each task %

FOR each $task_a = \{s_i^a\}$ of the work station DO

Define an ordered set of places $p_1^a,...,p_n^a$ (i.e., $P^S = P^S \cup \{ p_1^a,...,p_n^a \}$), where each operation s_i^a of the task is represented by a place p_i^a in the corresponding order.

IF there is a flag signal $(s_i^a, flag_k^a)$ THEN

Label p_i^a as $\varphi^S(p_i^a) = \{s_i^a, flag_k^a\}$

ELSE

Label p_i^a as $\varphi^S(p_i^a) = s_i^a$

ENDIF

% We represent the $\vee$ and $\wedge$ operators by adding transitions and joining labels %

FOR each event condition $cond_j^a$ related to two consecutive operations s_i^a and s_{i+1}^a DO

Denote the condition in a standard form as:

$cond_j^a = (q_{11} \wedge q_{21} \wedge ... \wedge q_{y1}) \vee (q_{12} \wedge q_{22} \wedge ... \wedge q_{y2}) \vee ... \vee (q_{1m} \wedge q_{2m} \wedge ... \wedge q_{ymm})$

Define a set T_j^a of m transitions (i.e., $T^S = T^S \cup T_j^a$)

Add to Pre^S an arc from p_i^a to each $t \in T_j^a$, and add to $Post^S$ an arc from each $t \in T_j^a$ to p_{i+1}^a

Label each $t \in T_j^a$ as $\lambda^S(t) = \{q_{1h}, q_{2h},...,q_{yhh}\}$

ENDFOR

For each pair of consecutive operations s_i^a and s_{i+1}^a without event conditions, define a transition t_i^a (i.e., $T^S = T^S \cup \{ t_i^a \}$), add to Pre^S an arc from p_i^a to t_i^a and add to $Post^S$ an arc from t_i^a to p_{i+1}^a.

Similarly, if there is no event condition involving the last operation and the first one, connect the last task place to the first one through a new transition.

ENDFOR

% Merge the tasks *IPN*s%

Merge the first places of each task into one place, denoted as p_{home} that represents the home position of the work station.

Set $M_0^S(p_{home}) = 1$ and $M_0^S(p) = 0$ for the rest of places.

% A state machine is generated for each subtask %

FOR each $subtask_b$ of the work station DO

Repeat the same steps as in the generation of state machines for tasks but using the information of the subtask table, without merging the subtasks and without connecting the last place to the first one, then set all the subtasks places initially unmarked.

ENDFOR

% Subtasks are merged with the rest of the *IPN*%

Add an arc from each $\bullet p$ s.t. $\varphi^S(p) = sub_b \in \Sigma_{sub}$ to the first place of $subtask_b$, and add an arc from the last place of $subtask_b$ to $p\bullet$

ENDFOR

Example 7. Consider again the Example 4 (Fig. 1.9). In this system, the drill must be turned on whenever a part is drop at the Driller station, next, the actuator of the drill station is extended to secure the part. After the part is perforated, the actuator is retracted and the arm can unload the station. The arm performs four tasks: 1) It moves untagged parts from the *In* position to the RFID station; 2) It moves tagged parts from the *In* position to the Drill station; 3) It transports tagged parts from the RFID to the Drill station; 4) It transports drilled parts from the Drill station to the *Out* conveyor's position. If there are parts in the *In* position, RFID and Driller stations at the same time, the arm must wait until the part in the Driller is perforated and then unload the driller.

In this case, the set of tasks performed by the arm is defined as $\tau_R = \{In\_2\_RFID,\ RFID\_2\_Driller,\ In\_2\_Driller,\ Driller\_2\_Out\}$, corresponding to the tasks: moving a part from the *In* position to the *RFID* station, from the *RFID* to the *Driller*, from the *In* position to the

Driller and from the *Driller* to the *Out* position, respectively. Tasks are formally defined as *In_2_RFID = (H, take, {XB, YA}, place)*, *RFID_2_Driller = (H, {XB, YA}, take, {XB, YB}, place)*, *In_2_Driller = (H, take, {XB, YB}, place)*, *Driller_2_Out = (H, {XB, YB}, take, {XA, YB}, place)*, where *H = {XA, YA, ZA, VA}* and the subtasks *take* and *place* represent the tasks taking-a-part-from-a-station and placing-a-part-on-a-station, which are performed by the actuators by reaching the sequences of positions *ZB,VB,ZA* and *ZB,VA,ZA*, respectively. These tasks define the work station *R = {X, Y, Z, V}*.

The event conditions are defined for each task as follows, *In_2_RFID: (H, take, (In∧¬tagIn∧SRFID))*, meaning that the subtask *take* is performed if there is a part at the *In* position, it is untagged and the *RFID* station is available; *RFID_2_Driller: (H, {XB,YA}, (RFID∧tagRFID∧SDrill))*, meaning that the arm goes to position *{XB,YA}* if there is a part at the *RFID* station, the part is tagged and the *Driller* station is available; *In_2_Driller: (H, take, (In∧ tagIn ∧SDrill))*, meaning that the subtask *take* is performed if there is a part in the *In* position, it is tagged and the *Driller* is available; *Driller_2_Out: (H, {XB,YB},(FDrill∨(In∧RFID∧SDrill)))*, meaning that the arm goes to position *{XB,YB}* if the *Driller* has finished its task or there is a part on the *In* position, at the *RFID* station and at the *Driller* station; and *({XB,YB},take,FDrill)*, meaning that the subtask *take* is performed if the *Driller* has finished its task. There are not flag signals. The set of tasks performed by the driller work station *D = {P,M}* is defined as τ_D *= {Drill}*, where *Drill = ({DA,*M_{off}*}, Mon,DB,DA)*. The event conditions for *Drill* are *({DA,Moff}, Mon, SDrill)* and *(DA, {DA,Moff}, ¬SDrill)*, meaning that the motor turns on when the Driller station is available and turns off when it is unavailable, respectively. The flag signal *(DA,FDrill)* means the flag symbol *FDrill* is activated when the Driller actuator is retracted.

Tables 1.6 and 1.7 describe tasks and subtasks, respectively. The *IPN* specification is obtained by applying the Algorithm 1, resulting in the *IPN* shown in Fig. 1.14. This *IPN* consists of two disconnected nets, the left one corresponds to the specification for the pneumatic arm, and the right one corresponds to the CNC Driller. The specification arm is formed by 4 place-transition chains, each one representing the specified tasks. The chains *ZB,VB,ZA* and *ZB,VA,ZA* represent the subtasks take and place.

Frequently, the *IPN* computed with the Algorithm 1 would not be a state machine, in such case it would be unsuitable for applying the existing tracking control synthesis algorithms [14-15]. However, an equivalent safe state machine *IPN* specification can be computed by expanding the reachability graph of the specification of the Algorithm 1.

Table 1.6. Tasks for Example 7.

Work Station	Task Name	Operations Sequence	Event Conditions	Flag Signals
R	*In_2_RFID*	*H, take, {XB,YA}, place*	*(H, take, (In ∧ ¬tagIn ∧ SRFID))*	
R	*RFID_2_Driller*	H, {XB,YA}, take,{XB,YB}, place	(H, {XB,YA}, (RFID ∧ tagRFID ∧ SDrill))	
R	*In_2_Driller*	*H, take, {XB,YB},place*	*(H, take, (In∧ tagIn∧ SDrill))*	
R	*Driller_2_Out*	*H,{XB,YB},take, {XA,YB}, place*	*(H, {XB,YB}, (FDrill∧ ¬SDrill) ∨(In∧ RFID∧ SDrill)) ({XB,YB},take,(Fdrill ∧ ¬SDrill))*	
D	*Drill*	*{DA,Moff},Mon, DB, DA*	*({DA,Moff},Mon, (SDrill)) (DA,{DA,Moff}, ¬SDrill)*	*(DA, FDrill)*

Table 1.7. Subtasks for Example 7.

Subtask Name	Operations Sequence	Event Conditions	Flag Signals
Take	*ZB,VB,ZA*		
Place	*ZB,VA,ZA*		

1.6. Control Synthesis

The design of a regulation controller was presented in [11-13]. In the sequel, the plant's sets, functions and markings will be denoted with a superscript ^{P} to distinguish them from those of the specification, denoted with a superscript ^{S}.

For the controller synthesis, three main steps are required:

1) Each specification reachable marking $M^S \in R(Q^S, M_0^S)$ is associated to a plant reachable marking $M^P \in R(Q^P, M_0^P)$ such that

$\varphi^S \boldsymbol{M}^S = \varphi^P \boldsymbol{M}^P$, this association can be described by a function Π: $R(Q^S, M_0^S) \rightarrow R(Q^P, M_0^P)$, defined such that $\Pi \boldsymbol{M}^S = \boldsymbol{M}^P$ (the computation of such function can be performed by an IPP ([11-13]);

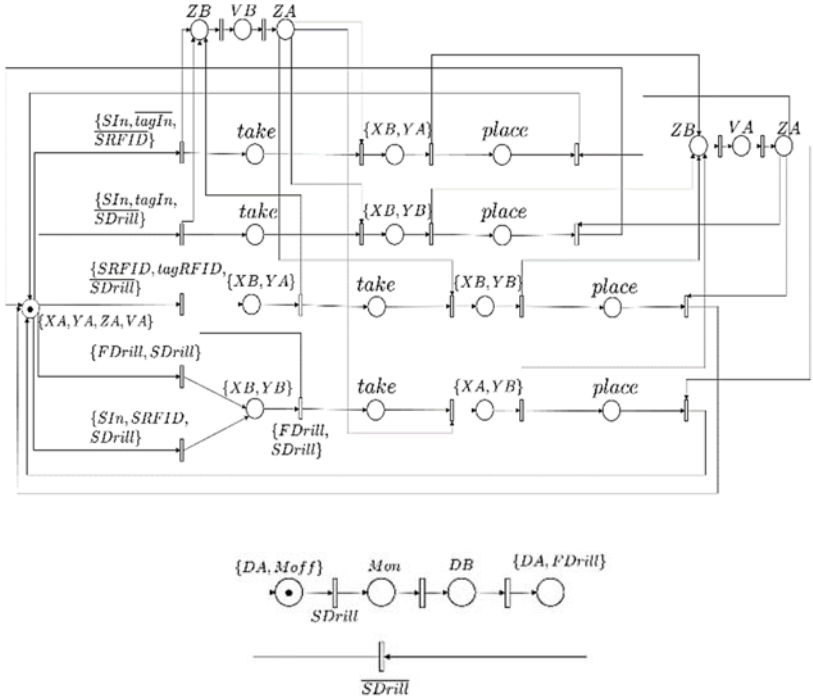


Fig. 1.14. *IPN* model obtained by applying the Algorithm 1 with the information in Tables 1.6 and 1.7. For the sake of clarity, only the symbols that change after each transition firing are shown.

2) Each specification transition $t_i^S \in \mathcal{T}^S$ is associated to a controllable plant firing sequence σ_i^P such that $\boldsymbol{M}_i^P[\sigma_i^P\rangle \boldsymbol{M}_j^P$, where $\boldsymbol{M}_i^S[t_i^S\rangle \boldsymbol{M}_j^S$, $\Pi \boldsymbol{M}_i^S = \boldsymbol{M}_i^P$ and $\Pi \boldsymbol{M}_j^S = \boldsymbol{M}_j^P$ (the sequences can be computed by exploring the plant reachability graph, for instance by using the A* algorithm ([11-13]);

3) Based on the knowledge of the mapping Π and the controllable sequences σ_i^P, a controller can be synthesized as an agent that indicates suitable plant input symbols so the sequences σ_i^P are enforced when required by the specification. This controller can be realized as a function on the plant and the specification markings or as an *IPN* ([11-13]).

The following algorithm (a generalization of Algorithm 4.1 of [14]) implements the computation of an *IPN* that represents the closed-loop system of the Plant under a proper tracking controller. Here, it is assumed that the Specification is a state machine *IPN*.

Algorithm 2. Calculation of the closed-loop *IPN*.

% Initialize the closed-loop system Q^{cl} as the union set of Q^P and Q^S %

Defined $P^{cl} = P^P \cup P^S$, $T^{cl} = T^P \cup T^S$, $\Sigma_I^{cl} = \Sigma_I^P \cup \Sigma_I^S$, $\Sigma_O^{cl} = \Sigma_O^P \cup \Sigma_O^S$, $\lambda^{cl} = \lambda^P \cup \lambda^S$, $\varphi^{cl} = \varphi^P \cup \varphi^S$, $\boldsymbol{Pre}^{cl} = diag(\boldsymbol{Pre}^P, \boldsymbol{Pre}^S)$, $\boldsymbol{Post}^{cl} = diag(\boldsymbol{Post}^P, \boldsymbol{Post}^S)$ and $\boldsymbol{M}_0^{cl} = [\boldsymbol{M}_0^P, \boldsymbol{M}_0^S]^T$.

% Define the mapping relating plant and specification markings with the same output %

Define a mapping $\Pi: RS(N^S, \boldsymbol{M}_0^S) \rightarrow RS(N^P, \boldsymbol{M}_0^P)$ as follows:

FOR each specification place $p_i^S \in P^S$ DO

Compute a Plant marking $\boldsymbol{M}_i^P$ reachable from $\boldsymbol{M}_0^P$ such that the set of symbols produced by $\boldsymbol{M}_i^P$, denoted as $\varphi^P(||\boldsymbol{M}_i^P||) = \{o \in \Sigma_O^P \mid o \in \varphi^P(p^P) \text{ where } M_i^P(p^P) > 0\}$, is coherent with the symbols produced by p_i^S (i.e., $\varphi^S(p_i^S) \cap \Sigma_O^{Pact} = \varphi^P(||\boldsymbol{M}_i^P||) \cap \Sigma_O^{Pact}$ and $\varphi^S(p_i^S) \cap \Sigma_O^{Psen} \subseteq \varphi^P(||\boldsymbol{M}_i^P||) \cap \Sigma_O^{Psen}$) and $\boldsymbol{M}_i^P$ does not enable actuators' uncontrollable transitions, i.e., transitions in the set $\{t \in T^{Pact} | \lambda^P(t) = \varepsilon\}$.

Define $\Pi(\boldsymbol{M}_i^S) = \boldsymbol{M}_i^P$, where $\boldsymbol{M}_i^S$ is the specification reachable marking at which p_i^S is marked (i.e., $M_i^S(p_i^S) = 1$ and $M_i^S(p_j^S) = 0, \forall p_j^S \in P^S \setminus \{p_i^S\}$).

ENDFOR

% Compute the plant sequences that bisimulate the specification transitions and define the *IPN* nodes to enforce such sequences %

FOR each specification transition $t_i^S \in T^S$ DO

Let $p_i^S = {}^\bullet t_i^S$ and $p_j^S = t_i^{S\bullet}$ be the input place and the output place of t_i^S, respectively. Moreover, denote as M_i^S the specification reachable marking such that p_i^S is marked at M_i^S and p_j^S is marked at M_j^S, respectively.

Compute a Plant firing sequence σ_i^P such that $\boldsymbol{M}_i^P[\sigma_i^P\rangle \boldsymbol{M}_j^P$, where $\boldsymbol{M}_i^P = \Pi(\boldsymbol{M}_i^S)$ and $\boldsymbol{M}_j^P = \Pi(\boldsymbol{M}_j^S)$, and every marking $\boldsymbol{M}_i^{P\prime}$ that is visited

when firing σ_i^P does not enable actuators' uncontrollable transitions that are not included in σ_i^P.

Define an ordered set of places $p_{i1}, p_{i2},..., p_{ik}, p_{i(k+1)}$, where k is the length of controllable transitions in σ_i^P. Connect t_i^S to p_{i1}. Connect each place p_{ir} to a new transition t_{ir} for any $r \in \{1,2,...,k\}$, and connect each transition t_{ir} to place $t_{i(r+1)}$ for any $r \in \{1,2,...,k\}$. Connect the place $p_{i(k+1)}$ to each transition in $p_j^S \bullet$

Label each r-th place p_{ir}, for any $r \in \{1,2,...,k\}$, with the input labels of the r-th controllable transition in σ_i^P, i.e., $\varphi^{cl}(p_{ir}) = \lambda^P(\sigma_i^P(r))$, where $\sigma_i^P(r)$ denotes the r-th controllable transition in the sequence σ_i^P.

Label each r-th transition t_{ir}, for any $r \in \{1,2,...,k\}$, with the output labels of the places marked after firing the r-th controllable transition in σ_i^P, i.e., $\lambda^{cl}(t_{ir}) = \varphi^P(||\boldsymbol{M}_i^{P\prime}||)$, where $\boldsymbol{M}_i^P[\sigma_i^P(1..r)\rangle \boldsymbol{M}_i^{P\prime}$ and $\sigma_i^P(1..r)$ denotes the subsequence formed with the first transitions of σ_i^P until the r-th controllable transition.

ENDFOR

% Adjustment for attribution specification places%

FOR each specification place p_i^S having more than one input specification transition DO

Denote as $\{t_a^S, t_b^S, ..., t_c^S\} = \bullet p_i^S$

Merge the corresponding places $p_{a(k+1)}, ..., p_{c(k+1)}$

ENDFOR

FOR each specification place p_i^S DO

IF p_i^S is initially marked THEN a token is added to the place $p_{i(k+1)}$ ENDIF

ENFOR

Remove all the output symbols from the original specification places.

The resulting *IPN* Q^{cl} represents the closed-loop system.

The controller can be computed from Q^{cl} by removing the Plant's nodes.

Example 8. Consider again the manufacturing cell of Example 4 (Fig. 1.9). Focusing on its pneumatic arm with the specification shown at Fig. 1.14, this specification is not a state machine, however, it can be

easily transformed into an equivalent state machine by substituting the places *take* and *place* by the chains describing the corresponding subtasks (i.e., the chain of places and transition with labels *ZB, VB, ZA* and *ZB, VA, ZA*).

By applying the Algorithm 2, the closed-loop system shown in Fig. 1.15 is computed. Algorithm 2 removes the original output labels from the specification places, but they are maintained in Fig. 1.15 (just by adding the character ′) in order to explain the closed-loop behavior.

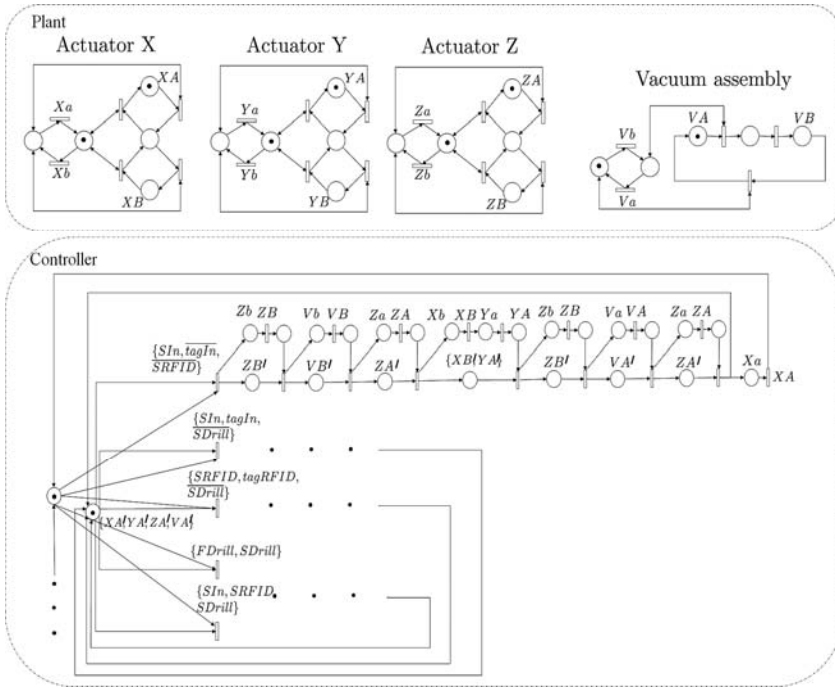


Fig. 1.15. *IPN* model of the closed-loop system of the pneumatic arm of Fig. 1.9 under the specification of Fig. 1.14. For the sake of clarity, only the first task with its control is drawn in detail. Similarly, at the new controller transitions, only the symbols that change after reaching the related plant markings are shown.

Let us describe the closed-loop behavior. From the initial marking, if the conditions for the first task holds (*SIn,*¬*tagIn,*¬*SRFID*), the tokens at the initial marking in the controller are moved from the initially marked

places to the places labelled with *Zb* and *ZB′*. At this marking, the transition *ZB′*• is not enabled, but the token at the place with label *Zb* enables the transition *Zb* in the Plant (actuator Z). The firing of this transition will eventually drive the actuator to the position *ZB*, i.e., the Plant has reached a marking having the required output. At this new Plant marking, the transition *Zb*• becomes enabled, moving the token from the place *Zb* to the place *Zb*••. From this new marking, the transition *ZB′*• becomes enabled and the specification can evolve. In this way, the controller provides symbols to the Plant in order to reach the marking that produces the required output symbols, when this marking is reached, the specification is allowed to evolve, requiring new output symbols.

The *IPN* controller can be translated to a ladder diagram for a PLC implementation. An example of this procedure can be found in [14]. In this, an internal coil is defined for each non-plant place (a coil is ON if the corresponding place has a token), a network (line) is defined for the places' initial condition which is executed after pressing a reset push-button, a network is added for each controller transition (Pre arcs are translated to NO switches and Reset coils, Post arcs are translated to Set coils), and other networks (lines) are added for defining the PLC inputs and outputs.

1.7. Towards Decentralized Control

Frequently, the number of components in industrial systems (sensor, actuators, machines, conveyors, etc.) to control is large, for this reason, the implementation of a centralized controller becomes complex and usually prohibitive due to hardware limitations (for instance, the limited number of input/output ports that commercial PLCs can handle). In order to deal with this problem, different decentralized architectures can be used. Among the most frequently adopted are:

- Distributed control without communication: different controllers (implemented in different devices) control sub-assemblies (sections) of the plant, there is no communication among the controllers;
- Distributed control in an industrial network: different controllers control different sub-assemblies of the plant but they are integrated in an industrial network, allowing to communicate discrete variables;
- Coordinated control: similar to the distributed scheme but with the addition of a coordinator agent (for instance, a SCADA system) in a

hierarchically upper level that supervises and provides directions to the local controllers.

Each of the previous decentralized control schemes requires a different control synthesis strategy. Let us provide some preliminarily ideas.

Frequently, the plant can be naturally seen as a distributed system, i.e., a collection of independent components that can evolve concurrently. Moreover, the specification of the complete system is frequently proposed as a set of sub-sequences defined for clusters of actuators and high-level sequences involving the complete system. For instance, if we identify a pneumatic-arm and pneumatic-machines, sequences for the arm and each of the machines are firstly defined to control these sub-assemblies, and then a high-level sequence involving the arm and the machines is proposed. In this sense, [15] proposes that the specification of large automation systems is in fact a distributed system, i.e., a distributed-specification.

1.7.1. Distributed Control without Communication

In this scheme, there is a collection of sub-specifications, each one describing sequences (and/or selections of sequences) of desired output signals involving different actuators. Thus, the plant can be split into sub-assemblies according to the actuators involved in the specifications. The interaction of the sub-assemblies in the closed-loop system appears due to the interaction in the plant and sensor/switches whose activation trigger more than one sub-sequence. Let us provide a formal definition.

Definition 13. A distributed-specification Q^s is the synchronous product of a disjoint collection of r subspecifications, denoted as $Q^{s1},...,Q^{sr}$, where each subspecification is defined as in Definition 5. Moreover, two different subspecifications cannot involve the same actuator, i.e., $\Sigma^{si}{}_O \cap \Sigma^{sj}{}_O \cap \Sigma^{act}{}_O = \emptyset$ for any $\Sigma^{si}{}_O \neq \Sigma^{sj}{}_O$, where $\Sigma^{act}{}_O$ denotes the alphabet of actuators' output signals.

1.7.2. Distributed Control in a Network

By connecting the controller devices to an industrial network, sending messages between controller agents becomes possible. Thus, in this scheme messages can be communicated at the specification level. Therefore, the interaction of the sub-assemblies in the closed-loop

system appears not only because the interaction in the plant but also due to the messages at the specification level.

Definition 14. A distributed-networked-specification Q^s is a safe and live *IPN* composed of: (1) the synchronous product of a disjoint collection of r subspecifications, denoted as $Q^{s1},...,Q^{sr}$, defined as in Definition 5, such that two different subspecifications do not involve the same actuator (i.e., $\Sigma^{si}{}_O \cap \Sigma^{sj}{}_O \cap \Sigma^{act}{}_O = \emptyset$ for any $\Sigma^{si}{}_O \neq \Sigma^{sj}{}_O$); (2) additional buffer places (messages) interconnecting subspecifications such that, for any buffer place p, $|{}^\bullet p| = |p^\bullet| \geq 1$, ${}^\bullet p$ belongs to a subspecification Q^{si} and $p^\bullet$ belongs to another Q^{sj}.

1.7.3. Coordinated Control

In a SCADA system, several local controller devices can be implemented in such a way that each device controls a sub-assembly (e.g., a pneumatic-arm or a machine) to describe particular sequences and a coordinator controller, implemented in the SCADA system, monitors and gives directions to the local controllers. In this scheme, the communication occurs between the local controllers and the coordinator. Here, the coordinator neither directly control actuators nor observe sensor signals.

Definition 15. A coordinated-specification Q^s is a safe and live *IPN* composed of: (1) a specification Q^{sC}, named coordinator, as defined in Definition 5 whose output alphabet is given by $\Sigma^C = \Sigma^{Csen} \cup \Sigma^{Ccor}$, where Σ^{Csen} denotes the Plant's sensor signals and $\Sigma^{Ccor} \cap \Sigma^p = \emptyset$; (2) a collection of r local subspecifications $Q^{s1},...,Q^{sr}$, where each subspecification is as defined in Definition 5 and whose output alphabet is a subset of $\Sigma^p \cup \Sigma^{Ccor}$.

Notice that in the distributed-specification and the networked-specification each subspecification involves different actuators, while in the coordinated-specification this condition is dropped, i.e., two local subspecifications may involve common actuators. Another difference between the corresponding architectures will appear in the implementation, in particular, in the coordinator scheme the coordinator can be implemented in a device that is not connected to the actuators and plant sensors, instead, it can be connected through a network to the local controller devices.

Example 9. Fig. 1.16 shows a Robotic Cell at the Manufacturing Lab in Tecnologico de Monterrey, which is simplified in the schematic of Fig. 1.17.

Fig. 1.16. Robotic manufacturing cell at the Manufacturing Lab.

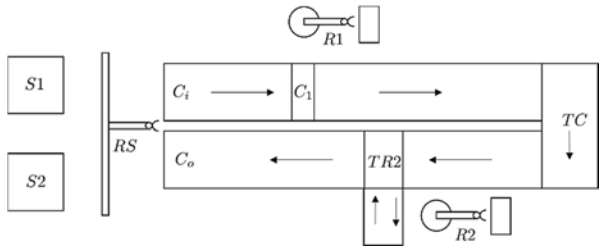


Fig. 1.17. Diagram of the simplified manufacturing cell.

This simplified system includes a store with two slots (positions) denoted as *S1* and *S2*, and a cartesian robot-arm (*RS*) with a gripper (*G*), a conveyor (*C*) split in two sections that transport pallets, a FANUC M-10iA robot arm (*R1*) that takes a component from a local buffer (not modelled) and place it on the pallet (when the pallet is located at position C_1), a FANUC delta M-1iA robot arm (*R2*) that takes a component from a local buffer (not modelled) and place it on the pallet (when the pallet is located in its working area on a conveyor's short branch) and two conveyor conveys (*TC*) and (*TR2*). The store robot *RS* is capable of retrieving and storing pallets in the store slots *S1* and *S2*, and of placing/picking pallets on/from the conveyor. In this model, one slot is used to store an empty pallet and the other is used to store a pallet with the complete assemble. There are key sites on the conveyor: the position

C*i* on the upper section is the entrance, the robot *R1* can place a component on the pallet when this is at position *C1* on the upper section, transfer *TC* moves pallets from the end of the upper section to the lower section (a physical barrier at the end of the conveyor's upper section prevents pallets to fall down), transfer *TR2* moves pallets from the conveyor's lower section to a short conveyor's branch on which the pallet can reach the working area of robot *R2*, finally, the position C_o on the lower section is the exit (a physical barrier at the end of the conveyor's lower section prevents pallets to fall down).

Fig. 1.18 represents the *IPN* model of the simplified manufacturing cell. This model was obtained by following the bottom-up modelling methodology of [20]. Notice that each robot (*RS*, *R1* and *R2*) is modelled as a state machine, where each transition represents the execution of a particular movement describing a trajectory that was previously programmed, moreover the places with subindex *H* represent home positions, the places with other subindexes represent other particular positions where the robots can take or leave a pallet or a component. Places P_u and P_d represent states of a pneumatic actuator that locks the position C_1 on the conveyor's upper section, thus no pallet is allowed to leave the position C_1 when the actuator is at state P_u. Places C_6 and C_7 represent a key position on the conveyor's lower section where the transfer *TR2* deviate a pallet to a short conveyor's branch to reach the working area of robot *R2*. This conveyor's branch is controlled by the places $TR2_a$, $TR2_H$ and $TR2_b$ that represent operation in backward direction, stop and operation in forward direction, respectively. Places $S1_f$ and $S2_f$ (resp. $S1_e$ and $S2_e$) denote when slot *S1* and *S2* has a pallet (resp. is empty), respectively. Places G_o and G_c denote when the gripper is open or closed, respectively. These places share input and output transitions that are connected to robot's and conveyor's places, representing permissive relations and synchronizations. For instance, the state of slot *S1* changes from $S1_f$ to $S1_e$ (i.e., it becomes empty) if the gripper simultaneously changes from open to closed and the store robot is at position RS_1, i.e., the robot takes the pallet from *S1*.

The *IPN* plant model of Fig. 1.18 can be seen as a collection of sections involving different machines and devices, as shown in Fig. 1.19, which are connected through the conveyor.

Fig. 1.20 shows a distributed specification, consisting of one machine specification for each section of Fig. 1.19 and a process specification.

Notice that machine specifications involve different actuators (robots, transfers, etc.), thus the specification is machine-disjoint.

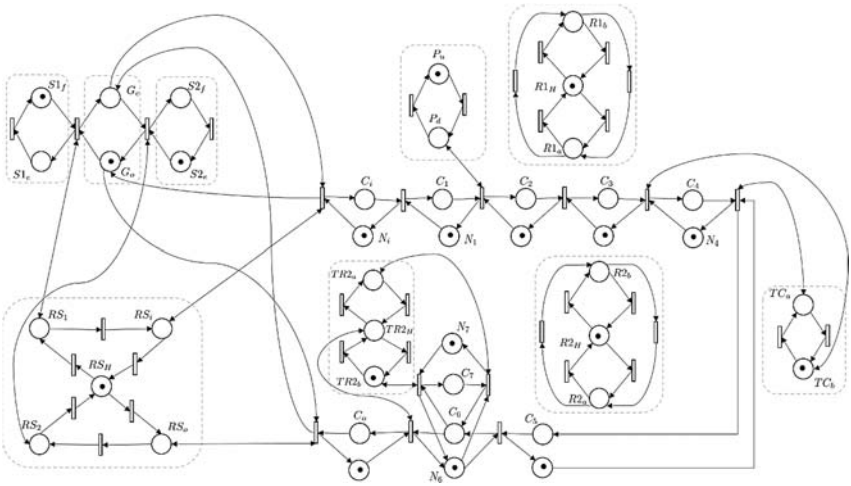


Fig. 1.18. Plant model in *IPN*, representing the manufacturing model of Fig. 1.17.

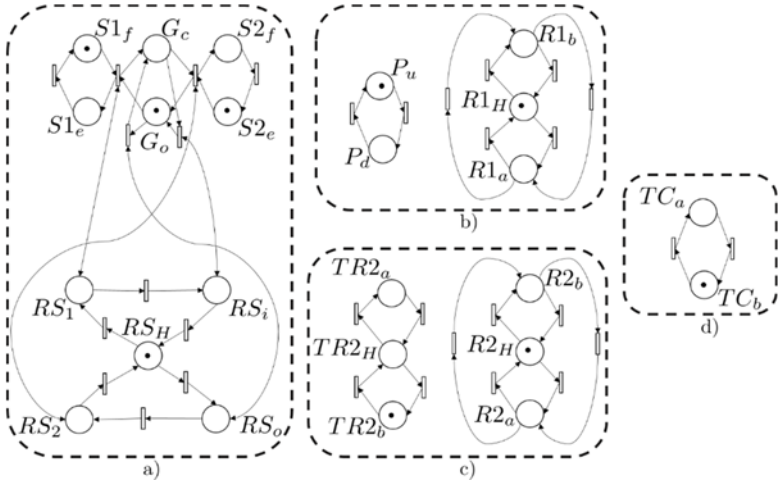


Fig. 1.19. Plant sections: a) storage section including the cartesian robot *RS*, its gripper and two slots, b) *R1* section including the actuator *{P_d, P_u}*, c) *R2* section including the transfer *TR2*, and d) the transfer section.

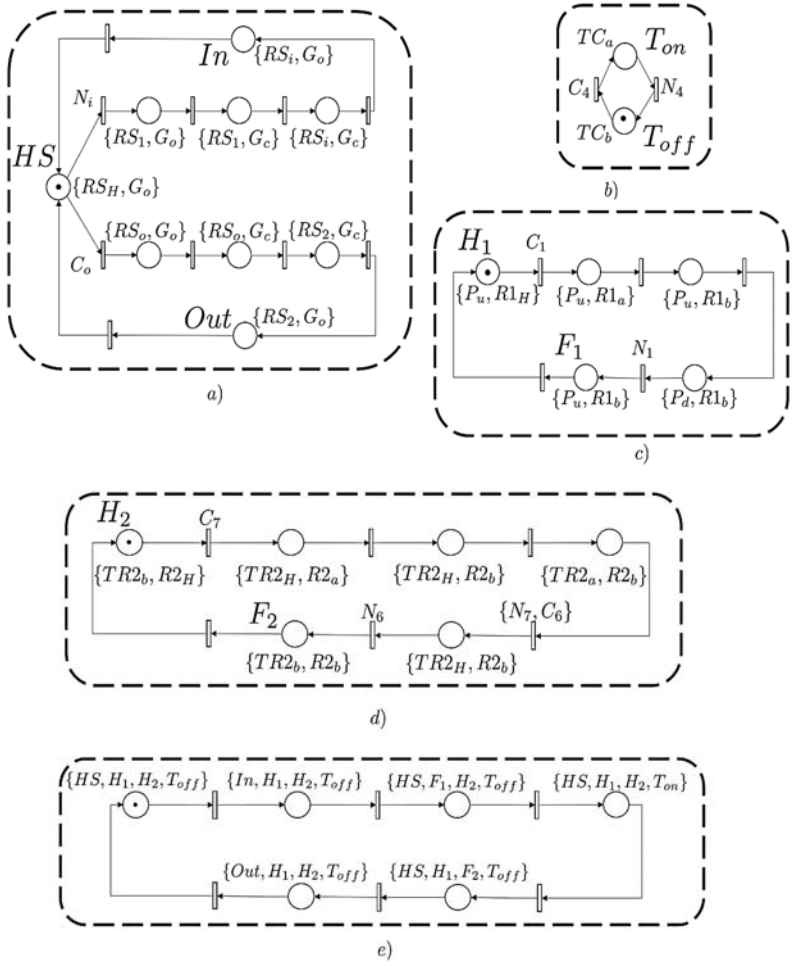


Fig. 1.20. Distributed specification, a) is the specification for the storage section, b) is the specification for the conveyor's transfer, c) and d) are the specifications for sections *R1* and *R2*, and e) is the process specification. Tasks symbols are written in large letters in the machine specifications.

By applying the Algorithm 2 for each machine specification, the closed-loop system of Fig. 1.21 is obtained.

As an advantage, the synthesis of this architecture is simple, moreover, the local controllers can be independently implemented, even in control devices of different manufacturers. Nevertheless, this architecture does not guarantee the fulfilment of the process specification. For instance,

the closed-loop system of Fig. 1.21 allows a pallet to cross the conveyor's position C_1 without receiving the component from $R1$, as required by the process specification (described by the task symbol F_1 at the third place of Fig. 1.20 e).

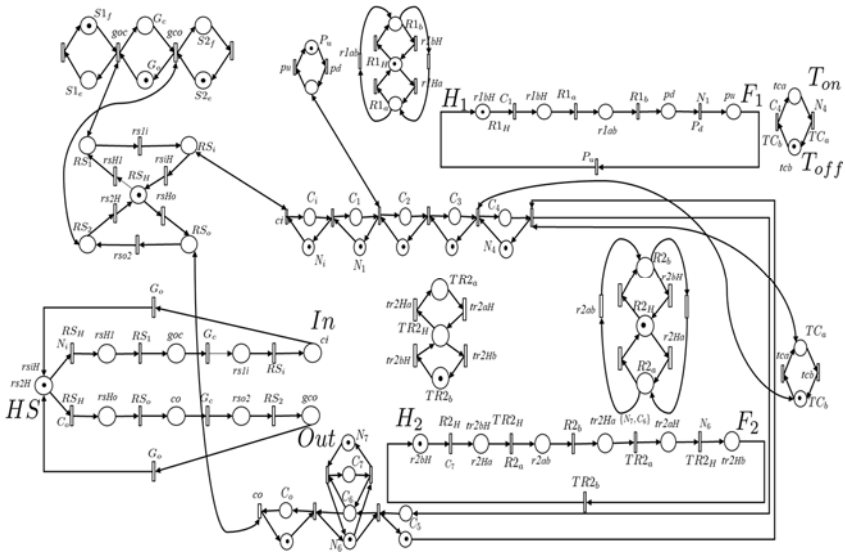


Fig. 1.21. Closed-loop system of the manufacturing cell without communication. For clarity, implicit places have been removed.

The closed-loop system of Fig. 1.21 is enriched by adding message places as described in the abstraction shown in Fig. 1.22, where the closed-loop behaviors of the Store, Robot 1, Robot 2 and Transfer sections are represented in an abstract way but maintaining the language projected on the tasks alphabet Σ_O^{Stasks}. In detail, the place $M_{S\text{-}R1}$ in Fig. 1.22 represents a message sent by the $R1$ section controller to the store section controller, in such a way that the store starts a sequence to locate a pallet on the conveyor only if $R1$ has previously performed an assembly. Places $M_{R2\text{-}TC}$ and $M_{TC\text{-}R2}$ represent messages from the transfer section controller to the $R2$ section controller, imposing that the transfer moves one pallet by each time that $R2$ performs an assembly.

By adding the message places described in Fig. 1.22 to the closed-loop system of Fig. 1.21, the process specification can be guaranteed. On the

other hand, the definition of messages may introduce drawbacks. First, an industrial network is required to implement the message communication, which may impose limitations in the hardware that can be used. Second, the message places may introduce deadlocks, since the reachability space is cut. Third, the synthesis of message places to enforce a process specification is not trivial and requires further analysis.

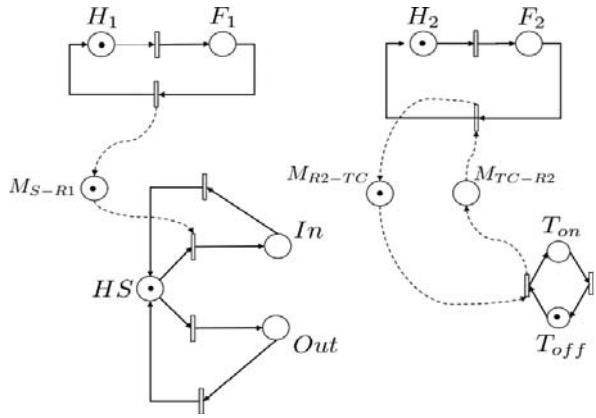


Fig. 1.22. Communication between controllers. A token in place $M_{S\text{-}R1}$ represents a message from the *R1* section controller to the store section controller. A token in place $M_{R2\text{-}TC}$ (resp. $M_{TC\text{-}R2}$) represents a message from the *R2* (resp. transfer) section controller to the transfer (resp. *R2*) section controller.

Finally, consider the closed-loop system of Fig. 1.21, (without communication), derived for the machine-specifications of Fig. 1.20. By applying the Algorithm 2 for the process specification of Fig. 1.20e to the machine specifications, the closed-loop system under the coordination control is obtained as shown in Fig. 1.23. In this, for the sake of readability, the machine specifications are shown as abstractions maintaining the language generated by the tasks alphabet.

In the closed-loop system, the fulfilment of the process specification is guaranteed by the coordination control. As a drawback, the implementation requires a controller device for the coordination in a network with the devices where the local controllers are implemented, which may impose limitations in the hardware that can be used.

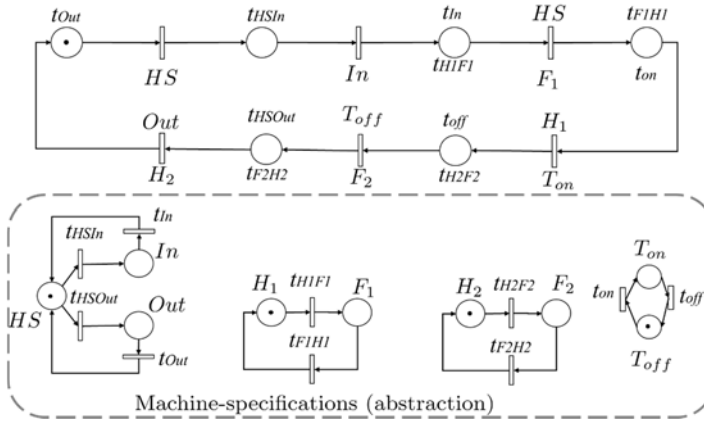


Fig. 1.23. Closed-loop coordination. For implementation and simulation, the labels at the abstracted-plant transitions must be properly interpreted. For instance, the label t_{Out} must be applied to each transition in the storage-section specification between places *Out* and *HS*.

1.8. Conclusions and Future Work

In this chapter, the main concepts of the regulation control framework for industrial automation systems were presented and discussed. The objective of this framework is to design a controller that enforces the plant to perform predefined tasks, where these tasks and the plant are modelled as *IPN*. Modelling methodologies for electro-pneumatic systems and processes systems as *IPN* were presented, as well as a methodology for the automatic generation of specifications, which only require the engineer to provide minimum information about the plant and its desired behaviour. The design of a centralized controller for the regulation control framework was described and illustrated. However, the synthesis of controllers for distributed specifications, that is, for a set of local specifications for large plants, still requires further investigation. Even so, some architectures with their respective advantages and drawbacks are proposed in this context.

Acknowledgements

The research leading to these results has received support from the Conacyt Fondo Sectorial de Investigacion para la Educacion, project number 288470.

References

[1]. R. J. Ramadge, W. M. Wonham, Supervisory control of a class of discrete event processes, *SIAM Journal on Control and Optimization*, Vol. 25, Issue 1, 1987, pp. 206-230.

[2]. L. E. Holloway, B. H. Krongh, A. Giua, A survey of Petri net methods for controlled discrete event systems, *Discrete Event Dynamic Systems*, Vol. 9, Issue 2, 1997, pp. 151-190.

[3]. M. Iordache, P. Antsaklis, A survey on the supervision of Petri nets, in *Proceedings of the Workshop on the Control of Hybrid and Discrete Event Systems*, Miami, FL, USA, June 21, 2005, pp. 61-80.

[4]. A. Giua, Supervisory control of Petri nets with language specifications, in Control of Discrete-Event (C. Seatzu, M. Silva, J. van Schuppen, Eds.), *Springer*, 2013, pp. 235-255.

[5]. W. M. Wonham, K. Cai, K. Rudie, Supervisory control of discrete-event systems: A brief history, *Annual Reviews in Control*, Vol. 45, 2017, pp. 250-256.

[6]. A. Giua, F. DiCesare, M. Silva, Generalized mutual exclusion constraints on nets with uncontrollable transitions, in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'92)*, Chicago, IL, USA, 18-21 October 1992, pp. 974-979.

[7]. F. Basile, R. Cordone, L. Piroddi, Integrated design of optimal supervisors for the enforcement of static and behavioral specifications in Petri net models, *Automatica*, Vol. 49, Issue 11, 2013, pp. 3432-3439.

[8]. Y. F. Chen, Z. W. Li, M. Khalgui, O. Mosbahi, Design of a maximally permissive liveness-enforcing Petri net supervisor for flexible manufacturing systems, *IEEE Transactions on Automation Science and Engineering*, Vol. 8, Issue 12, 2011, pp. 374-393.

[9]. Z. Li, N. Wu, M. Zhou, 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*, Vol. 42, Issue 4, 2012, pp. 437-462.

[10]. C. Seatzu, M. Silva, J. H. van Schuppen, Control of Discrete-Event Systems: Automata and Petri Nets Perspectives, *Springer*, 2012.

[11]. G. Ramírez-Prado, A. Santoyo, A. Ramírez-Treviño, O. Begovich, Regulation problem in discrete event systems using interpreted Petri nets, in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'2000)*, Nashville, TN, USA, 8-11 Oct. 2000, pp. 2174-2179.

[12]. J. F. Sánchez-Blanco, A. Ramírez-Treviño, A. Santoyo, Regulation control in interpreted Petri nets using trace equivalence, in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'04)*, The Hague, Netherlands, 10-13 Oct. 2004, pp. 1843-1848.

[13]. A. Santoyo, I. Jiménez-Ochoa, A. Ramírez-Treviño, A complete cycle for controller design in discrete event systems, in *Proceedings of the IEEE Int.*

Conference on Systems, Man and Cybernetics (SMC'01), Tucson, AZ, USA, 7-10 Oct. 2001, pp. 2688-2693.

[14]. C. R. Vázquez, J. A. Gómez-Castellanos, A. Ramírez-Treviño, Petri nets tracking control for electro-pneumatic systems automation, in Informatics in Control, Automation and Robotics (O. Gusikhin, K. Madani, Eds.), *Springer*, 2019.

[15]. D. Guevara-Lozano, C. Renato Vázquez, A. Ramírez-Treviño, Towards decentralized tracking control for Petri nets, in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA'19)*, Zaragoza, Spain, 10-13 Sept. 2019, pp. 428-435.

[16]. D. Rozo-Ibañez, J. Ruiz-León, D. Guevara-Lozano, C. R. Vázquez, Petri net modelling of industrial processes from a P&ID description, in *Proceedings of the International Conference on Control, Decision and Information Technologies (CoDIT'20)*, Prague, Czech Republic, 29 June – 2 July, 2020.

[17]. D. Guevara-Lozano, C. R. Vázquez, A. Ramirez-Treviño, Automatic specification generation for tracking control in interpreted Petri nets, in *Proceedings of the International Conference on Control, Decision and Information Technologies (CoDIT'20)*, Prague, Czech Republic, 29 June – 2 July, 2020.

[18]. R. David, H. Alla, Discrete, Continuous, and Hybrid Petri Nets, *Springer*, 2010.

[19]. A. Ramírez-Treviño, I. Rivera-Rangel, E. López-Mellado, Observability of discrete event systems modeled by interpreted Petri nets, *IEEE Transactions on Robotics and Automation*, Vol. 19, Issue 4, 2003, pp. 557-565.

[20]. I. Rivera-Rangel, A. Ramírez-Treviño, E. López-Mellado, Building reduced Petri net models of discrete manufacturing systems, *Mathematical and Computer Modelling*, Vol. 4, Issues 8-9, 2005, pp. 923-937.

[21]. M. Toghraei, Principles of P&ID development: the tips provided here will streamline efforts to develop piping & instrumentation diagrams, *Chemical Engineering*, Vol. 121, Issue 4, 2014, pp. 62-72.

[22]. A. Ramirez-Treviño, E. Ruiz-Beltran, I. Rivera-Rangel, E. Lopez-Mellado, Online fault diagnosis of discrete event systems. A Petri net-based approach, *IEEE Transactions on Automation Science and Engineering*, Vol. 4, Issue 1, 2007, pp. 31-39.

[23]. L. Feng, W. M. Wonham, Supervisory control architecture for discrete-event systems, *IEEE Transactions on Automatic Control*, Vol. 53, Issue 6, 2008, pp. 1449-1461.

Index

H

I

J

K

L

M

N

O

V

W

IFSA

Advances in Robotics and Automatic Control: Reviews Volume 2

Sergey Y. Yurish, Editor

The 2nd volume of *'Advances in Robotics and Automatic Control: Reviews'* Book Series contains 5 chapters written by 12 authors from 5 countries: Australia, Egypt, Mexico, Russia and Spain. But it is not a set of reviews. As usually, each chapter contains the extended state-of-the-art followed by new, unpublished before, obtained results.

This book ensures that our readers will stay at the cutting edge of the field and get the right and effective start point and road map for the further researches and developments. In order to offer a fast and easy reading, every chapter in this book is independent and self-contained. All chapters have the same structure: first an introduction to specific topic under study; second particular field description including applications. Each of chapter is ending by well selected list of references with books, journals, conference proceedings and web sites.

With this unique combination of information in each volume, the *'Advances in Robotics and Automatic Control: Reviews'* book Series will be a valuable tool for those who are involved in research and development of different industrial robots, automation and robotic systems.

ISBN 978-84-09-25862-8