Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e matematica Applicata

Dottorato di Ricerca in Ingegneria dell'Informazione
XI Ciclo – Nuova Serie

TESI DI DOTTORATO

Issues in Modeling and Identification of Discrete Event Systems

CANDIDATO: **JOLANDA COPPOLA**

TUTORS: **PROF. PASQUALE CHIACCHIO,**
ING. FRANCESCO BASILE

COORDINATORE: **PROF. ANGELO MARCELLI**

Anno Accademico 2011 – 2012

Contents

To my husband:
this thesis has been made because
he is able to understand before of everybody
what will be clear to the others only later.

Chapter 1

Preface

The term Discrete Event System (DES) was introduced in the early 1980s to identify an increasingly important class of dynamic systems in terms of their most critical feature: the fact that their behavior is governed by discrete events occurring asynchronously over time and solely responsible for generating state transitions. Examples of such behavior abound in technological environments such as computer and communication networks, automated manufacturing systems, air traffic control systems, C^3I (Command, Control, Communication, and Information) systems, advanced monitoring and control systems in automobiles or large buildings, intelligent transportation systems, distributed software systems, material handling systems and so forth. The operation of such environments is largely regulated by human-made rules for initiating or terminating activities and scheduling the use of resources through controlled events, such as hitting a keyboard key, turning a piece of equipment "on", or sending a message packet. In addition, there are numerous uncontrolled randomly occurring events, such as a spontaneous equipment failure or a packet loss, which may or may not be observable through sensors.

DESs are particulary used in the field of the manufactured systems, handling systems and transportation systems: even if such system are being studying for long time, because of their complexity, they still present many issues that attract research interest.

In particular this dissertation focuses about handling system modeling and DES identification.

Obtaining a good model of a system (both time-driven and event-driven) allows to more easily execute operations as performance analysis, control, monitoring of system evolution. However, in some cases modeling of a system is not simple because of several complications due to the behavior of the system or of the context it belongs to.

As example, sometimes, especially in the context of material handling and transportation, systems present both an event-driven and a time-driven behavior. In all that cases a very hight accuracy is not requested it is usual neglect the latter and “looking” at the system as a DES (as example modeling a handling system it is possible to be interested in knowing if a vehicle is or not in a zone of the path while it is not important to know its exactly position). When the time-driven behavior plays a fundamental role in the obtaining the overall system performance, such dynamics can not be neglected and they have to be explicitly modeled. This is the case, as example, of the automated warehouse systems, where the handling subsystem, as will be shown in the rest of this dissertation, presents time-driven dynamics that greatly influence the warehouse’s performance. Consequently a new way to model the system behavior has to be used.

However, there are situation in which the difficult issue is not choosing the right formalism to model the system but it is the modeling itself.

This is typical in many practical contexts, where it can occur that one has to work with unknown ready made systems and no documentation about their behavior is available, or the model of a very complex system is needed. In these and other cases modeling becomes hard and another way to obtain the model of the system is needed: automated identification can be the solution.

1.1 Thesis contribution

In the modeling environment, contribution of this thesis consists in presenting a new methodology to obtain a model oriented to the control and performance analysis of complex material handling systems that is highly modular, compact and made of parameterized modules.

First a discrete event model is presented and then a new formalism that merges the concepts of Hybrid Petri Nets and Colored Petri Nets is introduced: the Colored Modified Hybrid Petri Nets (CMHPNs). Hence a new CMHPN model is proposed: it allows to model both the event nature and the continuous nature of the system. As more, to allow the monitoring of system evolutions, a freeware simulation tool for the CMHPNs is presented.

Finally it is shown how the CMHPN model can be used to execute analysis and performance evaluation. Liveness analysis is performed by means of a hybrid automaton obtained from the net model. A deadlock prevention policy is synthesized working on an aggregated model. To prove the effectiveness of this new formalism an existing large automated warehouse system is presented as case study: its CMHPNs model is used to simulate the system behavior and to analyze the warehouse's performance.

In the identification environments, the guidelines of a new "active" approach to identify the model of a preexisting system is described. The proposed preliminary algorithm identifies a free labeled PN model on the basis of the observed output sequences and of the modifiable input consisting of the enabled controllable transitions set.

The main idea is to use the knowledge of the set of enabled controllable transitions together with additional information on the conflicting transitions to accelerate the net identification with respect to the passive identification approaches. In particular, the system assumes that the maximum time that must elapse from the enabling of a transition until it fires is known and that it is possible to detect if the system is entered in a cyclic behavior. Using this additional information, it is possible to determine a set of constraints to represent sequences that are not accepted by the system. Such constraints can be used to improve the net identification.

1.2 Thesis Overview

The thesis is organized as follow:

Chapter 2 contains a brief background on Petri Net (PN), Colored PN and Hybrid PN formalisms, necessary to understand the others chapters. Because of their importance in the developing of the contribute of this thesis, a brief literature review about the Hybrid PNs is also presented.

Chapter 3 introduces the problem of properly modeling a complex automated warehouse system and compares the proposed approach with the related works.

Chapter 4 describes two modular, compact, scalar approaches to model complex automated warehouse systems, the first based on the Colored Timed PN formalism and the second based on a new Petri net formalism that merges the concepts of Hybrid Petri Nets and Colored Petri Nets.

Chapter 5 presents a CMHPNs simulator that allows to design and simulate not only the net, but also the controller, allowing the user to create models ad hoc for several kinds of systems.

Chapter 6 deals with results obtained simulating the behavior of a real warehouse system.

Chapter 7 introduces the problem of the system identification. After a review of the existent literature, the guidelines of a new active algorithm are described.

Chapter 8	contains a summary table of the notations used in the dissertation.

Chapter 2

Notations and Definitions

In this Chapter the formalisms used in this dissertation are briefly recalled.

At first PNs are introduced. Then, a brief overview on CPNs is furnished. For further details on PNs and on simulation of Petri Nets (PNs), the reader can refer to [Mur89] and to [BCC07b].

Finally a background on the Hybrid Petri Nets (HPNs) is presented.

2.1 Petri Nets (PNs) and Colored Petri Nets (CPNs)

A *Place/Transition* (P/T) net is a 4-tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$, where P is a set of w places (represented by circles), T is a set of n transitions (represented by black bars), $\mathbf{Pre} : P \times T \mapsto \mathbb{N}$ ($\mathbf{Post} : P \times T \mapsto \mathbb{N}$) is the *pre* (*post-*) *incidence* matrix. $\boldsymbol{C} = \mathbf{Post} - \mathbf{Pre}$ is the incidence matrix. The net *marking* is represented as a vector $\boldsymbol{m} \in \mathbb{N}^m$. The marking of a place p is a scalar value $m_p \in \mathbb{N}$. A transition t is enabled at $\boldsymbol{m}$ iff $\boldsymbol{m} \geq \mathbf{Pre}(\cdot, t)$ and this is denoted as $\boldsymbol{m}[t\rangle$. An enabled transition t may fire yielding the marking $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C}(\cdot, t)$ and this is denoted as $\boldsymbol{m}[t\rangle\boldsymbol{m}'$. The symbols ${}^\bullet p$ (${}^\bullet t$) and $p^\bullet$ ($t^\bullet$) are used for the *pre-set* and *post-set* of a place $p \in P$ (transition $t \in T$), respectively, e.g. ${}^\bullet t = \big\{p \in P \mid \mathbf{Pre}(p, t) \neq 0\big\}$.

As shown in Fig. 2.1(a), there is a *structural* conflict when $^\bullet t_i \cap^\bullet t_j \neq \emptyset$. If t_i and t_j are both enabled, the conflict becomes a *behavioral* conflict.

Let $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ be a *Petri net system*, where $\mathcal{N}$ is a PN and $\boldsymbol{m}_0$ is its initial marking. Marking of $\mathcal{S}$ can be (partially) observable. In such a case, it can be divided in $\boldsymbol{m} = [\boldsymbol{m}_O, \boldsymbol{m}_{uO}]$, where $\boldsymbol{m}_O(\boldsymbol{m}_{uO})$ is the marking of observable (unobservable) places. We call $P_O(P_{uO})$ the set of observable (unobservable) places.

A *firing sequence* from $\boldsymbol{m}$ is a sequence of transitions $\sigma = t_1 \dots t_k$ such that $\boldsymbol{m}\big[t_1\rangle \boldsymbol{m}_1 \big[t_2\rangle \boldsymbol{m}_2 \dots \big[t_k\rangle \boldsymbol{m}_k$, and this is denoted as $\boldsymbol{m}[\sigma\rangle \boldsymbol{m}_k$. An enabled sequence σ is denoted as $\boldsymbol{m}\big[\sigma\rangle$, while $t_i \in \sigma$ denotes that the transition t_i belongs to the sequence σ. The function $\boldsymbol{\sigma} : T \to \mathbb{N}$, where $\boldsymbol{\sigma}(t)$ represents the number of occurrences of t in σ, is called *firing count vector* of the firing sequence σ. As it has been done for the marking of a net, the firing count vector is often denoted as a vector $\boldsymbol{\sigma} \in \mathbb{N}^n$. Note that, if a sequence is made by a single transition, i.e., $\sigma = t_i$, then the corresponding firing count vector is the i-th canonical basis vector denoted as $\boldsymbol{e}_i$.

A marking $\boldsymbol{m}'$ is said to be *reachable* from $\boldsymbol{m}_0$ iff there exists a sequence σ such that $\boldsymbol{m}_0[\sigma\rangle \boldsymbol{m}'$. $R(\mathcal{N}, \boldsymbol{m}_0)$ denotes the set of reachable markings of the net system $\langle N, \boldsymbol{m}_0 \rangle$.

A net system $\mathcal{S}$ is *bounded* if there exists a positive constant K such that $\boldsymbol{m}(p) \leq K, \ \forall \ \boldsymbol{m}(p) \in R(\mathcal{N}, \boldsymbol{m}_0)$.

A net system $\mathcal{S}$ is *live* if all its transitions are live. A transition t is live under the initial marking $\boldsymbol{m}_0$ if for every marking $\boldsymbol{m}$ reachable from $\boldsymbol{m}_0$, it exists a sequence σ, fireable from $\boldsymbol{m}$, which contains transition t. In other words, whatever the net evolution, a possibility always remains for firing t.

A PN system $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is said to be *reversible* if, for each marking $\boldsymbol{m} \in R(\mathcal{N}, \boldsymbol{m}_0)$, $\boldsymbol{m}_0$ is reachable from $\boldsymbol{m}$. Thus, in a reversible net one can always get back to the initial marking.

The *reachability graph* of a bounded net $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is a directed graph RG such that: i) the root node of RG is the initial marking of the net; ii) the other nodes of RG are associated to the reachable markings of $\mathcal{S}$; iii) an arc labeled t between two nodes X and Y of RG represents that the firing of transition t leads the net system from the marking associated to the X to the marking associated to Y

A net system $\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ is bounded, live and reversible iff its reachability graph is finite, strongly connected and each transition t labels at least one arc [DHP$^+$93].

All the formal definitions given for PNs can be naturally extended to Colored PNs (CPNs). Formally, a *CPN* is a 6-tuple $\mathcal{C} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, Cl, Co \rangle$. As in PNs, P is a set of m places (represented by circles), T is a set of n transitions (represented by bars). Cl is the set of colors. Co: $P \cup T \longrightarrow Cl$ is a color function that associates to each element in $P \cup T$ a non-empty ordered set of colors in the set of possible colors Cl. Forall $p \in P, Co(p_i) = \{a_{i,1}, a_{i,2}, ..., a_{i,u_i}\} \subseteq Cl$ is the ordered set of possible colors of tokens in p_i, and u_i is their number. Forall $t \in T, Co(t_j) = \{b_{j,1}, b_{j,2}, ..., b_{j,v_j}\} \subseteq Cl$ is the ordered set of possible occurrence colors in t_j, and v_j is their number. For each place $p_i \in P$, the marking m_i is defined as a non-negative multi-set over $Co(P_i)$. The mapping $m_i : Co(P_i) \to \mathbb{N}$ associates to each possible token color in P_i a non-negative integer representing the number of tokens of that color that is contained in the place p_i. The column vector of u_i non-negative integers, whose h-th component $m_{p_i}(h)$ is equal to the number of tokens of color $a_{i,h}$ that are contained in p_i, is denoted as $\mathbf{m}_{p_i}$. The marking of a CPN is an m-dimensional column vector of multisets: $\mathbf{m} = [\mathbf{m}_{p_1}...\mathbf{m}_{p_m}]^T$. For the sake of simplicity, a token of color "$c1$" contained in a place p_i will be indicated with the symbol (c_1).

In literature, more than one formal definition for CPNs exist, depending on how the incidence matrix and transition colors are defined. In the formalism chosen in this work, matrix entries are represented by matrices. **Pre** and **Post** are the pre-incidence and post-incidence $w \times n$-sized matrices, respectively. $\mathbf{Pre}(p_i, t_j)$ is a mapping from the set of occurrence colors of t_j to a non-negative multiset over the set of colors of p_i, namely, $\mathbf{Pre}(p_i, t_j) : Co(t_j) \to \mathbb{N}(Co(p_i))$, for $i = 1, ..., w$ and $j = 1, ..., n$. $\mathbf{Pre}(p_i, t_j)$ represents a matrix of $u_i \times v_j$ non-negative integers whose generic element $Pre(p_i, t_j)(h, k)$ is equal to the weight of

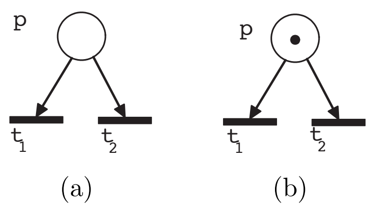


Figure 2.1 Conflict in Petri Nets: (a) structural conflict and (b) behavioral conflict.

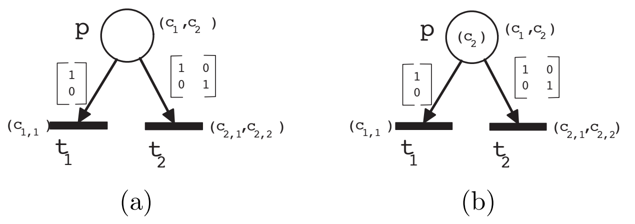


Figure 2.2 (a) Unmarked CPN ; (b) Marked CPN.

the arc from place p_i w.r.t color $a_{i,h}$ to transition t_j w.r.t color $b_{j,k}$. $\mathbf{Post}(p_i, t_j) : Co(t_j) \to \mathbb{N}(Co(p_i))$, for $i = 1, ..., m$ and $j = 1, ..., n$. $\mathbf{Post}(p_i, t_j)$ represents a matrix of $u_i \times v_j$ non-negative integers whose generic element $Post(p_i, t_j)(h, k)$ is equal to the weight of the arc from transition t_j w.r.t color $b_{j,k}$ to place p_i w.r.t color $a_{i,h}$. The incidence matrix C is a $m \times n$ matrix, whose generic element $C(p_i, t_j) : Co(t_j) \to \mathbb{Z}(Co(p_i))$, for $i = 1, ..., w$ and $j = 1, ..., n$, is the $u_i \times v_j$ matrix of integer numbers $C(p_i, t_j) = \mathbf{Post}(p_i, t_j) - \mathbf{Pre}(p_i, t_j)$. The concepts of *pre-set* and *post-set* of a place $p \in P$ or a transition $t \in T$ are naturally inherited from PNs, but colors must be also considered: ${}^\bullet t_i c_j = \big\{t_i c_j \in T \mid \mathbf{Pre}(p_h c_k, t_i c_j) \neq 0\big\}$.

In Fig. 2.2(a) a CPN with a structural conflict is shown. it is made up of a place p, having $C_o(p) = \{c_1, c_2\}$, and of two transitions, t_1 and t_2 with $C_o(t_1) = c_{1,1}$ and $C_o(t_2) = \{c_{2,1}, c_{2,2}\}$. When t_1 fires, one token, corresponding to color $c_{1,1}$, is removed from place p; t_2 can fire both under color $c_{2,1}$ and $c_{2,2}$ and when it fires, one $c_{2,1}$ or $c_{2,2}$ token, respectively, is removed from p. In Fig. 2.2(b) a token is added to the CPN in Fig. 2.2(a); notice that the conflict is still structural (not behavioral), since no $c_{1,1}$ token is present in p and, consequently, transition t_1 cannot fire.

When time is added to PNs and CPNs a time function is defined, which associates to each transition t_i in the case of PNs, or to each transition color $t_i c_j$ in the case of CPNs, a time duration from enabling to firing. In this case the PNs and CPNs become TPNs and CTPNs. Notice that timed and un-timed (also said immediate in the next) transitions will be represented with empty filled boxes and black bars, respectively.

2.2 Hybrid Petri Nets (HPNs)

A hybrid system is defined like a system consisting of a mixture of a continuous time system and a discrete event system (DES), having each one an own state space. These two systems are not independent but they influence each other. For the continuous time system, the influence of DES results in abrupt changes in the dynamic and can occur either as switches in the vector field or as jumps in the state. Reversely, the continuous evolution influences the DES one by generating events that affect the discrete states [PL95].

A continuous system can be described by differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{2.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ is the input vector and $\boldsymbol{y} \in \mathbb{R}^r$ is the output vector. In particular, if the interest is focused on the class of hybrid systems having autonomous commutations, i.e. systems for which changes in the dynamic occur if an analytical boundary condition about the instantaneous state value is reached, the equation

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.3}$$

with

$$\boldsymbol{f} = \begin{cases} \boldsymbol{f}_1(\boldsymbol{x}(t), \boldsymbol{u}(t)) & for \quad \boldsymbol{h}(\boldsymbol{x}(t)) \leq 0 \\ \boldsymbol{f}_2(\boldsymbol{x}(t), \boldsymbol{u}(t)) & for \quad \boldsymbol{h}(\boldsymbol{x}(t)) > 0 \end{cases} \tag{2.4}$$

can be used, where it has been supposed the system can switch only between two possible dynamics ($\boldsymbol{f}_1$ and $\boldsymbol{f}_2$) and $\boldsymbol{h}$ is the boundary condition.

For systems having linear, time-invariant, continuous part, like the ones treated in this thesis, each dynamic in (2.4) can be written as:

$$\boldsymbol{f}_i(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \boldsymbol{A}_i \cdot \boldsymbol{x}(t) + \boldsymbol{B}_i \cdot \boldsymbol{u}(t) \tag{2.5}$$

where $\boldsymbol{A}_i$ is a constant n-order square matrix and $\boldsymbol{B}_i$ is a (n×m)-order matrix.

To model hybrid systems behavior HPNs can be used [PL95, GU98, DA05, DPP09].

In more general hybrid systems, switching between different dynamics is caused not only by the boundary conditions but also by external input events, also called exogenous events. An exogenous event, as the term suggests, is an event originating from the outside world; by opposition, a change in internal state, as the occurrence of a boundary condition, can be called endogenous event or internal event. The external events can be "controllable" (i.e. their occurrence can be forced/disabled by an external agent, for example by a controller) or not controllable (i.e. their occurrence cannot be forced/disable by an external agent); an endogenous event is always not controllable. When changes in dynamic are ruled also by exogenous events, the HPNs used to model the system behavior are said *synchronized*, as those used in this dissertation: for these HPNs, an external event is associated with some transitions and the firing of these transitions occurs when the transition is enabled and the associated event occurs. Transitions whose firing is controlled by the occurrence of an external or internal event are called "synchronized". If the external event is a controllable event, then also transitions synchronized to such an event are called controllable, otherwise if a transition is synchronized to an uncontrollable event, then such a transition is said uncontrollable.

A HPN can be view as the combination of a "discrete" PN and a "continuous" PN.

Literature about HPNs is wide: a their complete presentation

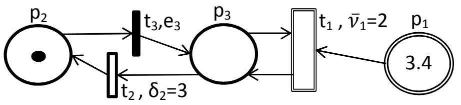


Figure 2.3 A basic HPN.

is given in [DA05]; in [PL95] it is shown how HPNs can be used to describe a general hybrid system having jumps in the state space and switches in its dynamic. Application of HPNs to oil refinery can be found in [WZC08, WCZ09, WCCZ10].

Several variants of HPNs have been proposed. Differential Petri Nets (DPNs) are introduced the first time in [DK98]; in these nets the marking of a differential place may be negative as well as the weights of arcs to or from a differential place. In [DA05], it has been shown how the behavior of DPNs can be obtained using HPNs whose transitions firing speeds is a function of the net marking, and for this reason they are called Modified HPNs (MHPNs) [DA05]. Then, it is not a limitation the use of no-negative markings and weights, as it is done in this dissertation.

To model systems having first-order continuous behavior, which can be studied by linear algebraic tools, Balduzzi et al. introduce the First-Order HPNs (FOHPNs) [BGM00] and use them to model manufacturing systems [BGS01]. In FOHPNs continuous transition firing speeds are constant values, chosen by a control agent in a fixed range. When an event occurs, the net state changes, and a controller can decide to vary speed values, while between two event occurrences the firing speeds remain constant. In this thesis firing speed values are not chosen in a fixed set but they are function of the marking of the net.

In formal way, a HPN is a 7-tuple $\mathcal{H} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, h, \boldsymbol{\delta}, \boldsymbol{\nu} \rangle$ such that: $P = P^D \bigcup P^C$, with $P^D \bigcap P^C = \emptyset$, where P^D (P^C) is the set of w_d discrete (w_c continuous) places, drawn like one (two) line circles; $T = T^D \bigcup T^C$, with $T^D \bigcap T^C = \emptyset$, where T^D is the set of n_d discrete transitions, which can be both immediate (drawn like black bars) and timed (drawn like white bars) and T^C is the set of n_c continuous transitions, drawn as a two lines boxes;

$\mathbf{Pre} : P \times T \rightarrow \mathbb{R}^+$ is the pre-incidence matrix; $\mathbf{Post} : P \times T \rightarrow \mathbb{R}^+$ is the post-incidence matrix; $h : P \bigcup T \rightarrow \{D, C\}$, called "hybrid function", indicates for every node whether it is a discrete node (sets P^D and T^D) or a continuous one (sets P^C and T^C); $\boldsymbol{\delta} : T^D \rightarrow (\mathbb{R}^+)^{n_d}$ is the firing delay vector, whose element δ_i is the firing delay associated to each discrete transition t_i^D: if $\delta_i = 0$ then the transition t_i^D is immediate, else if $\delta_i > 0$ then t_i^D is timed. Function $\boldsymbol{\nu} : T^c \rightarrow (\mathbb{R}^+)^{n_c}$ is the firing speed vector. Note that in case of discrete nodes, $\mathbf{Pre}$ and $\mathbf{Post}$ assume integer positive values. The incidence matrix of the net is defined as $\boldsymbol{C} = \mathbf{Post} - \mathbf{Pre}$ and it can be written as the block matrix:

$$\boldsymbol{C} = \left(\begin{array}{c|c} \boldsymbol{C}_{CC} & \boldsymbol{C}_{CD} \\ \hline \boldsymbol{C}_{DC} & \boldsymbol{C}_{DD} \end{array} \right) \tag{2.6}$$

where $\boldsymbol{C}_{CC}$ is the block regarding connections between continuous nodes, $\boldsymbol{C}_{DD}$ is the block regarding connections between discrete nodes, $\boldsymbol{C}_{CD}$ is the block regarding connections between continuous places and discrete transitions and $\boldsymbol{C}_{DC}$ is the block regarding connections between discrete places and continuous transitions.

HPN marking is a function $\boldsymbol{m} = \{\boldsymbol{m}^C, \boldsymbol{m}^D\}$, with $\boldsymbol{m}^C : P^C \rightarrow \mathbb{R}^+$, $\boldsymbol{m}^D : P^D \rightarrow \mathbb{N}$, that assigns to each continuous place a real number and to each discrete place a nonnegative integer number of tokens (graphically represented as black dots in the discrete places). The notation $\boldsymbol{m}(\tau_k)$ is used to denote the value of the marking of the net at the instant τ_k. The marking of a place p at a time τ_k is denoted by $m_p(\tau_k)$. The symbols $^\bullet p$ ($^\bullet t$) and $p^\bullet$ ($t^\bullet$) are used for the *preset* and *postset* of a place $p \in P$ (transition $t \in T$), respectively, e.g. $^\bullet t = \{p \in P \mid \mathbf{Pre}(p, t) > 0\}$.

A discrete transition t^D is enabled at time τ_k if $m_p(\tau_k) \geq \mathbf{Pre}(p, t^D)$, $\forall p \in {}^\bullet t^D$. A transition t^D can be either autonomous or synchronized to a logical expression, function of an external control input g and/or of an internal condition e. Both g and e are boolean functions $g, e : T^D \rightarrow \{0, 1\}$. The former becomes true, so generating an exogenous event, when a controller sets to true the external event it is associated to; the latter becomes true, so generating an endogenous event, when the internal event it is associated to is verified. A discrete transition t^D can fire if it is enabled and the associated logical expression becomes true, i.e. both the endogenous and the exogenous events its firing is synchronized to occur. As for example, in a system formed by two masses traveling along a guidepath, an internal condition can be associated to the reaching of a threshold distance that makes masses decelerate; an external control input for the same system is an asynchronous stop command arriving from an external controller; a logical expression can be the logic function AND between g and e, e.g. $g \wedge e$.

A continuous transition $t^C \in T^C$ is enabled at time τ_k if i) $m^D_{p^D}(\tau_k) \geq \mathbf{Pre}(p^D, t^C)$, $\forall p^D \in {}^{\bullet}t^C$ and ii) $m^C_{p^C}(\tau_k) \geq 0 \, \forall p^C \in {}^{\bullet}t^C$. To each continuous transition t^C_i is associated the instantaneous firing speed (in the following also called simply firing speed) ν_i: if t^C_i is disabled $\nu_i = 0$; when t^C_i is enabled ν_i is equal to the maximal firing speed $\bar{\nu}_i$, indicated near the transition. The firing of continuous transitions cannot change the marking of discrete places, consequently $\boldsymbol{C}_{DC}(p^D, t^C) = 0$, $\forall p^D \in P^D$, thus $\boldsymbol{C}_{DC} = \mathbf{0}$. The time derivative of the marking of a continuous place p^C, $\frac{d\boldsymbol{m}_{p^C}}{dt}$, is called *balance* and it is defined as: $\dot{\boldsymbol{m}}_{p^C} = I - O$ where $I = \sum_{t^C_j \in {}^{\bullet}p^C} Post(p^C, t^C_j)\nu_j$ is the *feeding speed* of the place p^C, while $O = \sum_{t^C_k \in p^{C\bullet}} Pre(p^C, t^C_k)\nu_k$ is the p^C *draining speed.* The evolution of the net can be described by its fundamental equation (written in a way pointing out the continuous part and the discrete part):

$$\begin{bmatrix} \boldsymbol{m}^C(\tau_k) \\ \boldsymbol{m}^D(\tau_k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{m}^C(\tau_{k-1}) \\ \boldsymbol{m}^D(\tau_{k-1}) \end{bmatrix} + \\ + \begin{bmatrix} \boldsymbol{C}_{CC} & \boldsymbol{C}_{CD} \\ \mathbf{0} & \boldsymbol{C}_{DD} \end{bmatrix} \left(\begin{bmatrix} \mathbf{0} \\ \boldsymbol{\sigma}(\tau_k) - \boldsymbol{\sigma}(\tau_{k-1}) \end{bmatrix} + \int_{\tau_{k-1}}^{\tau_k} \begin{bmatrix} \boldsymbol{\nu} \\ \mathbf{0} \end{bmatrix} \right) \tag{2.7}$$

where $\boldsymbol{\sigma}(\tau_k) : T^D \to \mathbb{N}^{n_d}$ is the discrete firing vector whose component $\sigma_{t^D_i}(\tau_k)$ represents the number of times the discrete transition t^D_i is fired up to the current time τ_k.

For the sake of clarity, from now on the term "synchronized transition" will not be used any more and synchronized transitions will be called just controllable or uncontrollable.

A basic HPN is shown in Fig. 2.3, having:

- $P^C = \{p_1\}$, $P^D = \{p_2, p_3\}$;
- $T^C = \{t_1\}$, $T^D = \{t_2, t_3\}$ where t_3 is an immediate uncontrollable discrete transition, with associated the internal condition e_3 and t_2 is a discrete timed transition;
- $\delta = \{\delta_2\}$;
- $\boldsymbol{C} = \left(\begin{array}{r|rr} -1 & 0 & 0 \\ \hline 0 & 1 & -1 \\ 0 & -1 & 1 \end{array}\right)$.

For basic HPNs, the maximal firing speed of continuous transitions is a constant value, but powerful modifications have been proposed where continuous transition maximal firing speed is a function of the input places marking, of the input vector and of the time:

$$\nu_t(\tau) = f(\boldsymbol{m}(\tau), \boldsymbol{u}(\tau), \tau) \tag{2.8}$$

These kind of HPNs are called Modified HPNs (MHPNs).

Chapter 3

Issues about automated warehouse system modeling and control: a literature review

Since 1990 a big effort has been spent to find optimal strategies for planning and control of warehouse systems. Planning involves long-term optimization: it usually has a day or week time horizon and it is based on simplified models of warehouse systems and on statistical characterization of the system performance [Van99]. Conversely, a detailed model is used for the control which performs the short-term optimization of handling sequences, that usually has the objective to minimize the time to complete a little number of picking or storage operations and it is based on the current state of the system. A general warehouse architecture consists of a number of aisles, each one served by a crane, an Interface System (IS) and picking positions (see Fig. 3.1).

On both sides of each aisle there is a storage rack composed of n_r rows and n_k columns; moreover, as it has been said, each aisle is served by a crane, capable of moving both vertically and horizontally at the same time, which performs the following operations: i) picking of the Stock Unit (SU) at the input buffer/bay

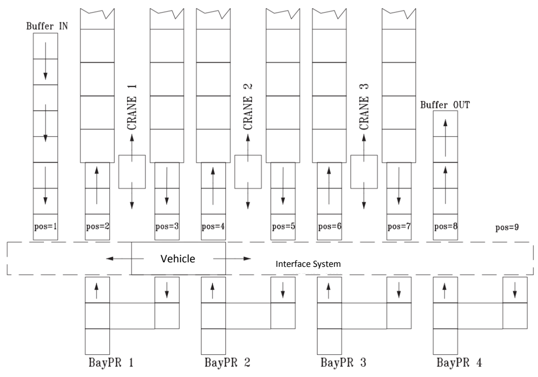


Figure 3.1 Scheme of a general warehouse architecture

of the aisle to be stored in a rack location S; ii) storage of the SU into the assigned location S of the rack; iii) movement to location R where a retrieval has been requested; iv) retrieval of the SU stored in R; v) movement to the output buffer/bay of the aisle to deposit the SU. This set of operations is called, in the warehouse system context, a Dual Command (DC) machine cycle [GHS77], [BW84], [HMSW87], [LdSO96]. The DC cycle can be generalized to the case of multiple storages and retrievals for cycle.

The IS consists of vehicles which can move a number of SUs. The vehicles move along a mono-dimensional guidepath placed orthogonally with respect to the aisle axis. They perform picking actions (from the aisles output bays and from the picking area output bays) and deposit actions (into the aisles input bays and into the picking area input bays).

The picking area represents the output point of warehouse systems. A picking bay consists of a picking location connected via conveyors to the IS input and output interfaces, so that a SU can be partially emptied by a human operator and then carried back to an aisle rack location.

An input buffer represents the interface of the warehouse with the incoming area. It is used to load full SUs in the warehouse.

A set of missions is given as input to this kind of systems. Each mission requires that a certain quantity of an item, which can be stored in more than one aisle, is moved to a picking bay. Hence, the execution of a mission requires the choice of the SU to move among those containing the desired item (this choice includes also the choice of the crane since there is one crane in each aisle), the choice of a vehicle to transfer the SU to the picking area, the choice of the picking bay, again the choice of a vehicle to return the SU in the storage area and the choice of the location where the SU must be stored among those available.

The control problem consists in assigning each available resource (a location, a picking bay, a crane or a vehicle) to a mission. When one resource is available for a set of missions, a conflict occurs. The output of the control problem consists in determining Who has to do What and in Which Order in a manner that a certain objective is reached over a certain time horizon. In other words, the control must solve these conflicts. A detailed model is needed since it is important to detect in which order these conflicts occur.

This thesis focuses on how to obtain a model oriented to the control and to the performance analysis of these systems. In particular, the complexity of modern warehouse systems, like the real one considered in Chapter 5, requires big interfaces (e.g. a carousel, shuttles, rail guided vehicles) between cranes and picking area and so many vehicles must be used. Furthermore, when each crane cycle involves more than one picking and deposit, the number of SUs moved by vehicles at a time in the interface area grows, and then a significant time is required to cover the interface guidepath.

Discrete event systems have been proposed to obtain such a detailed model in [ABCC05] and in [DF05].

The challenging problem is the control of the IS since the control of the cranes has been studied a lot in the literature. In a certain sense, the control of the whole warehouse reduces to that of the cranes if the time to cover the IS is negligible. In [ABCC05] it is shown that a key point in the development of the warehouse optimization is that the crane optimization can be considered independent of the vehicle optimization if a vehicle requires a negligible time with respect to the crane mean cycle time to reach the crane bay from a picking bay. Thus, as soon as a cycle ends, a crane can start another cycle.

Once a crane cycle has been created (i.e. the list of locations to visit in a single travel to store and to pick SUs), the cycle time is deterministic and it can be analytically computed. In this dissertation it is assumed that the cranes work according to an extended version of the algorithm presented by the authors in [ABCC05] to optimize DC cycles, but the effectiveness of the approach here presented is independent of the crane algorithm.

Note that cranes have not a discrete event behavior. The discrete event behavior of an automated warehouse is caused by the IS. The activity of the IS is more relevant when the number of vehicles and the number of interface bays grows, and consequently the stop and go state of the vehicles related to event occurrences grows (e.g. a collision of two vehicles must be avoided, a vehicle stops when it reaches a certain interface bay, etc.). This increases the time to move a SU from the picking area to the aisle input bays and reduces the crane performances independently of the crane optimization algorithm.

Moreover, when the size of the IS grows also its continuous time phenomena cannot be neglected. Indeed, a more precise information about vehicles position becomes relevant. Using Petri Nets (PNs) [Mur89] a guidepath is represented by a number of places. These places model the presence of a vehicle in a certain zone of the IS. The exact position in this zone is unknown. A better precision requires many places. On the other hand, a continuous time system allows to represent the exact position as well as the mode changing in dynamics of vehicles (acceleration, deceleration or constant velocity).

In this dissertation a particular warehouse layout is considered, presented in Fig. 3.2, where the interface system is made up of a circular path where vehicles continually turns transporting SUs. Indeed, during the researches made to write this thesis, it has been pointed out how such a layout is very common in several real warehouses (as the one presented in the case study).

For such automated warehouses together with the IS, a detailed

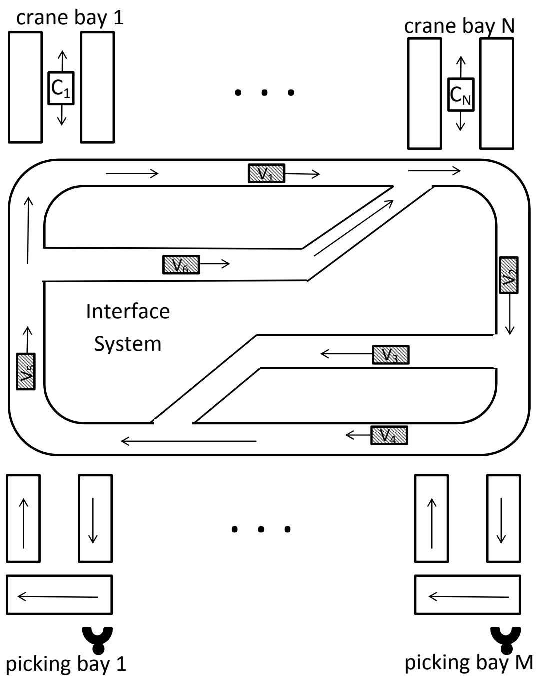


Figure 3.2 Layout of a general real warehouse architecture: at the top aisles and crane bays (blocks $C_1 \dots C_N$ represent the cranes serving the respective aisles), at the bottom picking bays, in the center Interface System routes with running vehicles (blocks V_i, with $i = 1 \dots 6$ represent the vehicles).

formal model is presented. Two different approaches have been used: first a discrete event model based on the standard Colored Timed Petri Net (CTPN) [Jen95, HHC98] formalism is proposed, and then the same is obtained using Hybrid Petri Nets (HPNs) [DA01]. Difference between the two approaches is that in the first case both picking/crane bays and the IS have a discrete event dynamic and for this reason they are modeled as DESs; in the second case instead the IS is modeled as a continuous system.

The both models are highly modular, compact and made of parameterized modules: the reuse of model components to model different warehouse systems is very easy. In the both cases the overall system model is obtained by composing elementary modules according to the constraints [TTV06] represented by vehicles route. The interface with a higher level scheduler (dispatcher) is embodied in the obtained model, thus allowing off-line performance evaluation of state-dependent dispatching control algorithms, as well as on-line implementation of complex control algorithms which are based on look-ahead (or what-if) techniques [HC99].

As more, the models of warehouse systems are used to obtain a deadlock prevention policy and to evaluate the systems performance via an experimental campaign based on simulations of a real case study in a very efficient way.

3.1 Warehouse Systems

A lot of results are available for unit-load automated storage systems, while few results are available for other storage systems [RRS+00, GGM10]. Moreover, there is an enormous gap between the published warehouse research results and the practice of design and operations. As raised in [GGM10], a challenge for the research in this field is the integration of optimization, simulation and modeling of the full warehouse systems, not only automated storage and retrieval systems. One of the contributions of this dissertation is to show the importance of an accurate modeling and simulation of these systems. The optimization is not the topic of this thesis, but the experimental results presented in Chapter 5 show that it is greatly influenced by the model accuracy.

In [DF05] a unique model for automated storage and retrieval systems, comprising rail guided vehicles and narrow aisle cranes, is proposed. In the approach presented in this dissertation, differently from [DF05] the activity of automated storage and retrieval subsystems reduces to a timed transition modeling the time to perform a given cycle: the aisles and their locations are not explicitly modeled. This allows one to obtain a model of a reasonable size for real warehouses like the one considered in the following chapters. Model oriented to the control of an AS/RS system is presented in [XWW$^+$07]. Using P/T PN, the authors first obtain a model of the system layout, than they simplify it to have a model more convenient for analysis and simulation. Finally, colors and temporized transitions are introduced to solve merging traffic problem of goods on conveyors. Colors are used to indicate the SUs destinations while time properly model temporized actions, as the moving of a SU from a conveyor to another one, in this way a FIFO rule for the passage of SUs on the conveyors is implemented. Also in this thesis a simpler model of the system, called *aggregate model* (see Chapter 4, Section 4.2.14), is used to detect the possible presence of deadlocks, but differently from [XWW$^+$07] deadlock prevention is carried out by means of an opportune resource allocation policy. In [HCL07] a modular CTPN model of an automated warehouse is presented. Each module describes a different action in the system (i.e. loading/unloading of a pallet on a crane, entering of a pallet in the warehouse...) and they can be composed to describe the warehouse overall behavior. Inhibitor arcs are used and a new kind of node is introduced, called virtual place, to accommodate instructions from schedulers (i.e. fixing destination of a pallet, what shuttle reserving for move the pallet...). The model is than implemented in a C++ code to simulate the system behavior. In the approach presented in this dissertation, neither inhibitor arcs nor virtual places are used. TPNs are used in [XH11] to model a logistic warehouse on the base of its operation procedure: the result is a modular model that can be used to simulate the flow of the operations in the system. The model presented in this thesis model both the logical operation procedure and the physical layout of the system.

Using commercial simulation tools like Arena© or Automod© is possible to model systems like the one presented in this thesis. However, they are not based on a formal model, and so, they are not enough general to be applied to every ISs. Moreover, they can be used only to simulation purposes, while a formal model allows to check system properties (e.g. deadlock avoidance) by formal analysis.

Among formal modeling methodologies, an appealing approach is the matrix-based framework proposed by [TL97]. Application to warehouse systems can be found in [GZNL08] where a variable dispatching rule control approach is used for operational control issues. In the same paper it is shown that the matrix-based model can be included into a multi-level control architecture where the model is used to determine when a control decision has to be taken from upper levels, and to feature operational control tasks too. Simulation is used to tune, off-line, some parameters of the control law. In this thesis, a similar approach is followed in a discrete as well as in a hybrid system context.

3.2 Deadlocks

Three basic approaches have been developed (see [FZ04] and the reference therein) for the deadlock resolution problem in automated manufacturing approach, "deadlock detection and recovery", "deadlock avoidance" and "deadlock prevention". In the context of Interface Systems (ISs) "deadlock prevention" or "deadlock avoidance" approaches must be used. Indeed, these systems have a very hight throughput and a deadlock recovery requires a too high price in terms of time and cost. The first one consists either to design a system such that deadlocks will never occur or to add a control mechanism on resource requests which prevents deadlocks to occur [HZL12]. The second one consists in using a real time deadlock controller to rule the resource assignment applying look–ahead strategies.

An IS can be considered a simple guidepath-based traffic system [RR08], like Automated Guided Vehicle (AGV) or Rail Guided Vehicle (RGV) systems, since it consists of a number of vehicles that travel among a number of locations, following some predetermined paths. However, links of this guidepath network are unidirectional, while in general case they are bi-directional; the motion of the vehicles on these links is unidirectional; vehicles can travel by following pre-specified routes in the guidepath network, while in general case they develop their route in real-time, based on the prevailing congestion conditions in the network; idle vehicles remain in the guidepath network during their idling period moving among various links, while in general case they are moved only to clear the way for some other vehicles or they will retire to a particular location of the guidepath network known as the system docking station. In [RR08] the problem of enforcing liveness for guidepath-based traffic systems is addressed in a discrete event context.

In [DF07] a deadlock avoidance policy is presented for a RGV system used to load/unload the automated warehouse where the picking area bay has infinite capacity, cranes have capacity one, and idle vehicles remain on the guidepath link until they receive a new mission.

The wrong management of the vehicles in ISs, when they remain in the guidepath network during their idling period moving among various links, can afford a particular deadlock condition, called "livelock": due to the bad vehicle assignment, picking and crane bays can become completely full. As consequence no more exchanges with the IS can occur. While in this condition the crane system is blocked (busy cranes are unable to unload the SUs on the bays and for this reason they are halted at the interface points), the IS is not: busy vehicles continue to run along the path, waiting for a free position on their bay destination. The system is not physically blocked, but no missions can be completed, energy is lost, and vehicles continue to run.

The problem of livelocks in the AGV systems in a discrete-event context with bidirectional guidepath network, unidirectional vehicles, zone control for avoiding collisions, and dynamic route planning is the topic of the preliminary paper [Ros02], and it has been generalized in [RR08]. As it has been explained before, ISs are simpler than AGV systems: there are not bidirectional paths and they do not have bridges between routes but (in case) only branches. Indeed, ISs are used to decouple the warehouse and the picking area to empty SUs as soon as possible, then the layout is chosen as simple as possible to make transport operations fast.

As for the blocking properties, another goal of this dissertation is to show that a livelock can occur in ISs. Moreover, it is proposed to achieve livelock-freeness enforcing policy by enforcing deadlock-freeness on a discrete event model properly obtained from the hybrid one.

Chapter 4

Warehouse system models

In this chapter two modular, compact, scalar approaches to model complex automated warehouse systems are presented.

First a discrete event model based on the standard Colored Timed Petri Net (CTPN) [Jen95, HHC98] formalism is proposed and then a hybrid model based on a new Petri net formalism that merges the concepts of Hybrid Petri Nets and Colored Petri Nets then is discussed. Both the models of the warehouse can be used for the performance evaluation as well as for online implementation of control algorithms.

4.1 Colored Timed Petri Net Model

In the next it will be show how the IS can be considered made up of several interacting modules, each one with own characteristics. To properly model each module, the concept of CTPN block is introduced:

Definition 4.1.1. A *CTPN block* is a tuple $B = (\mathcal{C}, T_{in}, T_{out})$ where $\mathcal{C}$ is a CTPN, $T_{in} \subset T$ is the set of the input transitions, $T_{out} \subset T$ is the set of output transitions and $T_{in} \cap T_{out} = \emptyset$. $\mathbf{Pre}(p,t)$ and $\mathbf{Post}(p,t)$ matrices are diagonal matrices $\forall p \in P$ and $\forall t \in T$, i.e. firing of a transition t w.r.t. color c_h only consumes c_h-color tokens from ${}^\bullet t$.

Input and output transitions allow connecting CTPN blocks together by means of dummy places. When M blocks are connected together a new CTNP block $B' = (\mathcal{C}', T'_{in}, T'_{out})$ is obtained where $\mathcal{C}' = (P', T', \mathbf{Pre}', \mathbf{Post}', Cl', Co')$ with $P' = \left(\bigcup_{i=1}^{M} P_i\right) \bigcup D$, where P_i=block i encapsulated net place set and D=set of dummy places needed to link blocks together; $T' = \bigcup_{i=1}^{M} T_i$, where T_i= block i encapsulated net transition set; $\mathbf{Pre}'$ and $\mathbf{Post}'$ are the new pre and post incidence matrices; $Cl' = \bigcup_{i=1}^{M} Cl_i$, where Cl_i is the block i encapsuled net color set; $T'_{in} = \left\{t : t \in T_{in,i}, {}^{\bullet}t = \emptyset\right\}$, with $i = 1 \dots M$ and $T_{in,i}$=block i input transition set; $T'_{out} = \left\{t : t \in T_{out,i}, t^{\bullet} = \emptyset\right\}$, with $i = 1 \dots M$ and $T_{out,i}$=block i output transition set.

Ways to link blocks together are:

- "$1 \to 1$" or *sequence*: given two blocks $B_1 = (\mathcal{C}_1, T_{in,1}, T_{out,1})$ and $B_2 = (\mathcal{C}_2, T_{in,2}, T_{out,2})$ they are connected in $1 \to 1$ when a dummy place d is added such that $d \in {}^{\bullet}t_{in}$ and $d \in t^{\bullet}_{out}$, with $t_{in} \in T_{in,2}$ and $t_{out} \in T_{out,1}$, $Co(d) = Co(t_{out})$, $Co(t_{in}) \subseteq Co(t_{out})$.

- "$N \to M$" or *hub*: given N blocks $B_j = (\mathcal{C}_j, T_{in,j}, T_{out,j})$, with $j = 1 \dots N$, they are connected in $N \to M$ way to the blocks $B_i = (\mathcal{C}_i, T_{in,i}, T_{out,i})$, with $i = 1 \dots M$, when a dummy place d is added such that $d \in {}^{\bullet}t_{in,i}$ and $d \in t^{\bullet}_{out,j}$, with $t_{in,i} \in T_{in,i}$ and $t_{out,j} \in T_{out,j}$, $Co(d) = \bigcup_{j=1}^{N} Co(t_{out,j})$. If $\exists (h,k) : h \neq k,\ Co(t_{in,h}) \cap Co(t_{in,k}) \neq \emptyset$ with $t_{in,h} \in T_{in,h}$, $t_{in,k} \in T_{in,k}$ then a structural conflict between $t_{in,h}$ and $t_{in,k}$ occurs. When $N = 1, M > 1$ hub connection is called *branch*; if $M = 1, N > 1$ hub connection is called *confluence*.

For each connection presented above the following conditions have to be respected:

(i) $\bigcup_{i=1}^{N} Co(t_{in,i}) \subseteq \bigcup_{j=1}^{N} Co(t_{out,j})$;

(ii) $\mathbf{Pre}(d,t) = 0\ \forall t \notin T_{in,i}$ and $\mathbf{Post}(d,t) = 0\ \forall t \notin T_{out,j}$;

(iii) $\mathbf{Pre}(d,t)$ ($\mathbf{Post}(d,t)$) is a diagonal matrix;

(iv) $|^\bullet t_{in,i}| = 1$ and $|t^\bullet_{out,i}| = 1$.

Proposition 4.1.1. *Connecting two or more CTPN blocks together, no synchronization is introduced.*

Proof. A synchronization may occur if $|^\bullet t| > 1$ or **Pre**(p,t) is not a diagonal matrix (two tokes of different colors in the same input place are required to enable an output transition under a certain color).

Because of condition (iv), the first case (i.e. $|^\bullet t| > 1$) cannot occur.

When connection of M CTPN blocks is performed the resulting pre-incidence matrix is

$$\mathbf{Pre}' = \begin{bmatrix} \mathbf{Pre}_1 & \dots & \mathbf{Pre}_i & \dots & \mathbf{Pre}_M \\ \mathbf{Pre}(D,T_1) & \dots & \mathbf{Pre}(D,T_i) & \dots & \mathbf{Pre}(D,T_M) \end{bmatrix}$$

where $\mathbf{Pre}_i$ is the block i encapsuled net pre-incidence matrix, $\mathbf{Pre}(D,T_i)$ is the weights matrix of the arcs connecting dummy places with block i encapsuled net transitions. Since $\mathbf{Pre}_i(p,t)$ is a diagonal matrix $\forall p \in P_i,\ t \in T_i$ and since $\mathbf{Pre}(d,t)$ is a diagonal matrix $\forall d \in D,\ t \in T_i$, $\mathbf{Pre}'(p,t)$ is still a diagonal matrix $\forall p \in P', t \in T'$. $\square$

4.1.1 CTPN Model of the IS

A IS consists of a unidirectional path along which vehicles move. This path, without loss of generality, can be divided in these basic components:

1. *Elementary zone*, where a unique route is possible and there are not interfaces with other subsystems.

2. *Switching zone*, where more than one route is possible since the path can branch off in different lines of travel.

3. *Interface zone*, where a vehicle stops to load (unload) a SU from (to) another subsystem bay.

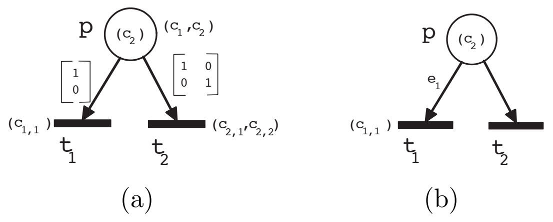


Figure 4.1 (a) Marked CPN as seen in Chapter 2; (b) Marked CPN in (a) with simplified notation.

In this section it is shown how these components can be modeled each one by a single CTPN block, so the whole IS can be modeled by a CTPN obtained properly connecting a set of CTPN blocks. This formalism allows an easy conflicts detection and then it allows an easy implementation of the controllers based on dispatching rules. The output of dispatching rules actions is to disable all the events except one. Although such controllers produce only a locally optimal solution to the conflicts problem, they are simple to implement and they face the possible combinatorial explosion of the control of the mission-to-resource-assignment problem.

For the sake of simplicity, with reference to Chapter 2, some special notations are used to draw CTPNs:

- When transition occurrence colors can fire under any input place color, no colors are indicated at transition side [see Fig. 4.1(b)] for transition t_2). If a transition occurrence color can fire only under some specific input place colors, they will be indicated near the transition [see Fig. 4.1(b) for transition t_1].

- If no matrix is indicated near an arc, an identity matrix I is intended to be associated to that arc [see Fig. 4.1(b); this is the case of the arc from place p to transition t_2].

- e_h near an arc from place p_i to transition t_j or viceversa denotes a column vector of size u_i, with the h-th element equal to one and the other elements equal to zero [see Fig. 4.1(b); this is the case of the arc from place p to transition t_1].

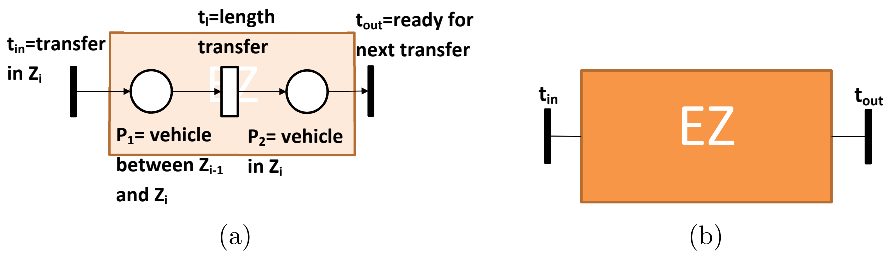

t_{in}=transfer in Z_i
t_l=length transfer
t_{out}=ready for next transfer
P_1= vehicle between Z_{i-1} and Z_i
P_2= vehicle in Z_i
(a)
t_{in}
EZ
t_{out}
(b)

Figure 4.2 Elementary zones module: (a) CTPN; (b) block representation.

Colors can be used to discriminate a free vehicle from a busy one, to identify where busy vehicles are directed and to identify a SU. SU identification is not essential in the IS since SUs are moved according to their destination in the IS path, but it is used to associate the crane location once a SU reaches a crane. For this reason, each net place has as many colors as the possible vehicles destinations plus one color that identifies an empty vehicle, but a SU number is associated to each token.

In real automated warehouses each zone can be occupied by just one vehicle at time: a vehicle has to halt while the next zone is busy.

Firing time of timed transitions depends on the length of the zone and represents the time necessary to the vehicle to cross the zone. For the sake of simplicity, in this first approach acceleration and deceleration have been neglected. Hence, vehicles can have only two speeds, $v = 0$ when they are stopped and $v = V_{max}$ when they are moving. With these simplifications, a IS Elementary Zone (EZ) can be modeled like a "simple belt conveyor". This technique simplifies also the modelling of interface zones since crane or picking bays consist of belt conveyors.

In Fig. 4.2 the block representation of an EZ is shown: in Fig. 4.2(a) the CTPN model is visible, while in Fig. 4.2(b) the black-box version of the same is reported.

Switching Zones (SZs) can be modeled as shown in Fig. 4.3. Notice that a behavioral conflict occurs every time a token is present in the place P_2 of Fig. 4.3(a), while a confluence is present at the beginning of Fig. 4.3(c).

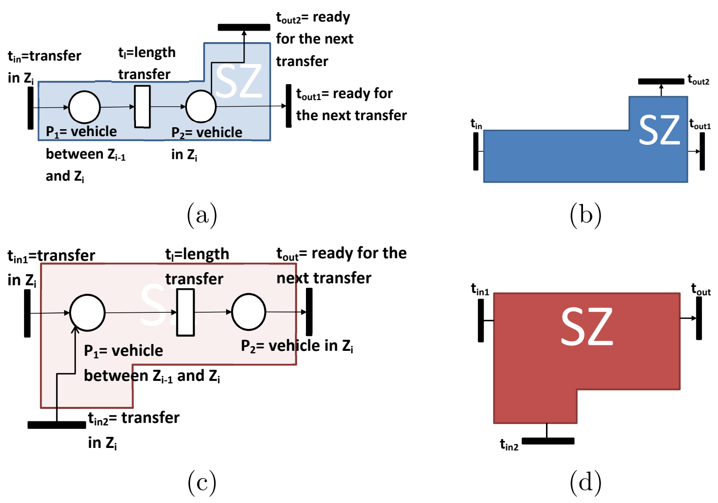


Figure 4.3 Switching zone: (a-b) output branch point and (c-d) input branch point .

In EZs and SZs the occurrence color of the firing output transitions is the same of the firing input transitions, i.e. no transformation color occur in the module.

Interface Zones (IZs) between vehicles and crane (or picking) bays can be modeled as shown in Fig. 4.4. Dotted places M_e and M_f are not part of the modules but they are *resource places* needed to model the conditions for enabling the SU exchange between picking bays and vehicles. As example, in Fig. 4.4(a) [Fig. 4.4(b)] exchange can occur only when transition t_{in2} (t_{out2}) is enabled, i.e. when a $c_e(c_h)$-color token is in resource M_e (M_f), that model the presence of a free (busy) vehicle in the IZ. When t_{in2} (t_{out2}) fires, a $c_h(c_e)$-color token is added in M_e (M_f), modeling the new state of the vehicle. After the adding of the new token, transition t_{out1} can fire and the token is passed in the next zone.

Unlike EZs and SZs, occurrence color of the IZs output transition t_{out} (t_{out1} for the IZ modeling passage from IS to the bay) can be different from the occurrence color of the input transition t_{in1}. This occur as consequence of a change in the state of the vehicle (from busy to free and viceversa), when an exchange with the bay occurs.

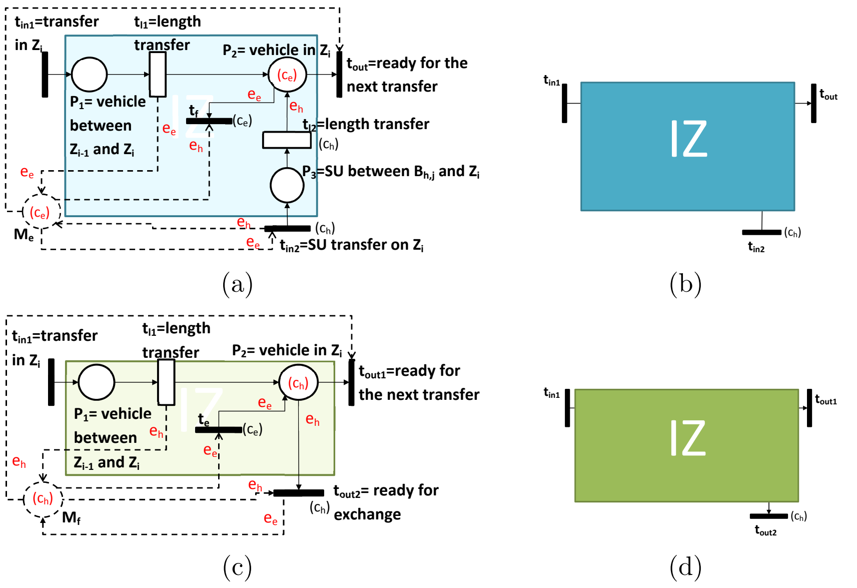


Figure 4.4 Interface zone: (a-b) passage of SUs from bay to vehicle and (c-d) viceversa .

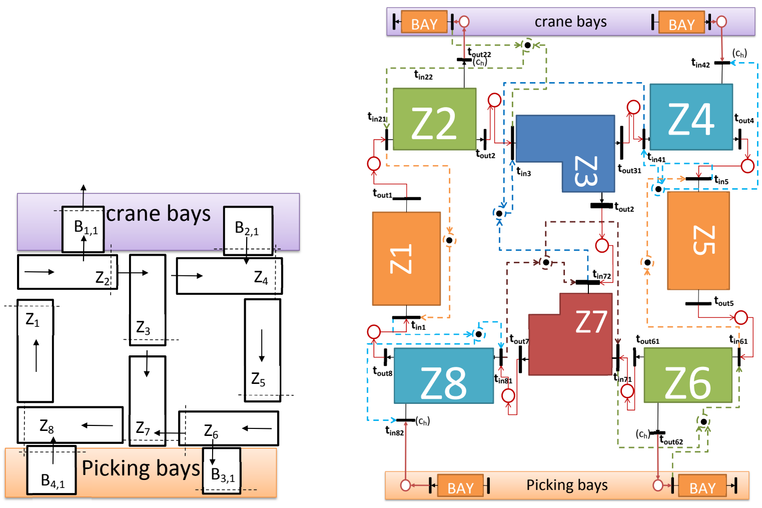
In the modules introduced above there are not synchronization, but only choices and confluences appear in SZs and IZs. As more, if only the skeleton of the nets are considered, without considering resources, it can be noted that the nets modeling each modules are State Machines. In the following the state machine definition and the liveness property are recalled.

Definition 4.1.2 (see [Mur89])**.** A State Machine (SM) is an ordinary net such that each transition t has exactly one input place and one output place, i.e.,

$$\forall t \in T, \sum_{p \in P} \mathbf{Pre}(p,t) = \sum_{p \in P} \mathbf{Post}(p,t) = 1.$$

Theorem 4.1.2 ([Mur89])**.** *A state machine* $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$ *is live iif it is strongly connected and it has at least one token in its initial marking.*

Modularity is an advantage of the CTPN model presented: it can be adapted at several layouts just adding or removing EZs; as



(a) (b)

Figure 4.5 A possible IS layout; (a) physical layout and (b) its model.

more, if a new bay or a new route is added, it can be connected at the IS just introducing the CTPN module modeling an IZ or a SZ respectively.

As example, in Fig. 4.5(a) a possible layout is reported. Notice how it can be modeled in a very simple way, properly connecting 2 EZs, 2 IZs and 2 SZs, forming a closed path [Fig. 4.5(b)].

4.2 Colored Modified Hybrid Petri Net Model

The previous CTPN model has been obtained looking to the warehouse system as if it is made up of only discrete event subsystem. However, when the spatial extension of this kind of systems grows, their continuous time behaviors cannot be neglected. Indeed, a more precise information about the position/state of the vehicles becomes relevant. As for example, using discrete event system formalism like PNs, a path is represented by a number of places. Such places model the presence of a vehicle in a certain zone. The

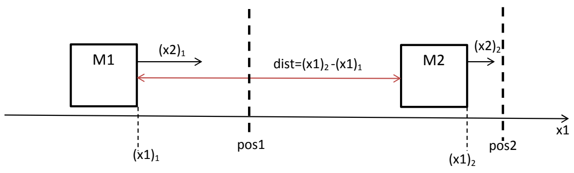


Figure 4.6 Mass system used in the example of section 4.2.2.

exact position in the zone is unknown. A better precision requires many places. On the other hand, a continuous time system allows to represent the exact position but the mode changing in dynamic of vehicles (acceleration, deceleration or constant velocity) as well as the stop and go state of the vehicles (e.g. a vehicle stops when it reaches a certain position) would not be easily modeled. Then a possible solution is to use a hybrid model: this allows one to model picking and crane bays still as discrete event systems (in particular like a sequence of belt conveyors) and, contemporaneously, to model the IS as a continuous system.

Before to present the hybrid model the Modified Colored Hybrid Petri Net are introduced. An example is discussed in detail to motivate the introduction of the new formalism.

4.2.1 Colored Modified Hybrid Petri Nets

In this dissertation MHPNs where (2.8) is a linear function of $\boldsymbol{m}(\tau)$ and $\boldsymbol{u}(\tau)$ are presented. In this way, systems switching between several linear, time-invariant, continuous dynamics can be modeled. Moreover, to compact the state representation, a structured marking is used, as proposed in [GU98] and [CPV99]. In addition, for the whole net, colors are used to define a more compact model of the systems. This new kind of net is named Colored MHPN (CMHPN). Before giving its formal definition, the CMHPN formalism is introduced by means of a motivation example.

4.2.2 Motivation Example

Consider two unitary masses moving, without friction, along a path with a uniformly accelerated linear motion, as shown in Fig.

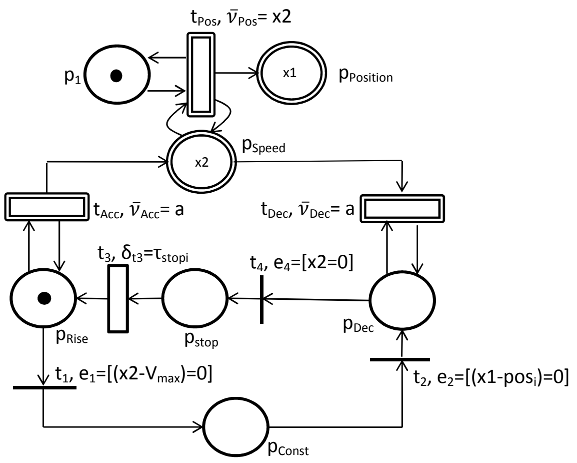


Figure 4.7 A MHPN model of mass i moving along a path.

4.6. Each mass state is described by position $x1$ and speed $x2$, related each other by the following equations:

$$\begin{pmatrix} \dot{x1} \\ \dot{x2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} a \tag{4.1}$$

where a is the constant acceleration. Assume the masses can accelerate until $(V_{max} - x2) = 0$, and then, they continue to move with constant speed, so (4.1) becomes:

$$\begin{pmatrix} \dot{x1} \\ \dot{x2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x1 \\ V_{max} \end{pmatrix} \tag{4.2}$$

To avoid collisions, the masses regulate their speed in the manner that distance between them is equal or greater than a fixed threshold Th. Moreover, each mass has to start to decelerate if its position $x1$ is equal to a certain value pos_i. It decelerates until its speed is zero, then it stays stopped for a time τ_{stopi}, and then it starts to accelerate again only if the distance with the next mass is still greater than the threshold.

To model each mass behavior the modified HPN shown in Fig. 4.7 can be used: the marking of place $p_{Positon}$ represents the actual mass position, while the marking of place p_{Speed} represents its actual speed. When the mass accelerates (decelerates), the speed value is incremented (decremented) by transition t_{Acc} (t_{Dec}) with a firing speed just equal to the input value a. Position depends on the firing speed of transition t_{Pos}, which is equal to p_{Speed} marking. Note that when $m_{p_{Speed}} = 0$, $\nu_{Pos} = 0$ and, consequently, even if t_{Pos} is still enabled, it does not change $m_{p_{Position}}$. Transition t_{Acc} (t_{Dec}) can fire only when discrete place p_{Rise} (p_{Dec}) is marked. Firing of discrete uncontrollable transition t_1 (t_2) is associated to the internal condition e_1 (e_2) that is verified when the vehicle reaches the maximum allowed speed value (the vehicle reaches a determinate position). When t_1 fires, discrete place p_{Const} becomes marked: both t_{Acc} and t_{Dec} are disabled, consequently the marking of p_{Speed} remains constant (i.e. the vehicle runs with constant speed). Uncontrollable discrete transition t_4 fires when marking x_2 is equal to zero (i.e. the vehicle is stopped): in such a case a token is put in discrete place p_{Stop}, enabling discrete timed transition t_3. When a time equal to the firing delay δ_{t3} is expired, t_3 fires, enabling t_{Acc} (i.e. the vehicle starts to move again).

The whole system is modeled replicating the net shown in Fig. 4.7 for each mass. To have a more compact representation colors, presented in [Jen95] and [HK98], are introduced in the HPN model. A different color is associated to each mass, so the system can be modeled with just one net that evolves w.r.t. two colors. For the sake of clarity, in Fig. 4.8 the marking of a discrete place w.r.t. the color i is indicated as c_i; the marking of the continuous places is indicated as $(x1)_i$ or $(x2)_i$ in the manner that its meaning is still obvious to the reader. Note that now firing speeds (both instantaneous and maximal), firing delays and logical expressions are column vectors, of dimension equal to the colors number. The i-th element of firing speeds (firing delays or logical expressions) vector associated to a continuous (discrete) transition is the firing speed (firing delay or logic expression) associated to the transition, w.r.t. the i-th color. Moreover, two new discrete uncontrollable transitions, t_5 and t_6, have been added compared to the single mass model; these transitions manage the mass speed when the distance between the masses violates the threshold. In-

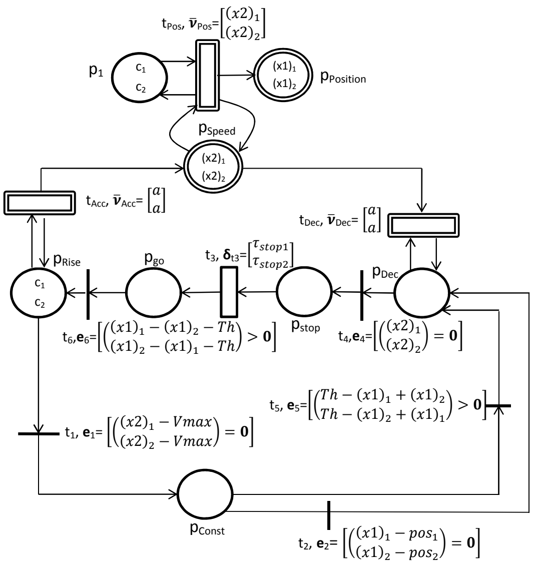

t_{Pos}, $\bar{\boldsymbol{v}}_{Pos}=\begin{bmatrix}(x2)_1\\(x2)_2\end{bmatrix}$
p_1
c_1 c_2
$(x1)_1$ $(x1)_2$
$p_{Position}$
p_{Speed}
$(x2)_1$ $(x2)_2$
t_{Acc}, $\bar{\boldsymbol{v}}_{Acc}=\begin{bmatrix}a\\a\end{bmatrix}$
t_{Dec}, $\bar{\boldsymbol{v}}_{Dec}=\begin{bmatrix}a\\a\end{bmatrix}$
p_{Rise}
p_{go}
t_3, $\boldsymbol{\delta}_{t3}=\begin{bmatrix}\tau_{stop1}\\\tau_{stop2}\end{bmatrix}$
p_{Dec}
c_1 c_2
t_6,$\mathbf{e}_6=\left[\begin{pmatrix}(x1)_1-(x1)_2-Th\\(x1)_2-(x1)_1-Th\end{pmatrix}>\mathbf{0}\right]$
p_{stop}
t_4,$\mathbf{e}_4=\left[\begin{pmatrix}(x2)_1\\(x2)_2\end{pmatrix}=\mathbf{0}\right]$
t_5, $\mathbf{e}_5=\left[\begin{pmatrix}Th-(x1)_1+(x1)_2\\Th-(x1)_2+(x1)_1\end{pmatrix}>\mathbf{0}\right]$
t_1, $\mathbf{e}_1=\left[\begin{pmatrix}(x2)_1-Vmax\\(x2)_2-Vmax\end{pmatrix}=\mathbf{0}\right]$
p_{Const}
t_2, $\mathbf{e}_2=\left[\begin{pmatrix}(x1)_1-pos_1\\(x1)_2-pos_2\end{pmatrix}=\mathbf{0}\right]$

Figure 4.8 The use of colors in a MHPN model of two masses moving along a path.

deed, transition t_5 fires when a threshold violation occurs (internal condition e_5 is verified). Its firing enables t_{Dec} w.r.t. the appropriate color so that the vehicle associated to such a color begins to decelerate. Transition t_6 fires if the distance between the two masses is greater of the threshold (internal condition e_6 is verified). Threshold violation can be managed as an internal condition associated to t_5 and t_6, since the model represents the behavior of both the masses; with the uncolored model of Fig. 4.7 threshold violation can be detected only using an external controller that, looking at the state of the two masses, properly manages their speeds.

Finally, using a structured continuous marking, a more compact representation of the masses state can be obtained, as shown in Fig. 4.9. Now, the two state variables are collected in a vector which is the marking of the new place p_{Mass}. This place is obtained by the fusion of $p_{Position}$ and p_{Speed}. There is a vector for each place color and it is used the notation $< x1, x2 >_i = < \boldsymbol{x} >_i$ to indicate the structured marking w.r.t. the color i. Marking of discrete places is still represented by c_i. Notice that, for the logic expressions, the following synthetic notations are used: given two vectors, $< x, y >$ and $< w, z >$, having two real components, notation $< x, y > + < w, z >$ stands for $< (x + w), (y + z) >$; $< x, y > - < w, z >$ stands for $< (x-w), (y-z) >$; $(< x, y >) = \mathbf{0}$ means $(x = 0$ and $y = 0)$; $(< x, y >) \gneq \mathbf{0}$ means the vector has at least one component greater than 0; $< x, K >_j$ is a vector having as first component the first attribute of the marking, w.r.t. color j, and as second element the constant value K.

Maximal firing speeds are still column vectors of two elements (one for each color of the net), but now, the i-th element is exactly either $\boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{B} \cdot a$ or $\boldsymbol{A} \cdot \boldsymbol{x}$ (according to (4.1) and (4.2)), with $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \boldsymbol{A}$, $\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \boldsymbol{B}$ and $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \boldsymbol{x}$; as for example, supposing transition t_{Acc} is firing during the time interval $[\tau_{k-1}, \tau_k]$, at the time τ_k, the state of the mass will be given by:

$$\begin{cases} x_1(\tau_k) & = & x_1(\tau_{k-1}) + \int_{\tau_{k-1}}^{\tau_k} (x_2(\tau))d\tau \\ \\ x_2(\tau_k) & = & x_2(\tau_{k-1}) + \int_{\tau_{k-1}}^{\tau_k} (a)d\tau \end{cases} \tag{4.3}$$

where with $x_i(\tau_{k-1})$ it is indicated the value of x_i at the instant τ_{k-1}.

Finally, note that the use of the structured marking requires the introduction of new arcs, connecting continuous transitions with place p_{Mass}, to modify separately masses position and speed, during the different dynamics. The separation of the effects is obtained using the following weights:

$$\mathbf{Post}(p_{Mass}, t_{Dec}) = \begin{bmatrix} \boldsymbol{\pi}_1 & 0 \\ 0 & \boldsymbol{\pi}_1 \end{bmatrix},$$

$$\mathbf{Pre}(p_{Mass}, t_{Dec}) = \begin{bmatrix} \boldsymbol{\pi}_2 & 0 \\ 0 & \boldsymbol{\pi}_2 \end{bmatrix},$$

$$\mathbf{Pre}(p_{Mass}, t_{Pos}) = \begin{bmatrix} \boldsymbol{\pi}_2 & 0 \\ 0 & \boldsymbol{\pi}_2 \end{bmatrix},$$

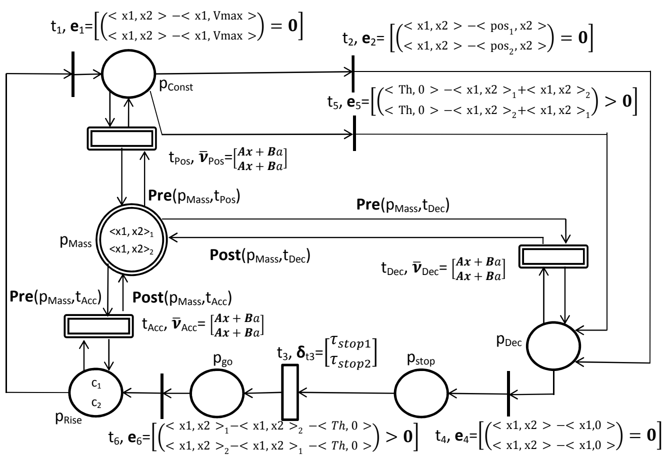

t_1, $\mathbf{e}_1=\left[\begin{pmatrix} <x1,x2> - <x1,Vmax> \\ <x1,x2> - <x1,Vmax> \end{pmatrix} = \mathbf{0}\right]$
t_2, $\mathbf{e}_2=\left[\begin{pmatrix} <x1,x2> - <pos_1,x2> \\ <x1,x2> - <pos_2,x2> \end{pmatrix} = \mathbf{0}\right]$
p_{Const}
t_5, $\mathbf{e}_5=\left[\begin{pmatrix} <Th,0> - <x1,x2>_1 + <x1,x2>_2 \\ <Th,0> - <x1,x2>_2 + <x1,x2>_1 \end{pmatrix} > \mathbf{0}\right]$
t_{Pos}, $\bar{\boldsymbol{\nu}}_{Pos}=\begin{bmatrix} Ax+Ba \\ Ax+Ba \end{bmatrix}$
$\mathbf{Pre}(p_{Mass},t_{Pos})$
$\mathbf{Pre}(p_{Mass},t_{Dec})$
p_{Mass}
$<x1,x2>_1$
$<x1,x2>_2$
$\mathbf{Post}(p_{Mass},t_{Dec})$
t_{Dec}, $\bar{\boldsymbol{\nu}}_{Dec}=\begin{bmatrix} Ax+Ba \\ Ax+Ba \end{bmatrix}$
$\mathbf{Pre}(p_{Mass},t_{Acc})$
$\mathbf{Post}(p_{Mass},t_{Acc})$
t_{Acc}, $\bar{\boldsymbol{\nu}}_{Acc}=\begin{bmatrix} Ax+Ba \\ Ax+Ba \end{bmatrix}$
p_{Dec}
t_3, $\boldsymbol{\delta}_{t3}=\begin{bmatrix} \tau_{stop1} \\ \tau_{stop2} \end{bmatrix}$
p_{go}
p_{stop}
c_1
c_2
p_{Rise}
t_6, $\mathbf{e}_6=\left[\begin{pmatrix} <x1,x2>_1 - <x1,x2>_2 - <Th,0> \\ <x1,x2>_2 - <x1,x2>_1 - <Th,0> \end{pmatrix} > \mathbf{0}\right]$
t_4, $\mathbf{e}_4=\left[\begin{pmatrix} <x1,x2> - <x1,0> \\ <x1,x2> - <x1,0> \end{pmatrix} = \mathbf{0}\right]$

Figure 4.9 A CMHPN model of two masses moving along a path.

$$\mathbf{Post}(p_{Mass}, t_{Acc}) = \begin{bmatrix} \boldsymbol{\pi}_3 & 0 \\ 0 & \boldsymbol{\pi}_3 \end{bmatrix},$$

$$\mathbf{Pre}(p_{Mass}, t_{Acc}) = \begin{bmatrix} \boldsymbol{\pi}_2 & 0 \\ 0 & \boldsymbol{\pi}_2 \end{bmatrix}$$

with

$$\boldsymbol{\pi}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{\pi}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{\pi}_3 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

These weights allow to represent with just one place and one transition concurrent decreasing of an attribute and rising of the other one, while with the unstructured net they are modeled using two separated continuous places.

4.2.3 Formal definition

A CMHPN is a four-tuple $\mathcal{M} = \{\mathcal{H}, Cl, Co, \boldsymbol{\nu}\}$ where $\mathcal{H}$ is a HPN; Cl is the set of colors. Co: $P \cup T \longrightarrow Cl$ is a color function that associates to each element in $P \cup T$ a set of colors; for all $t_i^C \in T^C$, $\boldsymbol{\nu}$ is the mapping $Co(t_i^C) \rightarrow \mathbb{R}^+$ that associates an instantaneous firing speed to each color of the continuous transition t_i^C. For all $p_i \in P$, $Co(p_i) = \{a_{i,1}, a_{i,2}, ..., a_{i,u_i}\} \subseteq Cl$ is the set of possible colors of p_i, and u_i is their number. For all $t_j \in T, Co(t_j) = \left\{b_{j,1}, b_{j,2}, ..., b_{j,v_j}\right\} \subseteq Cl$ is the set of possible occurrence colors of t_j and v_j is their number. For all $p_i \in P^D$, the marking $\boldsymbol{m}^D_{p_i}$ is defined as the mapping $Co(p_i) \to \mathbb{N}$ that associates to each possible color of p_i a non-negative integer representing the number of tokens of that color contained in place p_i. For the sake of simplicity, a discrete marking w.r.t. color r is indicated as c_r. Logical expressions associated to the discrete transitions are column vectors of size v_j: their r-th element corresponds to the logical function associated to the transition firing color r.

For all $p_i \in P^C$, the structured marking $\boldsymbol{m}^C_{p_i}$ is defined as the mapping $Co(p_i) \to (\mathbb{R}^+)^{(q)}$, thus, at each place $p_i \in P^C$, w.r.t. the color r, a vector of q non-negative real numbers,

$$< \underbrace{x_1 \dots x_q}_{attributes} >_r,$$

is associated. The q values of the marking are called "attributes" and they completely describe the state of the system.

For all $t^C_i \in T^C$, $\boldsymbol{\nu}(t^C_i) = \boldsymbol{\nu}_i = (\nu_{i,1}, \nu_{i,2}, \dots, \nu_{i,v_c})^T$ is the vector of firing speeds of the continuous transition t^C_i. Its r-th element $\nu_{i,r}$, is the firing speed of t^C_i w.r.t. the color r and, when t^C_i is enabled, it is a linear function of the marking of the t^C_i input places. Similarly, $\forall t^D_i \in T^D$, $\boldsymbol{\delta}_i = (\delta_{i,1}, \dots, \delta_{i,v_d})^T$ is the column vector of the discrete transition t^D_i firing delays. The r-th element $\delta_{i,r}$ is the firing delay associated to the color r.

$\mathbf{Pre}(p_i, t_j)$ is a mapping $\mathbf{Pre}(p_i, t_j) : Co(t_j) \to \mathbb{R}^+(Co(p_i))$, for $i = 1, \dots, w = w_d + w_c$ and $j = 1, \dots, n = n_d + n_c$. At the same way $\mathbf{Post}(p_i, t_j)$ is defined as the mapping $\mathbf{Post}(p_i, t_j) : Co(t_j) \to \mathbb{R}^+(Co(p_i))$, for $i = 1, \dots, w$ and $j = 1, \dots, n$. $\mathbf{Pre}(\mathbf{Post})(p_i, t_j)$ is a matrix of dimensions $u_i \times v_j$; the element $Pre(p_i, t_j)(r, s) = Pre_{ij,rs}$ $(Post(p_i, t_j)(r, s) = Post_{ij,rs})$ is the weight of the arc connecting p_i (t_i) w.r.t. the color r (color s) to t_j (p_i), w.r.t. the color s (color r). The nature of the element depends on the kind of nodes it connects, e.g. weighs of arcs connecting transitions to discrete places are non-negative integer numbers, while weighs of arcs connecting transitions to continuous places are row vectors of non-negative real numbers, with dimension equal to q. When weights are diagonal matrices, having all the diagonal elements equal to 1, weights are not reported near the arcs. The incidence matrix $\boldsymbol{C}$ can be written as

$$\boldsymbol{C} = \left(\begin{array}{c|c} \boldsymbol{C}_{CC} & \boldsymbol{C}_{CD} \\ \hline \boldsymbol{C}_{DC} & \boldsymbol{C}_{DD} \end{array} \right) \tag{4.4}$$

(the meaning of the blocks is the same of that in Chapter 2, Section 2.2). Elements of $\boldsymbol{C}$ are the matrices $\boldsymbol{C}(p_i, t_j) = \mathbf{Post}(p_i, t_j) - \mathbf{Pre}(p_i, t_j)$ with dimension $u_i \times v_j$.

4.2.4 Model of the IS

The IS can be viewed as formed of two interacting subsystems: the guidepath and the vehicles (that move along the guidepath). The guidepath subsystem can be divided in zones (or routes):

1. each zone can be traveled over by more than one vehicle at time;

2. each zone can border with other routes.

Vehicles turn along the IS following a guidepath depending on their destination. When they are in a zone, vehicles

1. can cross it with constant speed;

2. can stop at the interface points to load (unload) a SU from (to) another subsystem bay;

3. can vary their speed with constant acceleration if a particular condition occur (distance between two vehicles goes under a threshold, a particular point is reached...).

4.2.5 Controller

Vehicles run along the guidepath to move SUs from an interface point to another, where they halt to exchange SUs with the bays; the vehicle destination can be defined as one of the interface points. When an exchange with a bay occurs, the vehicle destination changes (a free vehicle can become busy so its destination becomes the bay where it has to unload the SU or, viceversa, a busy vehicle can become free and its destination becomes the first interface point where a SU is waiting to be loaded). As it is obvious, a vehicle runs along the guidepath, stopping only if it is too near to another vehicle: it has no information about its destination. The assignment of destinations and the relative stops are managed by an external controller, that interfaces with the IS model by means of vectors of control inputs (refer to Section 4.2.6 for more details). On the basis of the vehicles destinations, controller decides the next zone a vehicle will enter in (refer to Section 4.2.9 for more details).

4.2.6 Vehicles

Vehicles subsystem is reported at the bottom of Fig. 4.10. The colored continuous place p_s models the vehicles state: it has N_s different colors (where N_s is the number of the vehicles in the IS) and a structured marking with two attributes. For each color, attributes of the marking represent position x_1 and speed x_2 of the corresponding vehicle: as it will be explained in Section 4.2.7, it is $0 \leq x_1 \leq L_i$, where L_i is the length of the zone the vehicle is traveling along. Evolution of the vehicles state is due to the firing of the colored continuous structured transitions t_{acc}, t_{dec} and t_{slow}, t_{const} (each one having N_s colors) that are enabled if vehicles are accelerating, decelerating or moving with constant speed, respectively; the other discrete transitions have N_s colors and they govern the vehicle dynamics switches: for the sake of simplicity, in Fig. 4.10, the expressions of the internal conditions are not reported, but they can be read in Table 4.11. Internal conditions are analytical expression of the marking $\boldsymbol{m}_{p_s}$; the same synthetic notations of Section 4.2.2 are used. When a vehicle is near to an interface point where it has to stop, it starts to decelerate (it corresponds to the firing of the discrete controllable transition t_{sl} after that the

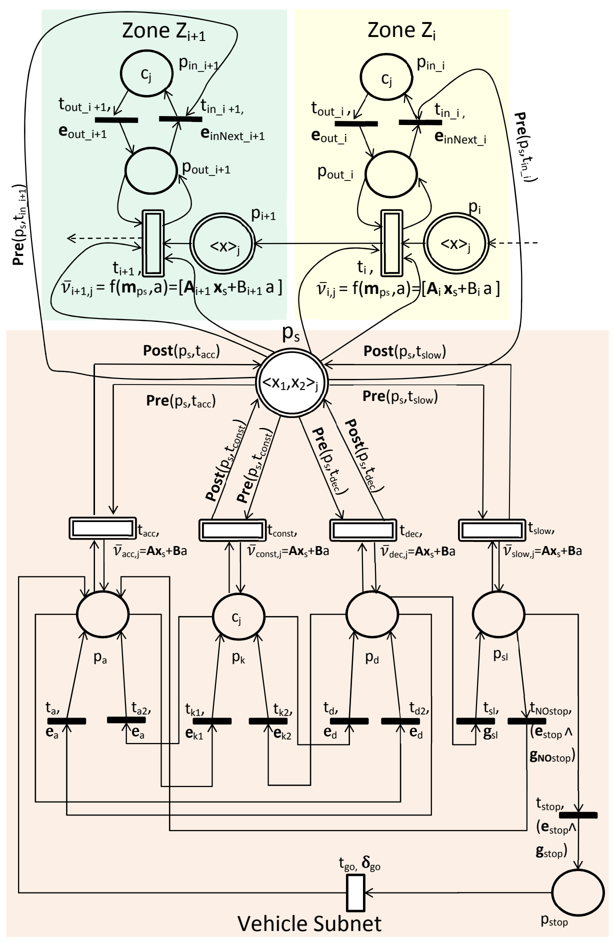

Zone Z_{i+1}
Zone Z_i
c_j
p_{in_i+1}
p_{in_i}
t_{out_i+1}, $\mathbf{e}_{out\_i+1}$
t_{in_i+1}, $\mathbf{e}_{inNext\_i+1}$
t_{out_i}, $\mathbf{e}_{out\_i}$
t_{in_i}, $\mathbf{e}_{inNext\_i}$
p_{out_i+1}
p_{out_i}
p_{i+1}
p_i
$<x>_j$
t_{i+1}, $\bar{v}_{i+1,j} = f(\mathbf{m}_{ps}, a) = [\mathbf{A}_{i+1}\,\mathbf{x}_s + B_{i+1}\,a]$
t_i, $\bar{v}_{i,j} = f(\mathbf{m}_{ps}, a) = [\mathbf{A}_i\,\mathbf{x}_s + B_i\,a]$
$\mathbf{Pre}(p_s, t_{in\_i+1})$
$\mathbf{Pre}(p_s, t_{in\_i})$
p_s
$<x_1, x_2>_j$
$\mathbf{Post}(p_s, t_{acc})$
$\mathbf{Pre}(p_s, t_{acc})$
$\mathbf{Post}(p_s, t_{slow})$
$\mathbf{Pre}(p_s, t_{slow})$
$\mathbf{Post}(p_s, t_{const})$
$\mathbf{Pre}(p_s, t_{const})$
$\mathbf{Pre}(p_s, t_{dec})$
$\mathbf{Post}(p_s, t_{dec})$
t_{acc}, $\bar{v}_{acc,j} = \mathbf{Ax}_s + \mathbf{Ba}$
t_{const}, $\bar{v}_{const,j} = \mathbf{Ax}_s + \mathbf{Ba}$
t_{dec}, $\bar{v}_{dec,j} = \mathbf{Ax}_s + \mathbf{Ba}$
t_{slow}, $\bar{v}_{slow,j} = \mathbf{Ax}_s + \mathbf{Ba}$
p_a
p_k
p_d
p_{sl}
t_a, $\mathbf{e}_a$
t_{a2}, $\mathbf{e}_a$
t_{k1}, $\mathbf{e}_{k1}$
t_{k2}, $\mathbf{e}_{k2}$
t_d, $\mathbf{e}_d$
t_{d2}, $\mathbf{e}_d$
t_{sl}, $\mathbf{g}_{sl}$
t_{NOstop}, $(\mathbf{e}_{stop} \wedge \mathbf{g}_{NOstop})$
t_{stop}, $(\mathbf{e}_{stop} \wedge \mathbf{g}_{stop})$
t_{go}, δ_{go}
p_{stop}
Vehicle Subnet

Figure 4.10 Model of one vehicle and two adjacent zones of the guidepath, Z_i and Z_{i+1}.

controller has set to 1 the corresponding element of the vector $\boldsymbol{g}_{sl}$). In this way the vehicle arrives to the interface with $x_2 = 0$, then it steadies in that position for the time it takes to load (unload) a SU (it corresponds to the firing of discrete controllable transition t_{stop}, first, and to the firing of timed transition t_{go} with firing delay δ_{go}, then). $\mathbf{Post}(p_s, t_{dec})$ and $\mathbf{Pre}(p_s, t_{dec})$ $\big(\mathbf{Post}(p_s, t_{slow})$ and $\mathbf{Pre}(p_s, t_{slow})\big)$ are two N_s-size diagonal matrices having, respectively, the diagonal's elements equal to $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ and the other elements equal to a 2×2 null matrix; $\mathbf{Pre}(p_s, t_{acc})$ and $\mathbf{Post}(p_s, t_{acc})$ are two N_s-size diagonal matrices having, respectively, the diagonal's elements equal to $\boldsymbol{\pi}_2$ and $\boldsymbol{\pi}_3$ and the other elements equal to a 2×2 null matrix; $\mathbf{Pre}(p_s, t_{const})$ is a N_s-size diagonal matrix having the diagonal's elements equal to $\boldsymbol{\pi}_2$ and the other elements equal to a 2×2 null matrix. Continuous transition maximal firing speeds are linear functions of marking $\boldsymbol{m}_{p_s}$ and of the input a. They are column vectors of size N_s having the j-th element equal to
$\boldsymbol{A}\cdot\boldsymbol{x}_s+\boldsymbol{B}\cdot a$ with $\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $\boldsymbol{B} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and $\boldsymbol{x}_s = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, where x_1 and x_2 are the attributes of $\boldsymbol{m}_{p_s}$ w.r.t. the color j;

Synchronization between vehicles and belts and vehicles and SUs at the interface zone is obtained by means of the controllable transitions t_{stop} and t_{NOstop}. They are the interface with the models of (picking and/or crane) bays and their firing depends on the external control inputs $\boldsymbol{g}_{stop}$ and $\boldsymbol{g}_{NOstop} = NOT(\boldsymbol{g}_{stop})$ (column vectors with dimension N_s); for each firing color j, the j-th element of $\boldsymbol{g}_{stop}$, $g_{stop,j}$, is 1 iff exchange is possible (destination belt is free or a SU is present to the interface point). If condition for exchange is not satisfied, the vehicle does not stay but it starts to move again soon after its stop.

4.2.7 Zones

At the top of Fig. 4.10 the model of two adjacent zones, Z_i and Z_{i+1}, with length L_i and L_{i+1}, respectively, is shown. When the IS is made up of more than two zones, subnet modeling Z_i or Z_{i+1}

Figure 4.11 Internal conditions and their meaning.

Internal Condition	Expression of the j-th element of the j-th element	Meaning
$\boldsymbol{e}_{k1}$	$\big[\ (< x_1, x_2 >_j - < x_1, V_{max} >_j) = \mathbf{0} \vee$	mass speed is equal to V_{max} OR
	$(< x_1, x_2 >_j - < x_1, V_{in} >_j) = \mathbf{0}\ \big]$	it is equal to the desired value
$\boldsymbol{e}_{k2}$	$\big[(< x_1, x_2 >_j - < x_1, x_2 >_k - < Th, 0 >) \geq \mathbf{0} \wedge$	distance between mass j and k is $\geq Th$ AND
	$((< x_1, x_2 >_j - < x_1, 0 >_j) = \mathbf{0} \vee$	(mass speed is $= \mathbf{0}$ OR
	$(< x_1, x_2 >_j - < x_1, V_{in} >_j)) = \mathbf{0}\ \big]$	mass speed is equal to a desired value)
$\boldsymbol{e}_a$	$\big[\ ((< x_1, V_{max} >_j - < x_1, x_2 >_j) \gneq \mathbf{0} \vee$	(speed value is lower than V_{max} OR
	$(< x_1, V_{in} >_j - < x_1, x_2 >_j) \gneq \mathbf{0}) \wedge$	it is lower than the desired value) AND
	$((< x_1, x_2 >_j - < x_1, x_2 >_k - < Th, 0 >) \gneq \mathbf{0})\ \big]$	distance between mass j and k is equal $\geq Th$
$\boldsymbol{e}_d$	$\big[\ (< x_1, x_2 >_j - < x_1, 0 >_j) \gneq \mathbf{0} \wedge$	(speed value is ¿0) AND
	$((< x_1, x_2 >_j - < x_1, V_{in} >_j) \gneq \mathbf{0} \vee$	(speed is greater than the desired value OR
	$(< Th, 0 > - < x_1, x_2 >_j + < x_1, x_2 >_k) \gneq \mathbf{0})\ \big]$	the distance with the next mass is $< Th$)
$\boldsymbol{e}_{stop}$	$\big[\ (< x_1, x_2 >_j - < x_1, 0 >_j) = \mathbf{0}\ \big]$	mass speed is $= 0$.

has to be repeated for each other zone to be added: in Fig. 4.10 doted arcs represent arcs connecting Z_i and Z_{i+1} with the previous and the next zone, respectively.

In the following, for the sake of brevity, only the description about places and transitions of the Z_i model are given: places and transitions of Z_{i+1} have the same meaning.

The continuous colored place p_i has N_s colors; its structured marking is made up of just one attribute, x. Its value indicates if a vehicle is totally in Z_i, it is out of Z_i or it is going in (out) to Z_i. In particular, said d the vehicle dimension, the marking of p_i is $x = d$, $x = 0$ or $0 < x < d$, respectively. The colored continuous transition t_i models the going out of a vehicle from Z_i. Discrete colored transitions t_{out_i} and t_{in_i} enable and disable t_i. All these transitions have N_s colors. Transition t_{out_i} is associated to the internal condition $\boldsymbol{e}_{out\_i}$ whose j-th element

$$e_{out\_i,j} = [< x_1, x_2 >_j - < Li, x_2 >_j = 0] \tag{4.5}$$

is the internal condition associated to the firing color j, where $< x_1, x_2 >_j$ is the j-th vehicle state vector and $< Li, x_2 >_j$ is a vector having as elements the length of the zone the vehicle is crossing and the vehicle speed; transition t_{in_i} fires, w.r.t. color j, when vehicle j is completely in the next zone: it is associated to the internal condition $\boldsymbol{e}_{inNext\_i}$ whose j-th element

$$e_{inNext\_i,j} = [< x >_j = 0] \tag{4.6}$$

is the internal condition associated to the firing color j, where $< x >_j$ is the marking of place p_i w.r.t. color j. When a vehicle enters in a new zone a jump in its state occurs and the component x_1 of $\boldsymbol{m}_{p_s}$ is forced to 0 by t_{in_i}. $\mathbf{Pre}(p_s, t_{in\_i})$ is a N_s-size diagonal matrix, having the diagonal's elements equal to L_i.

Maximal transition t_i firing speed $\overline{\boldsymbol{\nu}}_i$ is function of $\boldsymbol{m}_{p_s}$ (the marking of the vehicle model continuous place, p_s). In particular each element $\overline{\nu}_{i,j}$ is equal to the vehicle speed x_2, consequently $\mathbf{x_s} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $\mathbf{A_i} = \begin{pmatrix} 0 & 1 \end{pmatrix}$ and $B_i = 0$.

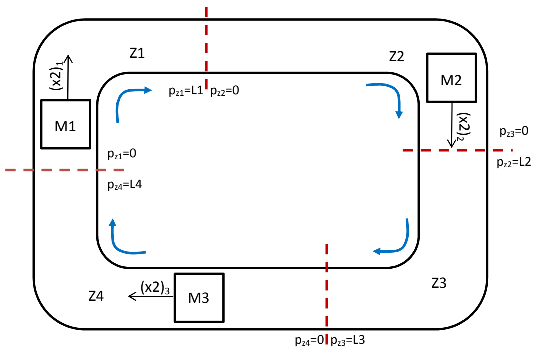


Figure 4.12 Three masses moving along a circuit.

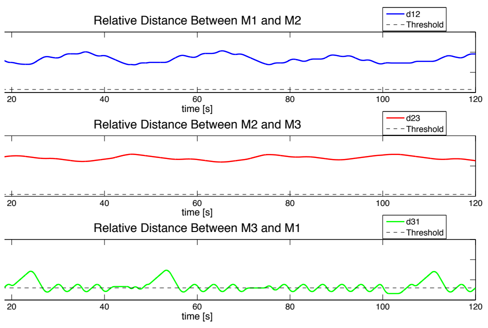
4.2.8 A toy example

Material handling systems, in general, present complex behaviors. The basic task for this kind of systems is the handling of something along a path, from a source to a destination point, avoiding collisions. In this section, the simple system of Fig. 4.12 is modeled to show the effectiveness of CMHPNs in modeling warehouse system.

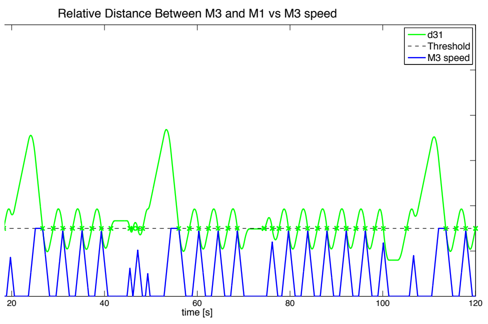
Consider 3 unitary masses, of dimension $d = 1m$, named M_1, M_2 and M_3, moving along a ring, without friction, with initial speed V_{max}, that can be varied with constant acceleration, a. The ring is divided in four zones, each one with the own length L_i. Each mass M_j moves along the path until an external controller orders it to slow down so that M_j can stop at a desired position x_{d_i}. At this point the mass stays for a time δ_{stop} after that it starts to move again. Note that masses do not know neither where nor when they have to stop. Moreover, the distance between two consecutive masses has to be always less than or equal to a fixed threshold. This system can be modeled using the net shown in Fig. 4.10, adding to it other two colors and the subnets modeling the other two zones of the path.

Results of the toy example simulation are reported in Fig. 4.13.

In Fig. 4.13a) the evolution of the relative distance between the masses is shown: d_{ij} is the distance from M_i to M_j, moving in a clockwise direction. Notice that only d_{31} goes under the threshold



(a)



(b)

Figure 4.13 Results of the toy example simulation, with $L_i = 25m\ \forall i$, $V_{max} = 3m/s$ and $a = 2m/s^2$: (a) evolution of the relative distances between masses M_1 and M_2, masses M_2 and M_3 and masses M_3 and M_1; (b) evolution of M_3 speed w.r.t. relative distance between M_3 and M_1 .

during the system evolution, because of the masses initial disposition. As more, it can be seen how the relative distance is always greater than $1m$ (the length of the masses), i.e. no collisions occur. In Fig. 4.13b) evolution of M_3 speed w.r.t. the evolution of d_{31} is shown: to avoid collisions, speed decreases each time d_{31} goes under the threshold value.

4.2.9 Branch points

A branch point is a point where a zone ends and more than one other zone starts: what is the next zone the vehicle will enter in

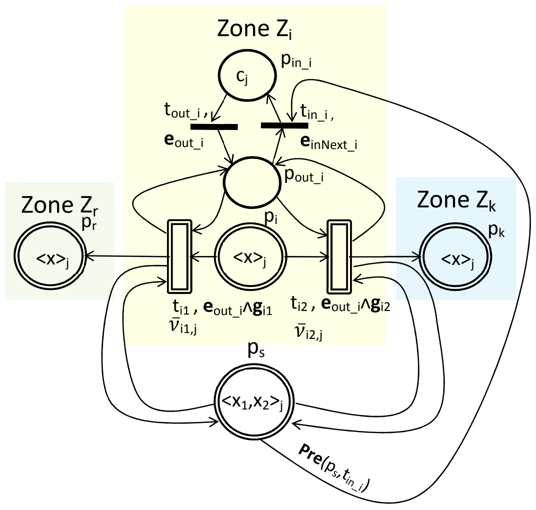
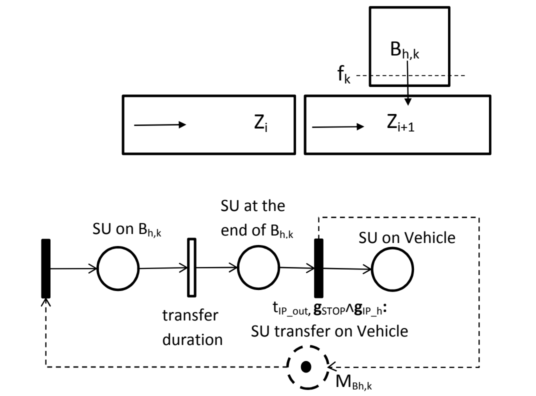


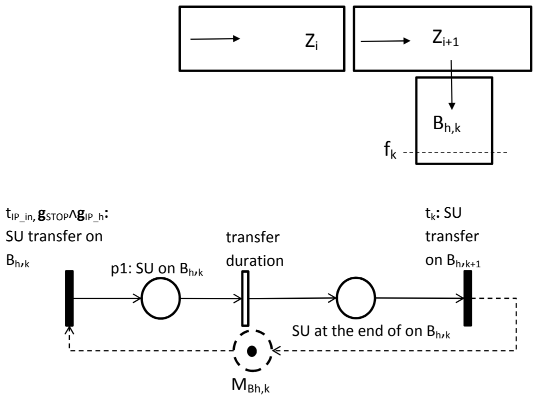
Figure 4.14 Model of a branch point.

depends on vehicle destination. Branch points are modeled in easy way linking the module of the generic i-th zone with that of all the other following routes, as shown in Fig. 4.14 where, for the sake of simplicity, only the continuous places corresponding to the following zones and only the place p_s have been drawn. Let N_d^i be the number of zones bordering on Z_i, N_d^i controllable continuous transitions t_{iz}, with $z = 1 \dots N_d^i$ are added. Each of these transitions is associated to the logical condition $\boldsymbol{e}_{out\_i} \wedge \boldsymbol{g}_{iz}$ where $\boldsymbol{e}_{out\_i}$ is the internal condition whose element $e_{out\_i,j}$ is equal to 1 when the j-th vehicle is arrived at the end of the zone Z_i, while $\boldsymbol{g}_{iz}$ is an external control input whose element $g_{iz,j}$ is equal to 1 if the j-th vehicle has to pass from zone Z_i to zone Z_z. As for the single zone, maximal firing speed $\overline{\nu}_{ik}$ depends on the current speed value of the j-th vehicle. Notice that transitions t_{iz} with $z = 1 \dots N_d^i$, are involved in a conflict, but at each time $\exists j | g_{iz,j} = 1,\ g_{iz,l} = 0,\ \forall l \neq j$ (i.e. the external control input solves the conflict so that vehicle j cannot be moved to more that one destination).



$B_{h,k}$
f_k
Z_i
Z_{i+1}
SU on $B_{h,k}$
SU at the end of $B_{h,k}$
SU on Vehicle
transfer duration
t_{IP_out}, $g_{STOP} \wedge g_{IP\_h}$: SU transfer on Vehicle
$M_{Bh,k}$

(a)



Z_i
Z_{i+1}
$B_{h,k}$
f_k
t_{IP_in}, $g_{STOP} \wedge g_{IP\_h}$: SU transfer on $B_{h,k}$
p1: SU on $B_{h,k}$
transfer duration
t_k: SU transfer on $B_{h,k+1}$
SU at the end of on $B_{h,k}$
$M_{Bh,k}$

(b)

Figure 4.15 Interface points model (a) between a bay and the IS and (b) between the IS and a bay .

4.2.10 Interface between the IS and bays

The interface between the IS and bays (and viceversa) can occur only when a vehicle is aligned at an interface point. When this happens a SU is passed from a bay to the vehicle or viceversa. In Fig. 4.15 the nets modeling the exchanges are shown. Transition t_{IP_in} (t_{IP_out}) is an immediate controlled colored transition, having N_s firing colors, whose firing depends on the occurrence of the logical condition $\boldsymbol{g}_{stop} \wedge \boldsymbol{g}_{IP_h}$. External input $\boldsymbol{g}_{stop}$ is the same condition managing the firing of controllable transition t_{stop} in the vehicles subsystem (refer to Fig. 4.10) and its j-th component is equal to 1 if the j-th vehicle is arrived at the interface point it was directed to. External input $\boldsymbol{g}_{IP_h}$ is a vector of N_s elements: the j-th element $g_{IP_h,j}$ is equal to 1 if at the interface point h an exchange with the vehicle j can occur. Notice that $\boldsymbol{g}_{stop}$ is common to all the interface points and can have more than one true element at time, instead there is a $\boldsymbol{g}_{IP_h}$ input for each interface point. This meas that even if at the same time there are more than one vehicle aligned to the bays (potentially even in the same zone), $\boldsymbol{g}_{IP_h}$ assures the exchange can occur only between the bay and the vehicle aligned to it.

4.2.11 Liveness analysis

In this section liveness of the CMHPN model of the IS considered in isolation (i.e. without the presence of the bay subsystems) is discussed.

It will be shown that, unless controllable transitions are disabled by the external controller, the CMHPN model of the IS is live, otherwise a livelock condition can be established.

In a PN system $S = \langle \mathcal{N}, \boldsymbol{m}(0) \rangle$ a livelock occurs iif S is deadlock-free and there exists $\boldsymbol{m} \in R(\mathcal{N}, \boldsymbol{m}(0))$, where $R(\mathcal{N}, \boldsymbol{m}(0))=$ the set of the reachable markings of the system, and $t \in T$ s.t. t is dead at $\boldsymbol{m}$ [HL09].

In Section 4.2.12, the relation between SUs exchange between vehicles and bays and livelocks in the IS will be shown.

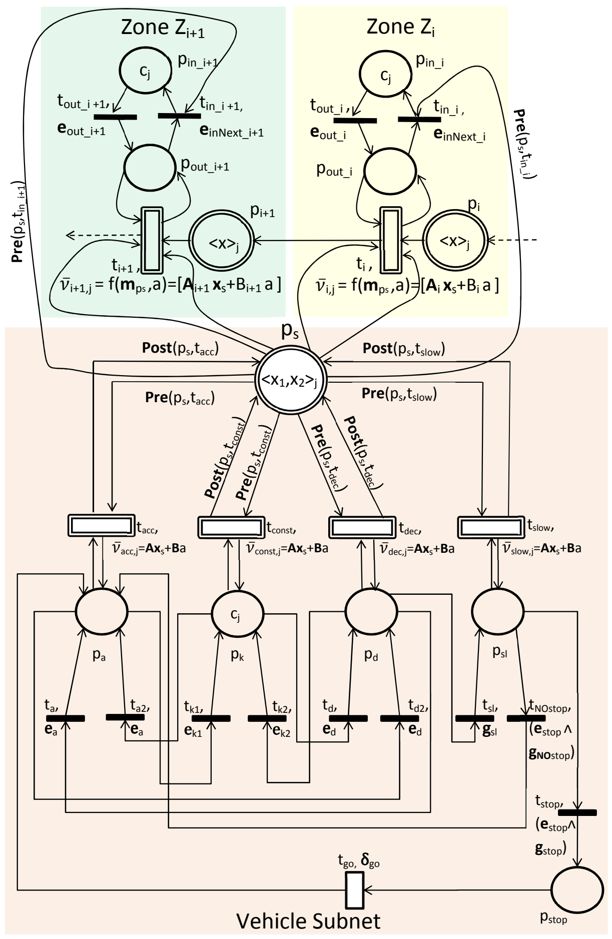


Figure 4.16 Model of vehicles running in the zone Z_i, w.r.t. color j.

A transition t is live under the initial marking $\boldsymbol{m}(0)$ if for every marking $\boldsymbol{m}(\tau)$ reachable from $\boldsymbol{m}(0)$, it exists a sequence σ, fireable from $\boldsymbol{m}(\tau)$, which contains transition t. In other words, whatever the net evolution, a possibility always remains for firing t. A transition t is "quasi-live" under the initial marking $\boldsymbol{m}(0)$ if there is a fireable sequence from $\boldsymbol{m}(0)$ which contains t. In other words, a transition is quasi-live if there is a chance that this transition may be fired. A PN system S is live if all its transitions are live.

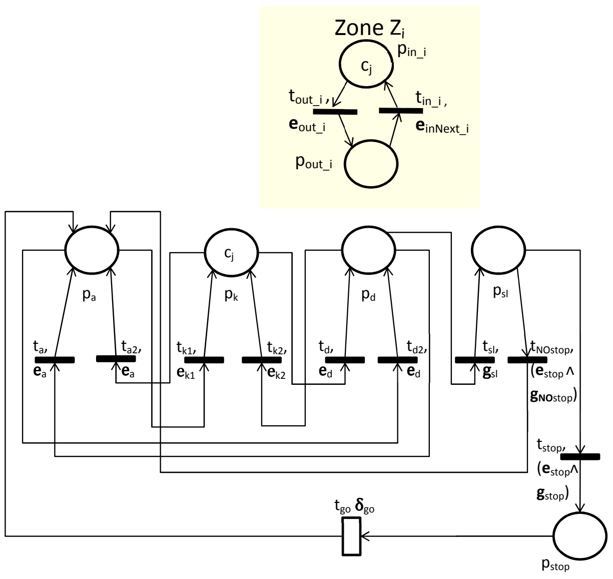


Figure 4.17 Discrete PN obtained eliminating all the continuous nodes from the net of Fig. 4.16.

Liveness of CMHPN system can be analyzed building a Hybrid Automaton (HA) as it will be discussed in the following.

HAs are able to model several hybrid dynamic systems. They were introduced for the first time in [ACH$^+$95].

In [GAS05] an algorithm is proposed to translate a HPN in a HA. Here, such an algorithm is adapted to translate a CMHPN in a HA: steps of the algorithm presented in [GAS05] are repeated for each color of the net model. For the sake of simplicity, in the following the algorithm is recalled referring to a single color:

1. *Isolate the discrete PN of the hybrid model and build its equivalent automaton.*

   From the CMHPN eliminate all the continuous nodes, so to obtain a pure discrete net. As for example in Fig. 4.16 the net modeling one vehicle moving in a single zone of the IS is shown; in Fig. 4.17 it is shown the discrete net obtained eliminating the continuous node from this net. Each reachable marking of the discrete net corresponds to a different hybrid

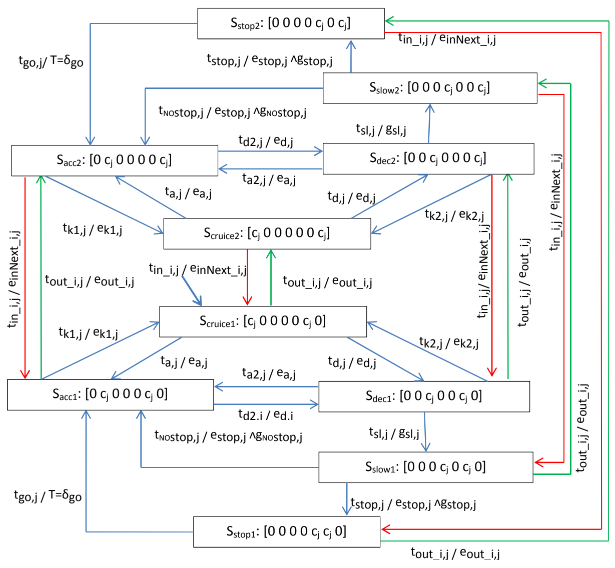


Figure 4.18 Translation algorithm Step 1: construction of macro-locations.

system dynamic. An automaton can be associated to that net; from now on, it is called equivalent automaton. Each reachable marking of the net corresponds to a different state of the equivalent automaton. Each state of this automaton is called "macro-location". In Fig. 4.18 the equivalent automaton resulting from the net of Fig. 4.17 is shown: the marking $\boldsymbol{m}_k^D = [m_{p_k}, m_{p_a}, m_{p_d}, m_{p_{sl}}, m_{p_{stop}}, m_{p_{in\_i}}, m_{p_{out\_i}}]$ corresponding to the macro-location S_k is indicated after the name of the macro-location. If a transition is synchronized to a logical expression then the expression is indicated near the arc, after the transition name; the same is done with the timed transition firing delays.

2. *Construct the HA corresponding to each macro-location of the automaton resulting from the previous step.*

   Starting from the discrete net equivalent automaton, for each macro-location:

   a) Draw the corresponding continuous net, eliminating all

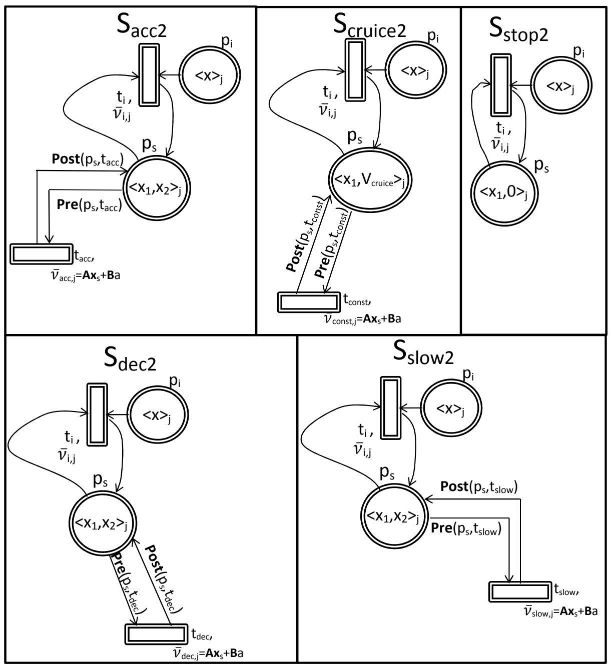


Figure 4.19 Continuous PNs associated to each macro-locations of the automaton of Fig. 4.18. For the sake of simplicity only the continuous net corresponding to macro-locations where $m_{p_{in}} = 0$ and $m_{p_{out}} = c_j$ have been drawn since the others are exactly the same unless the nodes p_i and t_i.

the nodes from the CMHPN but the enabled continuous transitions and their pre and postsets (see Fig. 4.19);

b) Referring to each obtained continuous net, for each continuous place p_j, compute its balance $\dot{\mathbf{m}}_{\mathbf{p_j}}$ (see Chapter 2).

c) Individuate the conditions (in terms of marking values) each dynamic is valid for. At each macro-location more than one continuous dynamic can be associated and transitions between them are due only to the continuous nodes. As for example, consider the macro-location $S_{cruise2}$ in Fig. 4.19: two continuous dynamics are associated to this macro-location. In the first one $m_{p_i,j}$ is greater than zero. Consequently the enabled continuous transition t_i fires with firing speed $\overline{\nu}_{i,j}$, decreasing $m_{p_i,j}$. When the marking of p_i becomes zero, even if the discrete marking is still the same, t_i cannot fire anymore: the second continuous dynamic has been reached.

The output at the current step is a hierarchical automaton whose macro-location contains a HA describing the continuous dynamics in it. In Fig. 4.20a), for the sake of brevity, only the part of the hierarchical automaton associated to the macro-location $S_{cruise1}$ and $S_{cruise2}$ of the equivalent automaton of Fig. 4.18, is shown: a structured notation is used to indicate both the continuous markings value (refer to Part I for the synthetic notation explanation) and their balance, $\dot{\mathbf{m}}_{\mathbf{ps}}, \dot{\mathbf{m}}_{\mathbf{p_i}}, \dot{\mathbf{m}}_{\mathbf{p_{i+1}}}$; $< x1, x2 >$ indicates that the value of the components of $\boldsymbol{m}_{ps}$ is any value in the set $\left[0, L_i\right]$ for $x1$ and $\left[0, V_{max}\right]$ for $x2$ s.t. the shown condition is satisfy; $< x >$ is a value minor than or equal to d (the vehicle dimension); $V_{cruise} \in \left\{V_{in}, V_{max}\right\}$, with $V_{max} \geq V_{in}$.

3. *Replace transitions between macro-locations with transitions between internal states.* This step consists in associating the input/output arcs of each equivalent automaton macro-

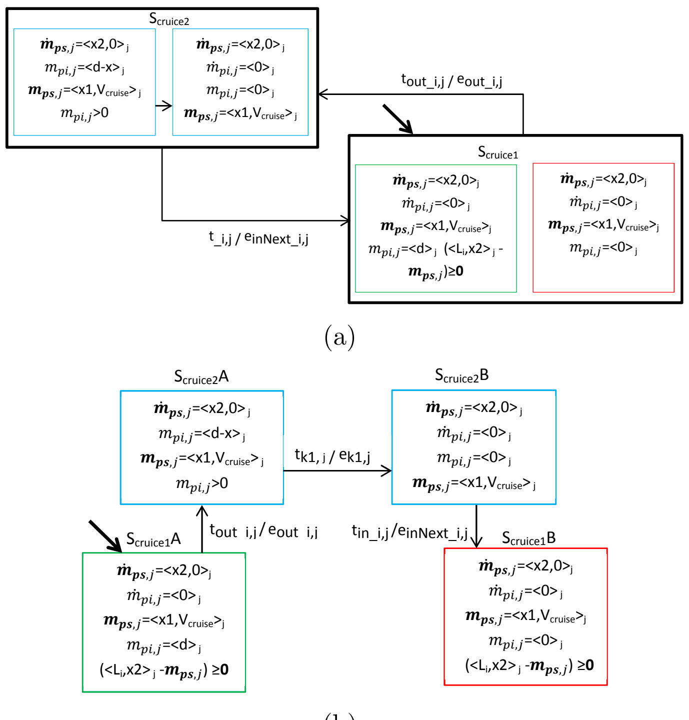


Figure 4.20 Translation algorithm Step 2: hierarchical hybrid automaton construction for macro-locations $S_{cruice1}$ and $S_{cruice2}$.

location to its HA proper state, so that a unique automaton is obtained, as it is shown in Fig. 4.20b).

In Fig. 4.21 the HA of the net in Fig. 4.16 is shown.

Once the HA has been built, liveness analysis can be performed.

The definition of *final* given in [TS93] is recalled:

Given an oriented graph $G = (V, E)$ where V is the nodes set and E is the arcs set, the $v \in V$ preset and postset are defined, respectively, by: ${}^{\bullet}v = \{u \in V | W(u; v) > 0\}$, $v^{\bullet} = \{u \in V | W(v; u) > 0\}$, where $W(u; v)$ $(W(v; u))$ is the weight of the arc going from u to v (from v to u). The preset (postset) of a set of nodes is the union of presets (postsets) of its elements.

Let U be a subset of nodes s.t. $U \subseteq G$, U is a *final* iff $(U =$

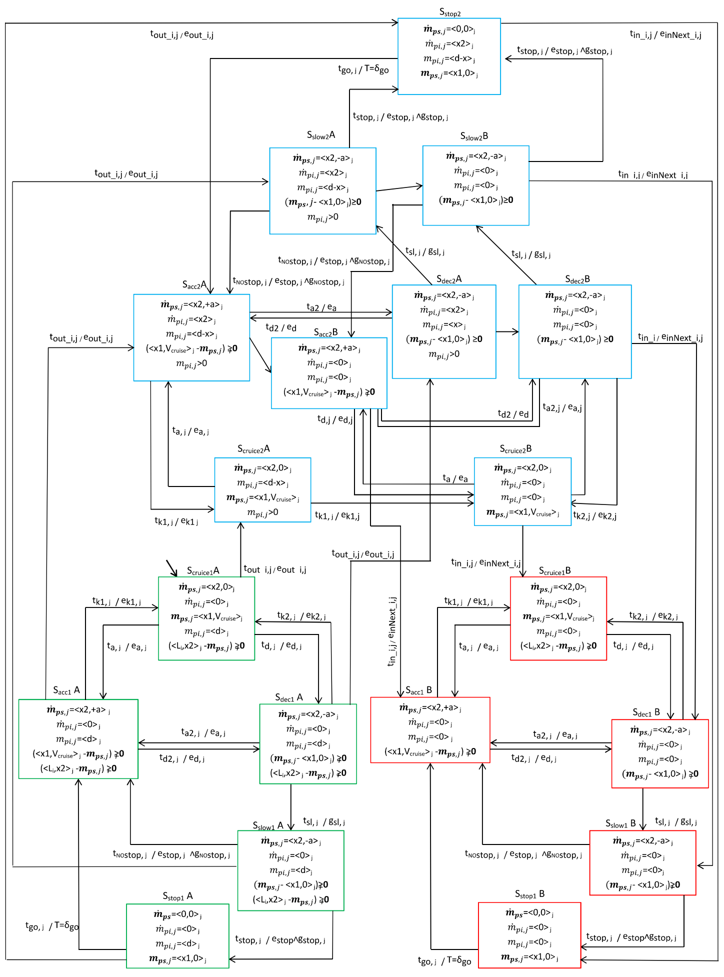


Figure 4.21 Hybrid Automaton of the net show in Fig. 4.16.

$\{v\} \wedge v^{\bullet} = 0$) or the subgraph associated to U, $G(U)$, is strongly connected and $U = succ(U)$, where $succ(U)$ is the successor of U, i.e. the minimal set of nodes s.t. $U^{\bullet} \subseteq succ(U)$ and $succ(U)^{\bullet} \subseteq succ(U)$. In the first case U is called *deadlock*, in the second case, it is called *active final* [TS93]. In words, a final represents a subset of nodes where the net evolution cannot more exit once it enters in.

To test liveness, in accord with its definition given at the beginning of this section, it has to be established if all the transitions are live. At this aim, the following checks, based on the concept of finals usual in discrete PN analysis, can be used:

1. Verify if the HA is strongly connected: if so all the transitions are live and, consequently, the CMHPN system is live.
2. Verify if in the HA there are finals being deadlocks: if so, the net is not deadlock free and, consequently, it is not live (refer to [TS93]).
3. Verify if in the HA there are *active finals*: if so, transitions associated to the arcs of these sub-sets are live.

For synchronized CMHPNs, the firing of enabled transitions is possible only if the associated enable conditions are verified. Supposing that this is true, looking at the HA of Fig. 4.21, it can be seen that the automaton is not strongly connected but there exist three strongly connected sub-sets of nodes, Γ_1, Γ_2, Γ_3 such that a) Γ_1 is the subset of states, drawn in blue in the figure, where $m_{p_i,j} = d - x$; b) Γ_2 is the subset of states, drawn in green in the figure, where $m_{p_i,j} = d$; c) Γ_3 is the subset of states, drawn in red in the figure, where $m_{p_i,j} = 0$;

1. Γ_3 is an *active final.*
2. Transitions t_{in_i} and t_{out_i} , managing vehicles passage between two adjacent zones, are "quasi-live". Indeed, once the system is arrived in Γ_3, transitions t_{in_i} and t_{out_i} cannot fire anymore.

3. All the transitions managing the dynamics switching are live (they belong to strongly connected sub-sets of nodes).

4. In the states S_{stop1A}, S_{stop1B} and S_{stop2}, $x2 = 0$ and all the continuous transitions managing dynamics switching are disabled, so $x2$ cannot change, but the system goes out from these states after that the firing delay δ_{go}, associated to the timed transition t_{go}, is expired.

Points 1) and 2) imply that the net modeling one vehicle running in a single zone is not live, indeed the *active final* is not live (a final is *live* if it includes all the net transitions, see [TS93]) and not all the transitions are live (once the place p_i becomes empty there is no way to change its marking again). However, the models of real ISs are live since they are circular guidepaths made up of at least two adjacent zones. As for example, consider the simple IS layout shown in Fig. 4.22a) consisting in three zones, Z_i, Z_r and Z_k, and a branch point at the end of Z_i. The corresponding CMHPN model is shown in Fig. 4.22b). In Fig. 4.22c) a compact version of its HA is reported: in each one of its nodes only the changes in continuous places marking are reported. In particular, nodes S_{inZ_i}, S_{inZ_r} and S_{inZ_k} group together states belonging to Γ_3 respectively for the zone Z_i, Z_r and Z_k: because of the presence of the adjacent zones, the system can go out from those nodes and it can return in the initial node where $m_{p_i} = d$. Thus, it can be concluded that, even if the single zone model is not live, the IS model is live (unless the enabled conditions are not verified). This is true whatever is the number of zones of the IS.

Points 3) and 4) imply that a vehicle can move along the IS unless its movement is not disabled by the external controller. There are only two motivations to stop a vehicle: i) collision with other vehicles has to be avoided (see Part I - Section IV-B); ii) an interface point with a bay is reached. In the former case a vehicle starts to run again when the previous one is sufficiently far from it. This condition is always reached if $L_c > N_s(L_s+Th)$ with $L_c =$ length of the interface guidepath, $N_s=$ number of vehicles in the interface guidepath, L_s = length of each vehicle and Th minimal space between two vehicles to avoid a collision. So if the number of vehicles has been chosen correctly, a vehicle can always start to move again. In the latter case, a vehicle moves again soon after the swap of the SU or immediately after the stop if the exchange cannot be executed.

Stops of vehicles at the interface bays are ruled by means of the immediate controllable transitions t_{stop} and t_{NOstop} whose firing is controlled by the external controller by means of the external inputs $\boldsymbol{g}stop$ and $\boldsymbol{g}_{NOstop}$ respectively. As more $\boldsymbol{g}_{NOstop} = NOT(\boldsymbol{g}_{stop})$ i.e. when the element $g_{stop,j}$ of the first condition is equal to 1, the correspondent element $g_{NOstop,j}$ of the second one is equal to 0 and viceversa. If it is no more possible to execute an exchange with whatever bay, because the interface points are full and they cannot be emptied, $g_{stop,j} = 0\ \forall j$ and stopping of vehicles at the interface points is disabled. As consequence macro-states S_{stop1A}, S_{stop1B} and S_{stop2} cannot be reached any more and the CMHPN model of the IS is no more live but just not dead: i.e. a livelock occurs.

4.2.12 Deadlock prevention

In Section 4.2.11 has been shown that the CMHPN system modelling an IS, when its controllable transitions are not disabled by a controller, is live. However, during a mission execution, when vehicles have to interface with the bays, blocks can occur. Blocks occurrence prevention in executing a mission is the topic of this section.

The execution of a mission in a complex warehouse system can be seen as the achievement of these subprocesses: a) passage of a Stock Unit (SU) from its storage location to the crane and viceversa, b) passage of a SU from the crane to the crane bay and viceversa, c) passage of a SU from the crane bay to the IS and viceversa, d) passage of a SU from the IS to the picking bay and viceversa. All these subprocesses require the use of resources to be completed. For example, passing a SU from the storage location to the crane requires at least a free place on the crane as well as the

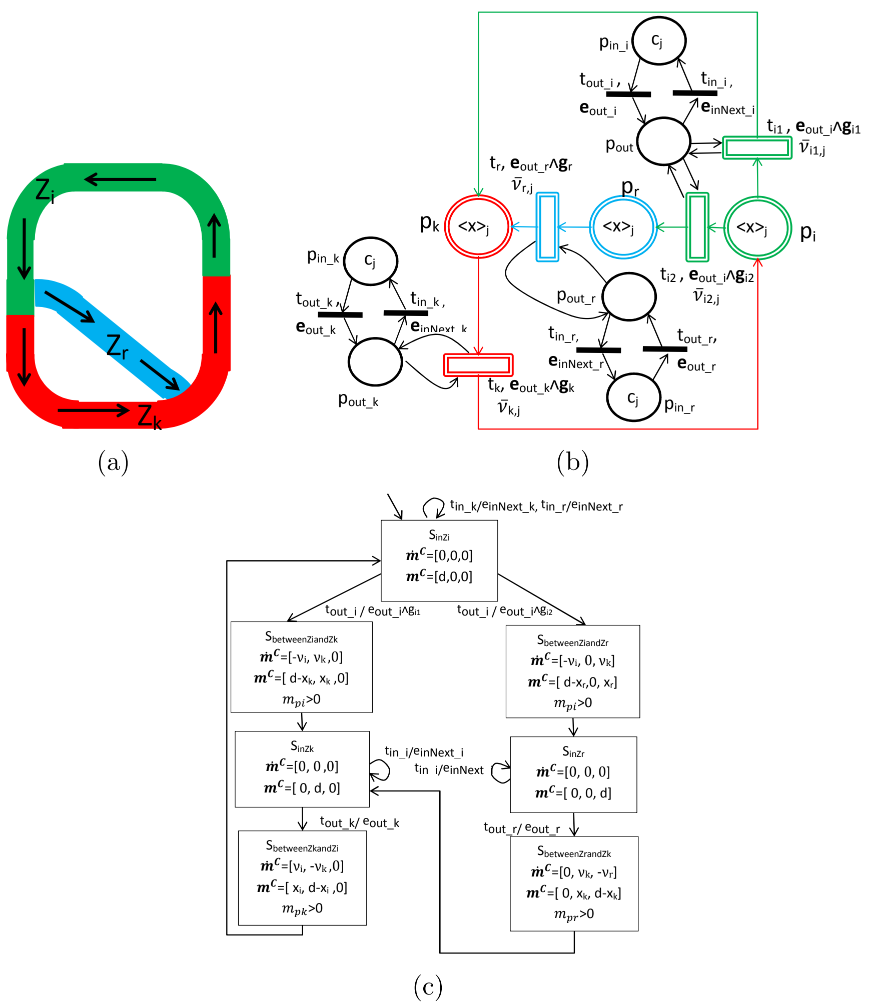


Figure 4.22 A simple IS layout (a), its corresponding CMHPN (b) and a compact version of its associated HA (c).

passage of a SU from the picking bay to the IS requires that there is at least a free vehicle running along the path. While some of these resources are used only by a specific subprocess (free places on the picking bay can be occupied only by SUs arriving from the IS), some others are shared between the subprocesses (free vehicles in the IS can be used both to move a SU from the crane bay to a picking bay and viceversa; free places on cranes can be used to carry SUs from storage locations to crane bays and viceversa).

The presence of shared resources can cause the occurrence of deadlocks, due to circular wait situations. It means that missions cannot be completed anymore and the state of the system, meant as the number of ended missions, is blocked.

4.2.13 Background

In this section some useful notions about S^3PR and about liveness in timed PN are reported: reader can refer to [ECM95] for more details about S^3PR.

Given a PN $\mathcal{N}$,

- A *siphon* is a set of places S s.t. ${}^\bullet S \subseteq S^\bullet$; S is *minimal* when any proper subset is not a siphon.
- A *P-semiflow* $\boldsymbol{Y}$ is a n-dimensional vector s.t. $\forall p \in P$, $Y(p) \in \mathcal{N}$ and $\boldsymbol{Y}^T \cdot \boldsymbol{C} = 0$, where P is the places set of the net and $\boldsymbol{C}$ is the incidence matrix.
- A support of a vector $\boldsymbol{Y}$ is the set $||\boldsymbol{Y}|| = \{p \in P \mid Y(p) \neq 0\}$
- A transition t is *dead* for a reachable marking $\boldsymbol{m}(\tau)$ when t cannot be fired at any state reachable from $\boldsymbol{m}(\tau)$.
- $\mathcal{N}$ is *live* if from each reachable marking it is always possible to fire any transition.

A S^3PR is a state machine (i.e. a net where each transition has exactly one input and one output place) plus a set of places modeling the availability of resources: these places are called *resources*, while the other places are called *holders*. Processes that can be modeled as S^3PR have to satisfy the following constraints:

1. The process describes the set of possible sequences of operations the system has to perform in order to complete a task.
2. It has an initial and a finial state.
3. Choices are allowed but iterations are not.
4. Only one shared resource is allowed to be used at each state in the process. The resource used in a state is released when the system moves to the next state. Two adjacent states cannot use the same resource.
5. Initial and final states do not use resources.

For a S^3PR the following theorem is valid:

Theorem 4.2.1. *Let $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$ be a marked S^3PR, let $\boldsymbol{m} \in R(\mathcal{N}, \boldsymbol{m}(0))$ and let $t \in T$ be a dead transition for $\boldsymbol{m}(\tau)$. Then, there exists a reachable marking $\boldsymbol{m}(\tau)' \in R(\mathcal{N}, \boldsymbol{m}(0))$ and a minimal siphon S such that $\boldsymbol{m}_S(\tau)' = 0$.*

Therefore the following Corollary can be deduced:

Corollary 4.2.2. *Let $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$ be a marked S^3PR. then $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$ is live if and only if, for every S^3PR reachable marking $\boldsymbol{m} \in R(\mathcal{N}, \boldsymbol{m}(0))$ and every (minimal) siphon S, $\boldsymbol{m}_S \neq 0$.*

As consequence of Corollary 4.2.2 and how it has been explained in [ECM95], a way to assure that a S^3PR is live is detecting all the siphons which can become empty (it means siphons that are not support of P-Semiflow), and, for each of them, adding a new place to the net (called *virtual resource*), having a proper initial marking, which goal is to avoid the emptying of the siphon.

Generally speaking, liveness of a TPN system is not enlaced by the liveness of its untimed version. Indeed, let $\langle \mathcal{N}_t, \boldsymbol{m}(0)\rangle$ be a timed PN system and $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$ be the untimed version of the same system (i.e. $\mathcal{N}$ is obtained by replacing an immediate transition to each timed transition). Timing delay associated to the timed transitions generates constraints on the evolution of $\langle \mathcal{N}_t, \boldsymbol{m}(0)\rangle$, which do not exist for the system $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$. It follows that 1) the set of the reachable markings of $\langle \mathcal{N}_t, \boldsymbol{m}(0)\rangle$ is included in the set of reachable markings of $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$, and that 2) the set of possible firing sequences in $\langle \mathcal{N}_t, \boldsymbol{m}(0)\rangle$ is included in the set of possible firing sequences in $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$. The consequence is that, as a rule, the condition that an untimed PN system $\langle \mathcal{N}, \boldsymbol{m}(0)\rangle$ be live for an initial marking $\boldsymbol{m}(0)$ is neither necessary nor sufficient for the timed PN system $\langle \mathcal{N}_t, \boldsymbol{m}(0)\rangle$ to be live for the same initial marking.

However, in case the set of firing sequences of the TPN coincide with the ones of the untimed PN, liveness of the untimed PN implies the liveness of the TPN.

In the following it is shown how to obtain an aggregate model to compute a deadlock prevention policy.

4.2.14 An aggregate view of the CMHPN model

When crane and picking bays are totaly full and they cannot be emptied because all vehicles are busy, a block occurs in the system. It is due to a wrong resources allocation, in particular to a wrong assignment of free vehicles. To design a control policy that solves the problem, a model representing the missions' execution steps, where free vehicles are involved, is needed.

Independently of the warehouse system layout, missions' execution can be divided in a given number of subprocesses that have to be sequentially completed for performing the mission.

Each subprocess describes the handling of a SU from a start point (e.g. its position on a crane, on a vehicle, on a conveyor

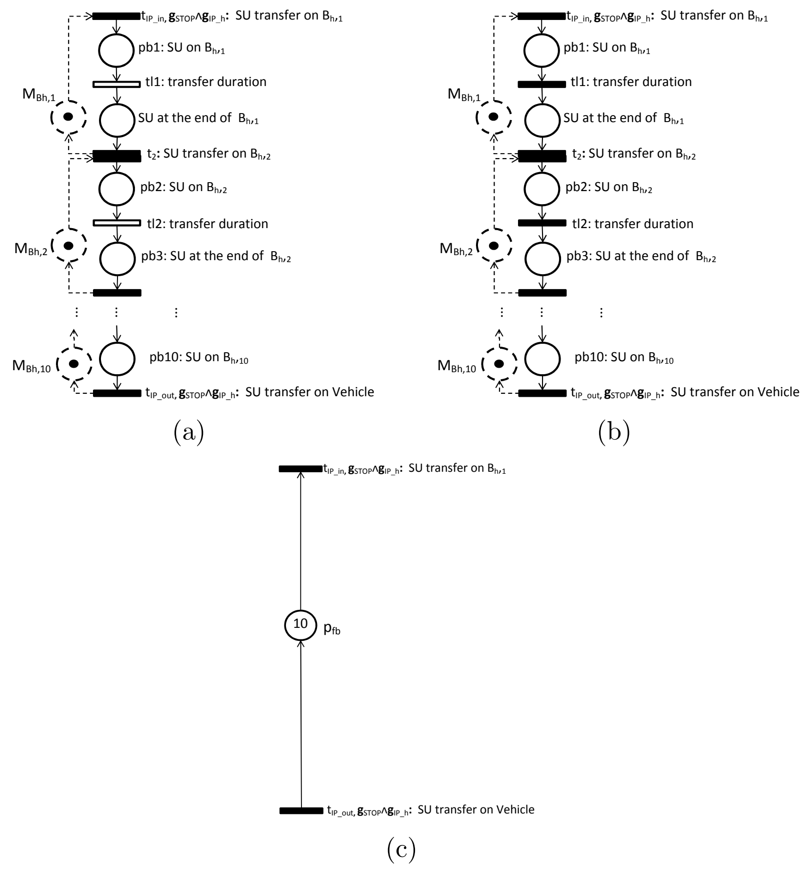


Figure 4.23 Model of the h-th picking bay made up of 10 conveyor belts: CTPN model (a); CPN model (b); picking bay reduced model (c).

belt) to a destination point (e.g. the belt at the interface point between the crane and the crane bay, the interface point between the vehicle and the picking bay). Model of these subprocesses can be obtained from the models of the IS and of the bays using a procedure presented in this subsection.

A bay is made up of a succession of conveyor belts. In Section 4.2.10, the Colored Timed Petri Net (CTPN) model of the conveyor belt interfacing with the IS has been shown. The CTPN model of the whole bay is obtained concatenating as many net modeling a belt as many are the belts composing the bay; as for example, in Fig. 4.23a) a picking bay made up of 10 conveyor belts, that have to move just one SU at time, is shown. Notice that the immediate transitions t_{IP_in} and t_{IP_out} are two controllable transitions that rules the SUs exchanges between vehicles and bays (see Section 4.2.10). From this model, information about SU position in the bay and SU destination can be obtained. An untimed version of the bay model can be obtained replacing all the timed transition with an untimed one as in Fig. 4.23b). It is easy to see that the two systems admit the same firing sequences and consequently, as it has been explained in Section 4.2.13, liveness of the net system in Fig. 4.23a) can be obtained imposing liveness for the net in Fig. 4.23b).

The system of Fig. 4.23b), considered in isolation (i.e. without interacting with the IS), is live unless controllable transition t_{IP_out} is disabled. Indeed each time a new SU is put on the bay (i.e., t_{IP_in} fires), it advances until the last free belt on the bay: this corresponds to the successive firings of the transitions t_k, with $k = 2 \dots 10$ of Fig. 4.23b). The arrival of a SU at the end of the bay is modeled by the presence of a token in place $pb10$: firing of t_{IP_out}, that is possible only if the external controller enable it, free the last belt of the bay allowing the other SUs to advance in the next position. Consequently, if it was occupied, the first belt becomes free again and a new SU can be loaded on the bay (i.e. t_{IP_in} can fire again). If it is no possible to free the last belt (i.e. the firing of t_{IP_out} is disabled) the bay can contain as many SUs as many are the belts it is made up of (10 for the bay of Fig. 4.23b)): after the loading of such a number of SUs, the bay is blocked, i.e. transitions cannot fire anymore and the net system is dead.

With this consideration in mind, for the liveness purposes, a bay can be modeled as shown in Fig. 4.23c), indeed the marking of place p_{fb} indicates the number of free belts on the bay, and when it becomes 0, the firing of t_{IP_in} is no more allowed, while the firing of t_{IP_out} makes the marking of p_{fb} to increase. As for the system net of Fig. 4.23b), if t_{IP_out} is disabled, the net becomes dead after a number of t_{IP_in} firings equal to the number of free places on the bay. Therefore, when transitions of the system of Fig. 4.23c) results to be live (dead) in the net of the aggregate model, also the system of Fig. 4.23b) (and as consequence the one of Fig. 4.23a) is live (dead).

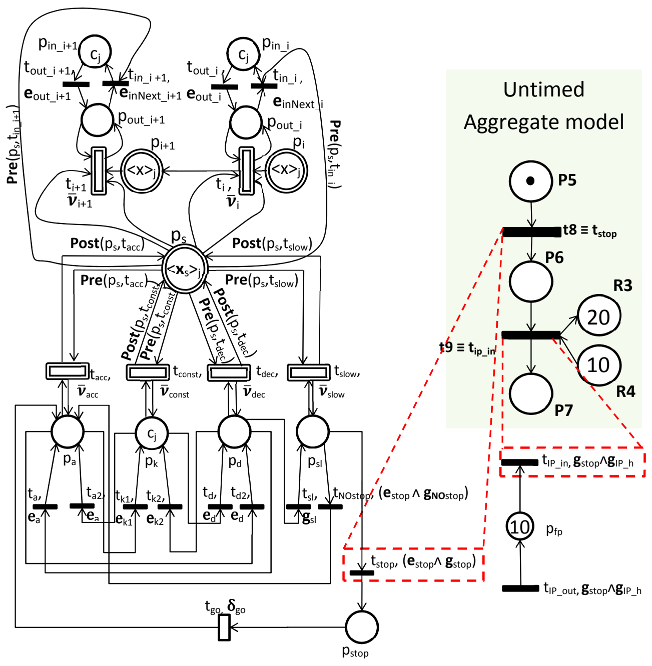


Figure 4.24 Relation between the CMHPN model and the aggregate model of the subprocess "Passage of a SU from the IS to the picking bay".

Once the aggregate model of a bay has been obtained, given a generic warehouse system, to obtain the aggregate model of the other subsystems, the following procedure has to be applied:

a) *Individuate the phases the subprocess can be divided in.*

Each subprocess can be modeled as a net such that: each place of the net represents a phase of the process; actions to be completed to pass from a phase to another one are modeled by transitions. Two sets of places can be distinguished: holders, discussed in this procedure step, and resources, discussed in the next one.

Holders can be further divided in two sets: "dummy places" and "aggregated places". Dummy places have not any connections with the CMHPN model and they only point out that an action is ended and the following one can start.

Aggregated places, instead, are directly obtained starting from the CMHPN: they give information about the presence of an element in a part of the system, without representing the exact zone the element is in and without discriminating between the elements (e.g. looking at the aggregate model it is possible to know that a busy vehicle is running along the IS, but it is not possible to know which is the vehicle).

An aggregate place is added to the aggregate model for each live subnets of the original models describing the passage of SUs from a bay zone to another one or the passage of vehicles from an interface path zone to the next one.

Marking of the aggregate places depends on the one of the CMHPN model. Colors of the live subnets can be divided in two sets: 1) colors associated to elements that are involved in a mission attainment and 2) colors associated to elements that are not involved in the mission. Each time a live subnet of the original model has a marking greater than zero, w.r.t. a color of the first set, a token is added to the relative aggregate place, while, if the color belongs to the second set, no tokens are added. Marking of dummy places depends on the state of the system: a token is added each time the corresponding action is completed. Dummy places marking, as well as aggregate places marking, can be greater than 1.

As for example, suppose to want to obtain the net modeling the subprocess “passage of a SU from the IS to the picking bay”, the following phases can be individuated: SU in IS going to picking bay, SU at the interface point, SU on picking bay. In Fig. 4.24 they correspond to the places $P5$, $P6$ and $P7$, respectively.

Places $P6$ and $P7$ are dummy places: a token in $P6$ indicates that a vehicle is aligned to the interface point and an exchange can occur, while a token in place $P7$ indicates a

SU is moving along the bay. $P5$ is an aggregated place associated to the IS net (as for example, the one represented in Fig. 4.22b)): just one aggregate place is necessary, since to apply the deadlock prevention policy it is not important to know what is the exact zone the vehicle is running in, but it is enough to know that the vehicle is in the IS. A token in $P5$ indicates that a busy vehicle is running in the IS to reach a bay. Notice that, even if free vehicles run along the IS too, their presence does not cause the adding of any token in $P5$. In Fig. 4.24 it is supposed that the color j of the p_s marking is associated to a busy vehicle, so one token is present in $P5$.

b) *Individuate resources involved in the subprocess.*

A resource is something that is needed to perform the actions associated to the subprocess; as for example, to complete "passage of a SU from the IS to the picking bay" at least a free vehicle is needed and at least a free place on the bay is needed: these are the subprocess resources, so two places modeling them have to be added. In Fig. 4.24 they are $R3$ = number of free vehicles and $R4$ = number of free places on the picking bay. As well as for the aggregate places, they are uncolored: the number of tokens of each resource place is equal to the number of copies of that kind of resource that there are in the original model.

c) *Individuate phases transitions.*

Transitions model actions needed to pass from a phase to the following one. Transitions of the aggregate model are uncolored copies of properly immediate discrete transitions of the CMHPN. Each time the original transition fires in the CMHPN, the corresponding transition of the aggregate place fires adding (keeping) an uncolored token in (from) the next (the previous) place.

As for example, consider again the subprocess "passage of a SU from the IS to the picking bay": two actions can be individuated 1) a busy vehicle is arrived at its destination point

2) a SU passing from the vehicle to the bay. In Fig. 4.24, the first action is modeled by transition $t8$, while the second one is modeled by transition $t9$. In particular, $t8$ is the uncolored version of discrete controllable transition t_{stop} of the vehicle model. Transition $t9$ models the SU swap: it fires when transition t_{IP_in} of the bay model fires. Notice that time needed to complete the exchange is neglected: passage of the SU from the vehicle to the bay is considered instantaneous.

In this way an *aggregate view* of the CMHPN model is obtained: in Fig. 4.25 all the subprocesses models are shown.

The firings of the discrete transitions of the aggregate model occurs only when the firings of the associated immediate transitions of the CMHPN/CTPN occurs. As consequence, detecting a deadlock in the aggregate model means to detect a condition that does not allow the CMHPN/CTPN model transitions to fire. In particular, as for the IS model, immediate controllable transition t_{STOP} is the only transition involved in the building of the aggregate model (it rules firing of t_8, see Fig. 4.24). Consequently when a deadlock occurs in the aggregate model, since t_8 cannot fire any more, also t_{STOP} firing is disabled. As it has be seen in Section 4.2.11, when t_{STOP} is always disabled, a livelock is established in the IS model. Then, if the aggregate model is live, then livelocks cannot occur in the CMHPN model of the IS.

4.2.15 Synthesis of deadlock prevention policy

During their handling, SUs pass between adjacent zones carried on vehicles which move in the rail guidedpath, following preassigned unidirectional routes. These passages can be modeled as sequences: in case just choices and confluences are need.

SUs exchanges from vehicles to the interface points of the bays (from the interface points to the vehicles) require that the second ones are not busy: free places at the interface points (free vehicles in the IS) are modeled as resources that have to be acquired during the missions' execution. Whit this consideration in mind, usually it is possible to model each subprocess of a mission as an S^3PR.

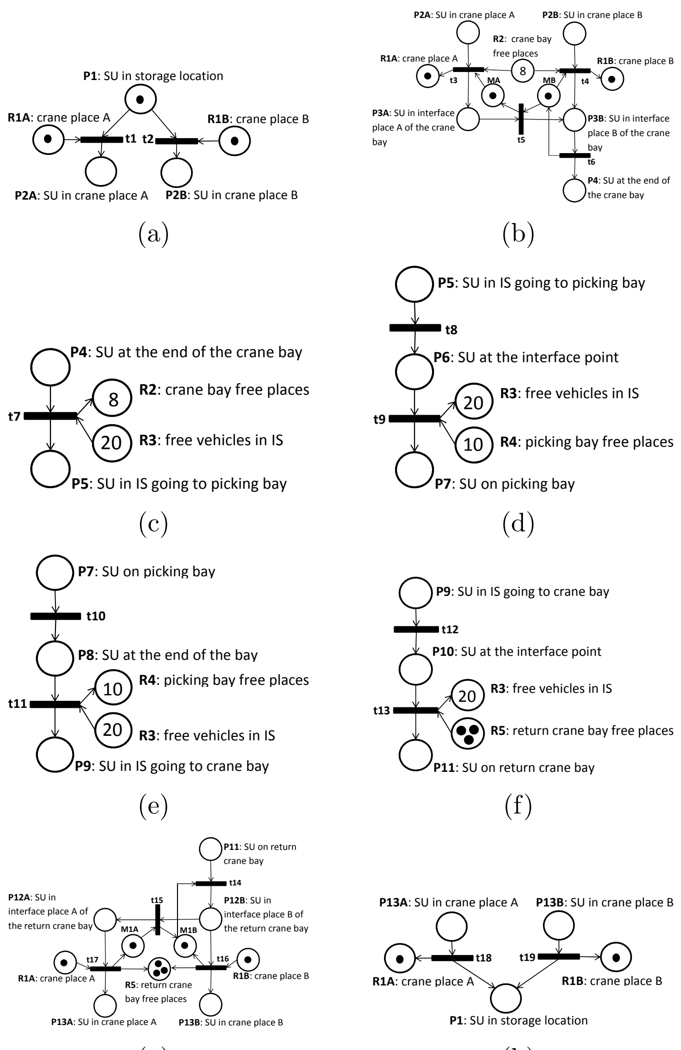

P1: SU in storage location
R1A: crane place A
R1B: crane place B
t1
t2
P2A: SU in crane place A
P2B: SU in crane place B
(a)
P2A: SU in crane place A
P2B: SU in crane place B
R2: crane bay free places
R1A: crane place A
R1B: crane place B
t3
t4
MA
MB
8
P3A: SU in interface place A of the crane bay
P3B: SU in interface place B of the crane bay
t5
t6
P4: SU at the end of the crane bay
(b)
P4: SU at the end of the crane bay
R2: crane bay free places
t7
R3: free vehicles in IS
P5: SU in IS going to picking bay
8
20
(c)
P5: SU in IS going to picking bay
t8
P6: SU at the interface point
R3: free vehicles in IS
t9
R4: picking bay free places
P7: SU on picking bay
20
10
(d)
P7: SU on picking bay
t10
P8: SU at the end of the bay
R4: picking bay free places
t11
R3: free vehicles in IS
P9: SU in IS going to crane bay
10
20
(e)
P9: SU in IS going to crane bay
t12
P10: SU at the interface point
R3: free vehicles in IS
t13
R5: return crane bay free places
P11: SU on return crane bay
20
(f)
P11: SU on return crane bay
t14
P12A: SU in interface place A of the return crane bay
t15
P12B: SU in interface place B of the return crane bay
M1A
M1B
t17
t16
R1A: crane place A
R5: return crane bay free places
R1B: crane place B
P13A: SU in crane place A
P13B: SU in crane place B
(g)
P13A: SU in crane place A
P13B: SU in crane place B
t18
t19
R1A: crane place A
R1B: crane place B
P1: SU in storage location
(h)

Figure 4.25 PNs of the subprocesses composing a mission.

Figure 4.26 Minimal Siphons of the net shown in Fig. 4.28.

Siphon	Places
S_1	$\{P13A, P13B, R1A, R1B, R2, R3, R4, R5\}$
S_2	$\{P5, P6, P13A, P13B, R1A, R1B, R2, R3, R5\}$
S_3	$\{P11, P12A, P13A, R1A, R2, R3, R4, R5\}$
S_4	$\{P5, P6, P11, P12A, P13A, R1A, R2, R3, R5\}$
S_5	$\{P9, P10, R3, R4\}$

Figure 4.27 Marking, Pre-set and Post-set of the Virtual Resources.

Place	Initial Marking	$^\bullet V_i$	$V_i^\bullet$
V_1	42	$\{t16, t17\}$	$\{t1, t2\}$
V_2	32	$\{t7, t16, t17\}$	$\{t1, t2, t11\}$
V_3	41	$\{t16, t13\}$	$\{t1, t4, t15\}$
V_4	31	$\{t7, t16, t13\}$	$\{t1, t4, t15, t11\}$
V_5	29	$\{t11\}$	$\{t7\}$

The advantage to obtain the aggregate model as an S^3PR is that in the literature there are several papers presenting policies to ensure liveness for this class of nets: in this work the deadlock prevention policy of [ECM95] has been chosen to be applied. However, a deadlock prevention policy can be applied also when the obtained aggregate model does not belong to such a class continuing assuring the liveness of the CHMPN system.

In Fig. 4.28, the net describing the behavior of a system made up of 1 crane bay, 1 picking bay and an IS with 20 vehicles is shown. Resources involved in the process are the number of free places on the crane, the number of free places on the bays and the number of free vehicles running along the IS. The net is not live, indeed there are 5 minimal siphons, which are not support of P-Semiflows (they are reported in Table 4.26). To make this net live, 5 virtual resources have to be added: their initial marking, their preset and their postset are reported in Table 4.27.

From a practical point of view, in this example, the consequence of the adding of the virtual resources is that a vehicle is always reserved for the emptying of the picking bay and the passage of a SU from the crane bay to the IS is prevented until at

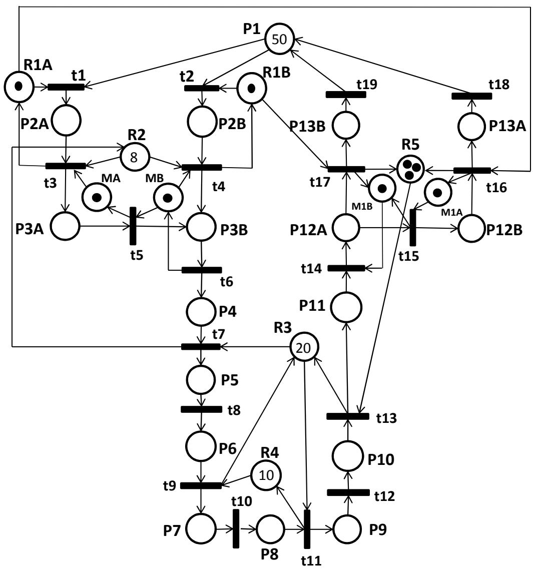


Figure 4.28 S^3PR modeling a system made up of 1 crane bay, 1 picking bay, an IS with 20 vehicles and 50 SUs to move.

least a free place is present in the subsystem IS+picking bays.

4.3 Conclusion and Future researches

The automated warehouse models presented in this dissertation allows to represent complex systems in compact way. Modularity, compactness and reconfigurability are some of the advantages of the CMHPN model presented: it can be adapted at several layouts just adding or removing elementary zones. As more, the adding of a new bay just correspond to the adding of new stop conditions, while adding (removing) colors allows to increase (decrease) the number of vehicles in the IS. If a new route is added, it can be connected at the IS just introducing the arcs linking it to the guidepath model. Future research can extend the approach for PNs to fault analysis and diagnosability [BCD12] to CMHPNs. The model can be used:

1. **To make offline system performance analysis.** Using the simulation environment the model can be used to test the performance of a certain control strategy. The controller manages SUs destinations and it uses this information to decide where and when vehicles must stop. A typical time-performance measure, for the activities of such a system, is the so-called "Makespan" defined as the time required to the system to complete a set of missions. In contexts where missions have very different time durations it can happen that there is one last resource involved in a long mission, while quite all the resources have completed their jobs, thus being available for new jobs. In this case the makespan behaves like non-balanced measure thus not capturing the goodness of a control action. For this reason the "Average of the Ending Times" of resources can also be used. However, other performance measures can be considered. Thanks to the clear interface between the model and the control action, any control strategy can be efficiently tested.

2. **To make online parallel forward simulations.** The model can be used also as a tool to make online simulations: it is possible to recur at forward simulations each time it is possible to choose between different possibilities to complete a mission (i.e. more than one SU can be retrieved, a SU can be stored in more than one storage location, a free vehicle is required by more than one bay at the same time, etc). Using online simulations, starting from the current system state (number of active missions, state of resources, states of the warehouse) and looking at a prefixed short horizon, the best solution can be determined: parallel simulations using different strategies are executed and the local optimal solution is found. Any time a choice occurs, a new set of forwards simulations starts.

3. **To decide strategies to prevent potential deadlocks.**A structural analysis of the model can be made to individuate probable deadlocks (i.e. waiting conditions that occur be-

cause of wrong resources sharing policies). In addition, the model can be used to synthesize deadlock prevention strategies that can be simulated to observe the system behavior.

Using a CMHPN model compared to use a discrete PN one, implies the following advantages:

- A more compact representation of the system: using a discrete PN model, N discrete places to model a zone of length L have to be used to know vehicle positions with a desired precision L/N; it means that to have a greater precision the number of places has to be increased (e.g. if $L = 10m$ and $N = 2$ a precision of $5m$ is obtained while if $N = 10$ precision raises to $1m$). With the hybrid model just one place is always used whichever is the length of the zone and whichever is the desired precision.

- Any possible configuration of the real system is represented: using the hybrid model the vehicle position can assume any real value between 0 and the length of a route (each vehicle can assume any real position), while it can assume just a finite set of values if a discrete event model is adopted. Thanks to such a continuous nature of the position, an increment in system performances can be obtained, since the controller knows the system's state with a precision better than the one achievable with a discrete event system.

- Switching between continuous time phenomena, as acceleration, deceleration or constant velocity, can be represented.

As more, future researches want to show how the use of the Colored Modified Hybrid Petri Nets, can be extended also in different contest, as the one of unmanned aerial vehicles, as shown in [BCCG12] where CMHPNs have been used to obtain compact models for online monitoring of aerial service robots.

Chapter 5

CMHPNs Simulator

In this chapter, a simulator for CMHPNs, developed by the Automatic Control Group of University of Salerno, is presented. It allows to design and simulate not only the net, but also the controller, allowing the user to create models *ad hoc* for several kinds of systems. The simulator has been obtained as an evolution of PNetLab, see [pne] and [BCC07b], a freeware simulation and analysis tool for PN/CPN model. It presents a graphical user interface to draw the net and the advantage to allow the definition of the controller as a standard C/C++ program, implementing logical predicates. All the features of the previous version have been held (i.e. it can be used also to work with basic PN/CPN): the important new feature is that it allows simulating of CMHPNs continuous part with variable sample rate. The simulation tool will be used in the Chapter 5 to evaluate the performance of a real warehouse system.

5.1 Simulation algorithm

At each instant τ_k, the current state $\boldsymbol{m}(\tau_k)$ of the system, starting from its previous state, $\boldsymbol{m}(\tau_{k-1})$, i.e. the state the system moved in, at the observation instant τ_{k-1}, are given as output by a simulation. To do it, the simulator has to follow the evolution both of the discrete and the continuous component of the systems.

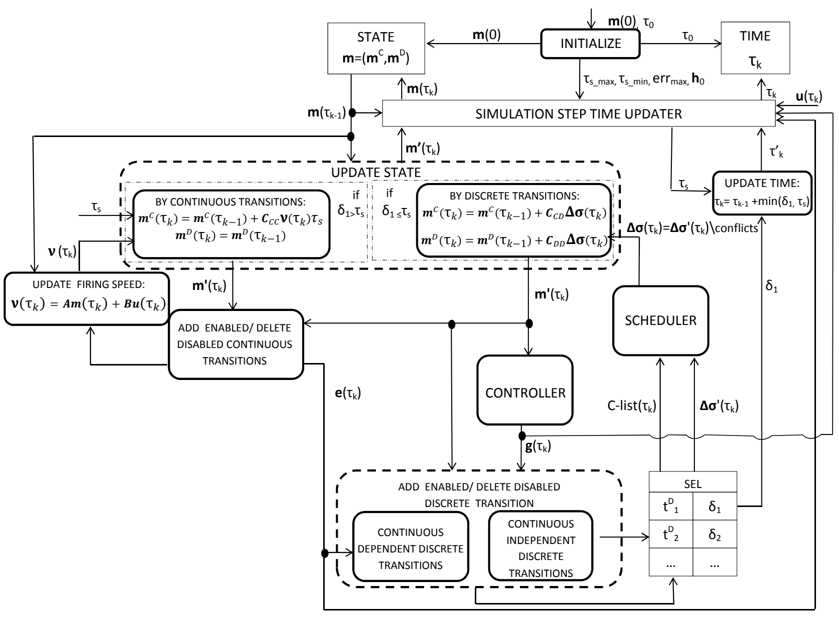

STATE
m=(m^C,m^D)
m(0)
m(0) τ0
INITIALIZE
τ0
TIME
τk
m(τk)
τs_max, τs_min, errmax, h0
τk
u(τk)
m(τk-1)
SIMULATION STEP TIME UPDATER
m'(τk)
τ'k
UPDATE STATE
τs
UPDATE TIME:
τk= τk-1 +min(δ1, τs)
BY CONTINUOUS TRANSITIONS:
m^C(τk) = m^C(τk-1) + C_CC v(τk)τs
m^D(τk) = m^D(τk-1)
if δ1>τs
if δ1 ≤τs
BY DISCRETE TRANSITIONS:
m^C(τk) = m^C(τk-1) + C_CD Δσ(τk)
m^D(τk) = m^D(τk-1) + C_DD Δσ(τk)
Δσ(τk)=Δσ'(τk)\conflicts
v (τk)
UPDATE FIRING SPEED:
v(τk) = Am(τk) + Bu(τk)
m'(τk)
ADD ENABLED/ DELETE DISABLED CONTINUOUS TRANSITIONS
m'(τk)
SCHEDULER
δ1
CONTROLLER
e(τk)
C-list(τk)
Δσ'(τk)
g(τk)
ADD ENABLED/ DELETE DISABLED DISCRETE TRANSITION
CONTINUOUS DEPENDENT DISCRETE TRANSITIONS
CONTINUOUS INDEPENDENT DISCRETE TRANSITIONS
SEL
t^D_1 δ1
t^D_2 δ2
... ...

Figure 5.1 Architecture of the hybrid systems simulator.

While changes in the discrete part happen only when particular events occur, continuous state changes nonstop. So, to determine changing in the discrete state, the simulator has to know what events occurred (i.e. what discrete transitions fired), while it has to define a sample rate (τ_s) to update the continuous state. The simulator's behavior (that will be described in the following) is shown in the Fig. 5.1.

The net can move in a new state for three reasons: i) at least a discrete transition fires; ii) it is expired a time equal to τ_s, during which at least a continuous transition has been enabling; iii) it is expired a time equal to τ_s, during which at least a continuous transition has been enabling and at least a discrete transition is fired. At the instant τ_k, an enabled discrete transition t^D fires only if $\tau_k = \tau_{k-1} + \delta_{t^D}$, where δ_{t^D} is the t^D firing delay . At each step of simulation, the list of the enabled discrete transitions and their firing delays is contained in the Scheduled Event List (SEL) [CL08]. After every firing of a discrete transition, the list has to be update: after the reaching of the new state $\boldsymbol{m'}(\tau_k)$, calculated in according with (2.7) and after the controller has generated the control events $\boldsymbol{g}(\tau_k)$, new enabled transitions are added to the SEL, while the disabled ones are deleted. The outputs obtained from the SEL are: the firing vector $\Delta\boldsymbol{\sigma}'(\tau_k) = \boldsymbol{\sigma}'(\tau_k) - \boldsymbol{\sigma}'(\tau_{k-1})$ that has all components equal to 0 but those associated to enabled discrete transitions are equal to 1; the list C-list(τ_k) of conflicting transitions (i.e. set of enabled transitions where only one transition can effectively fire) contained in $\Delta\boldsymbol{\sigma}'(\tau_k)$. The scheduler solves the conflicts according to a certain strategy and produces a new firing vector $\Delta\boldsymbol{\sigma}(\tau_k)$ that is used to update the marking.

Evolution of continuous part is simple: at each sample instant, the simulator, looking at the state of the net, obtained using (2.7), determines which are the enabled transitions and calculated their firing speed.

As it has been discussed in Chapter 2, Section 2.2, changes in continuous marking can influence enabling of discrete transitions and viceversa. Discrete transitions that are not influenced by changing on continuous state will be called "*continuous independent*"; viceversa, discrete transitions that can be enabled (disabled) after a changing on continuous state will be called "*continuous dependent*". When an update of continuous state occurs, the value of the internal condition $\boldsymbol{e}(\tau_k)$ is calculated for each continuous dependent discrete transition and, if it is necessary, the contents of the SEL are changed. After the firing of a discrete transition, the new state is observed to determine if a change in the continuous dynamic occurred: if so the new firing speeds, to use in the next sample instant, are calculated.

After every simulation step, the next observation time is calculated as:

$$\tau_k = \tau_{k-1} + min(\delta_1, \tau_s)$$

where, since the SEL is sorted on a smallest-scheduled-time-first, δ_1 is the lowest firing delay of the enabled discrete transitions and τ_s is the sample rate.

With the goal of improving the simulator performances, a not fixed sample time has been chosen. Indeed, the simulation time of a continuous time system is improved if a variable simulation

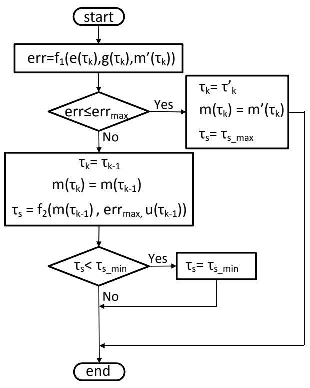


Figure 5.2 Simulation step time update.

step time is used. However, in a hybrid context the updating of simulation step time must consider also the interaction between discrete and continuous parts. In words, if a too large step time is used, boundary conditions which depend on the values of continuous time variables could be detected with a not negligible error. Since, enabling conditions of non autonomous discrete transitions can depend on these boundary conditions, a function f_1 evaluates this error (see Fig.5.2), and if a threshold error is exceeded, the step time is decreased according to a function f_2.

Functions f_1 and f_2 are written by the user and they depend on what is the system whose behavior has to be simulated: i.e. for the toy example presented in Section 4.2.8, a possible choice for f_1 can be $f_1 = g_{sl} \cdot (x1 - x_f)$, where x_f is the *slow down* point. Because of the sampling, generally the simulator cannot detect exactly when the mass is in x_f, so the start of the deceleration phase takes place when $x1 > x_f$. Consequently the mass will stop after the designed point x_{d_i} and the distance between the real stop point and the desired one is as greater as f_1 is bigger. Choosing the sample rate in the manner that $f_1 \leq err_{max}$ assures the mass exceeds the desired stop point at most of a prefixed quantity, function of err_{max}.

5.2 Closed-loop system simulation

The control of automated warehouse systems is usually implemented in two levels [ABCC05]. A lower level is dedicated to the control and coordination of the plant devices (conveyors, buffers, cranes, shuttle, etc.), essentially to enforce logical constraints such as deadlock freeness, mutual exclusion in using common resources, etc. An upper level is dedicated to the performance optimization.

At the lower level, supervisory control theory [RW89, CAD99] can be used to design an agent, called supervisor, that disables a controllable transition when a logical constraint could be violated. A supervisor can be modeled by a PN (CPN), or more in general by an algorithm [BCG07, BCC07a]. When the supervisor is modeled by a net system, the closed-loop simulation can be carried out directly in PNetLab. Indeed, a boolean variable whose value can be set by the supervisor is associated to each controllable transition (to each controllable transition color) in the plant model. Moreover, PNetLab allows the integration of a PN/CPN/CMHPN model with a standard C/C++ control algorithm thus allowing closed-loop simulations of supervised systems in the most general case. It is also easy to integrate external tools for the solution of programming problems in order to implement very sophisticated supervisory control algorithms. As for the controller implementation, if the supervisor is a PN, CPN, it can be implemented on a PC/PLC using one of the approaches in [FCSS99a, FCSS99b] or [BC07] or an object-oriented approach using UML [BCD09b]. If it is an algorithm, the C/C++ code tested on the PNetLab platform can be directly implemented on a PC where the access to the field signal is available.

The closed-loop system (plant plus supervisor) seen from the upper level can result in a non-deterministic system since conflicts may occur, i.e. more than one transition is enabled but only one can effectively fire. Since real systems must be deterministic, a scheduling strategy to be implemented at the upper level is required in addition to the supervisor. To this purpose the tool recognizes such conflicts. A scheduler solves them according to a certain policy. The scheduler can be directly implemented in PNetLab or the simulation engine, without the graphical interface, can be linked to a program implementing such a scheduler. In both cases, the simulation of the closed-loop system plus the scheduler is achieved. Moreover, thanks to the possibility to be linked externally, PNetLAB can be used to develop schedulers based on look-ahead and what-if techniques. At this aim, the scheduler can also impose a given state to the PN model of the plant. The scheduler tested by PNetLab is an algorithm, or a set of dispatching rules that can be implemented on a PC/workstation connected with the machine where the supervisor is implemented too.

PNetLab is available free of charge to interested readers and is downloadable from http://www.automatica.unisa.it.

Chapter 6

Case study

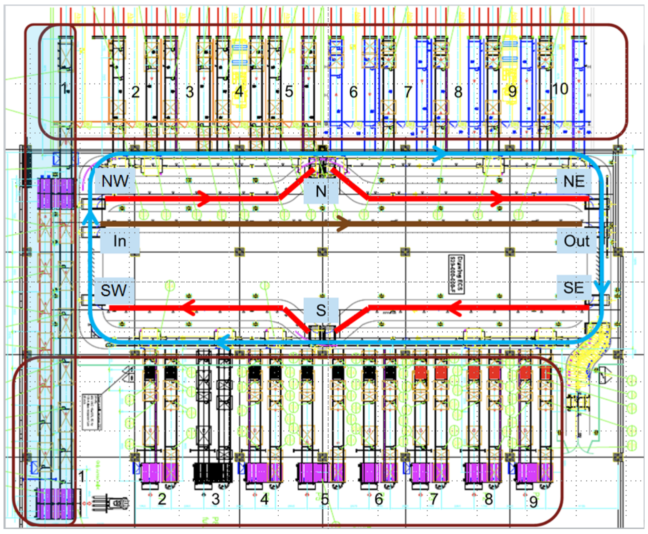


Figure 6.1 Layout of a real warehouse and its zones: crane bays (at the top), picking bays (at the bottom), direct link between aisle1 and bay1 (at the left), carousel routes and branch points (in center).

In this chapter results obtained simulating the behavior of a real warehouse system are reported. A comparison between performance obtained using the discrete CTPN model and the CMHPN, introduced in Chapter 4, is presented.

6.1 Plant description

The warehouse is divided in 10 aisles, each served by one crane. On both sides of each aisle there is a storage rack composed of 17 rows and 47 columns. The whole system is made up of 15980 locations. Each location can store one or more SUs, depending of their dimensions (mm x mm): EURO1, 800x1200; EURO3, 1200x1200; MINIJUMBO, 1600x1200; JUMBO, 2500x1200. In the warehouse 47940 EURO1 SUs can be stored. The crane moves along horizontal (vertical) axis with a maximum speed $4m/s$ ($0.8m/s$).

The picking area consists of 9 bays where SUs are emptied by human operators. Aisle1 and bay1 are directly connected by a system of conveyors and they can move only Jumbo SUs. The other 9 aisles are connected with the remaining 8 bays by a carousel (Fig. 6.1) where up to 23 vehicles can be moved. Carousel presents several parts: an "outer ring" (blue path in Fig. 6.1) where the carousel interfaces with cranes or picking bays; an "inner ring", connected with the "outer ring" by 6 switching points (named NW, N, NE, SE, S, and SW in Fig. 6.1), with 4 alternative routes (from NW to N, from N to NE, from SE to S, from S to SW) vehicles can use to arrive at a specific set of bays; a "rail of parking" (brown path in Fig. 6.1) where vehicles can be stopped when they are not useful.

Vehicles can move in continue way with constant speeds or with a uniformly accelerated motion, depending on the part of carousel they are going through (on linear rail they can move with maximal speed, but they must slow down in climb/dismount or in bend). As more if the distance between the following vehicle goes under a fixed threshold the speed is slowed down with constant acceleration.

In the warehouse, full SUs are transported in and out from the aisles, to be (partially) emptied at the picking bays. The replenishment of the items in the structure represents the 10% of the warehouse activity, and it is performed during the night, in the manner it does not influence warehouse performance. For this reason, at the start of any working-day, it can be assumed all the items necessary to satisfy the orders are already present in the warehouse.

6.2 System Performance

In the literature, the automated warehouse systems performance optimization usually reduces only to the crane subsystems one, since the crane subsystem is considered to be independent of the performance of the IS. This is true when a vehicle requires a negligible time with respect to the crane mean cycle time to reach the crane bay from a picking bay.

As for cycle time performance, automated warehouse systems are usually sized according to FEM 9.851(see [Eur03]). More in detail, the performance of each aisle is estimated by computing the time needed to perform a dual command cycle, named FEM cycle, consisting of storing a SU and retrieve another one in specific points of storage rack. The coordinates of these points are obtained from rack geometry. A similar approach could be used also in general case, where the cranes in a single cycle can store and retrieve more than one SU (in the case study up to 2 SUs).

The estimated average cycle time for each crane of the considered warehouse is 50 SUs/h (500 SUs/h for the overall system), but, as it will be shown later, this is not realistic. Indeed, this could be acceptable when the carousel effects can be neglected.

In the real warehouse considered here, the size of the system makes the crane subsystem performance to be dependent on the carousel ones. In this case, an effective way to estimate warehouse performance is simulating its behavior, as done in this thesis.

In case of simulation, the activity of each crane reduces to a single timed transition whose firing delay is fixed after the cycle has been created. When two SUs to be stored in a certain aisle reach the crane bay, a crane cycle is built by the controller properly adding two pickings so that the crane performance is optimized (see [ABCC05]). Details on this are omitted here, since this is not the focus of this dissertation and it is a well known problem in the literature.

Warehouse performance optimization can be obtained devised several control laws, also very complex, both for the crane system than for the IS. As for example, dispatching rules can be adopted as suggested in [ABCC05] to obtain a short-term optimization, that usually has the objective to minimize the time to complete a little number of picking or storage missions and it is based on the current state of the system.

Performance optimization is out of the scope of this dissertation. For this reason the following control strategies, actually adopted in the real plant, have been used in the simulation both with the discrete and with the hybrid model:

- Storage Control Policy

  For each SU waiting for a storage:

  1. For each aisle, create the associated storage queue (i.e. the list of items to be stored in the aisle free locations)
  2. Create a list of free locations having the same size of the SU.
  3. Select the free location whose aisle has the shorter storage queue.
  4. Change in "Reserved" the state of the selected location.

- Vehicle Assignment Policy

  Create a list of SU waiting for a storage. For each SU waiting at an interface point between crane/picking bays and IS:

  1. For each free vehicle compute the time it will take to carry on the SU.
  2. Select the closet vehicle respect the SU.
  3. Set the bay interface point as the destination of the selected vehicle.

6.3 Simulations

Machine used to carry out the simulations is a workstation equipped with 2 Quad-Core CPU *Intel*® *Xeon*® E5530@2.40GHz, 12GB RAM, and Windows 7 Professional 64-bit operative system. PNetLab has been used as simulation environment (see [pne] and [BCC07b]) for simulating the CTPN model, while the extended version for CMHPNs of PNetLab, presented Chapter 5, has been used as simulation tool for CMHPN model.

6.3.1 Simulations using a discrete model

First the simulation of the CTPN model of the case study warehouse is presented: it is obtained properly connecting the CTPN modules introduced in Chapter 4. The firing delay of timed transitions depends on the length of the zone and represents the time necessary to the vehicle to cross the zone. For the sake of simplicity, acceleration and deceleration have been neglected, since the modeling of such continuous dynamics requires the implementation of a too complex discrete model. Hence, vehicles can have only two speeds, $v = 0$ when they are stopped and $v = V_{max}$ when they are moving.

Simulations with different values of system parameters have been made, showing that crane and IS are not independent and the whole system has a nonlinear behavior. In particular, starting from an initial warehouse occupation of about 80%, the behavior of the real warehouse during a single working-day including 6000 missions has been simulated varying vehicle speed. The missions required SUs equally distributed in each aisles. Simulation have shown that:

A. Varying of vehicle speed influences the crane subsystem performances in a nonlinear way. In particular, named crane occupation time the percentage of time a crane remains busy during a simulation, it depends on how much time the carousel spends to deposit on crane bays SUs that must be stored in the aisles. Indeed, if there are not SUs on the bay, a crane must wait and as consequence the crane occupation time decreases;

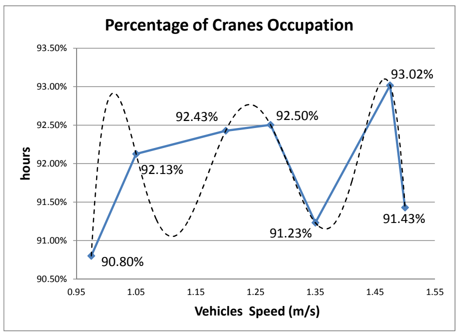
B. A typical time-performance measure, for the activities of such a system, is the so-called “makespan” defined as the time required to the system to complete a set of missions. Studying variation of makespan depending on vehicle speed, it has been shown that an increment of vehicle speed does not produce mandatorily an increment of the makespan: a nonlinear behavior is obtained, due to the stop and go events of the vehicles that halt to load (unload) SUs at the interface zones;

C. Increasing the number of vehicles in the carousel does not imply a consequently increment of the performance. Indeed, after a threshold (20 vehicles) adding other vehicles chokes up the carousel and gets worse the performances. This result confirms the nonlinear behavior due to the interface zones where the discrete event dynamic is relevant.

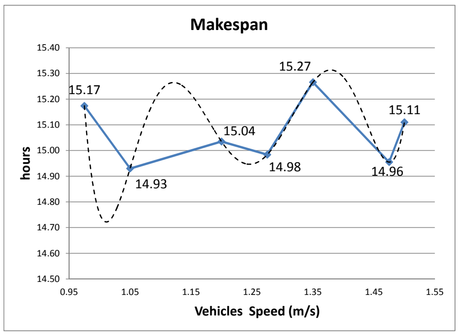
Successively, an aggregate model of the CTPN model has been obtained using the same algorithm shown in 4.2.14 for the CMHPNs, with the difference that, in this case, the marking of the aggregate places has been obtained starting from the marking of live CTPN subnets in stead of CMHPN ones.

Starting from the aggregate model, the same deadlock prevention control policy of Section 4.2.12 has been implemented using the approach presented in [BC07] and the set of the previous simulations has been repeated.

Performances are shown in Fig. 6.2 and Fig. 6.3. Previous considerations are still true. Notice that with 20 vehicles in the carousel, a makespan of 15.09 hours is obtained that corresponds to an average cycle time of 398 SUs/h. As more, comparing these results with the previous one, it can be seen how the deadlock prevention policy makes the system slower. Indeed when a bay becomes completely busy, a vehicle is reserved to be used only to empty that bay. As consequence the available vehicles number decreases and the makespan increases.



(a)



(b)

Figure 6.2 Variation of the system performances with 20 vehicles in the carousel, using a discrete model, increasing vehicles speed: (a) percentage of cranes occupation; (b) makespan .

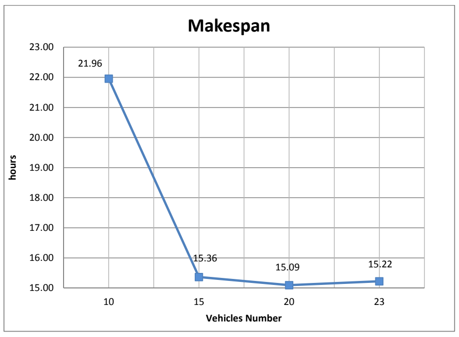


Figure 6.3 Variation of makespan, using a discrete model, increasing the number of vehicles, with vehicle speed $V_{max} = 1.5m/s$.

6.3.2 Simulations using a hybrid model

The use of a CMHPN model, instead of a discrete one, leads to more accurate results with the same computational efforts.

In this section, simulations of a CMHPN model of the case study warehouse are presented: the model has been obtained using the CMHPN modules presented in Part 1 - Section IV;

Two sets of analysis have been made:

1. valuation of the dependence of warehouse system performance on the value of simulator parameters (as sample time and kind of simulator used);

2. valuation of the dependence of warehouse system performance on the value of system parameters (as number of vehicles running in the carousel and maximal vehicles speed).

In the first set, simulations using fixed and variable sample time simulator has been executed.

Results obtained with a fixed sample time τ_s are reported in Fig. 6.4a) and in 6.5a): better performances are obtained decreasing τ_s. Indeed, since the state of the system is frozen between two sample times, the controller knows the vehicle position with a precision depending on the number of the samples: as bigger is the number, as greater is the precision. That allows the controller to better manage the interactions between vehicles and bays.

Using a variable sample time, at particular vehicles configurations, the allowance by which a target position is considered reached does not depend on the value of the sample rate but it is fixed a priori. Indeed, with a variable sample time simulator, τ_s can vary in the range $\left[\tau_{sMin}, \tau_{sMax}\right]$. At any simulation step, τ_s is set equal to τ_{sMax} but it is decreased any time the system state needs to be observed with a better precision (e.g. the error between a target position and that of a vehicle, obtained with the current τ_s, is greater than a fixed threshold). In that case, the previous state of the system is re-established and a new value of τ_s is calculated, using an appropriate function, depending on the nature of the system. In Fig. 6.4b) and 6.5b) results obtained

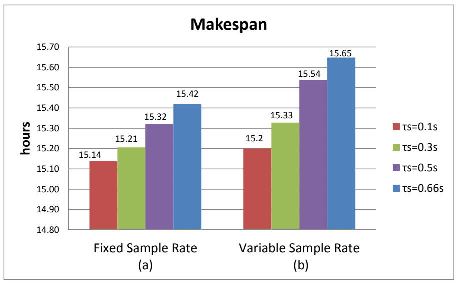


Figure 6.4 System performance on a working-day with 20 vehicles and: a) different fixed sample rates; b) different variable sample rates.

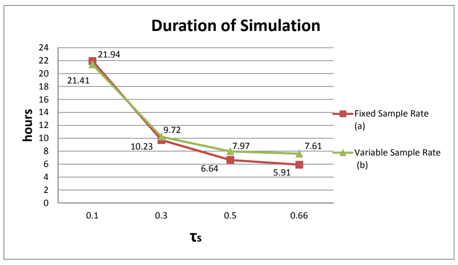


Figure 6.5 Duration of the simulation increasing fixed and variable sample rate.

with $\tau_{sMin} = 0.05s$ and $\tau_{sMax} \in \left[0.1, 0.66\right]s$ are reported: performances are a little bit minor than those obtained with the fixed sample rate simulator. It is because with fixed τ_s target position is considered reached with a great allowance (e.i. about $1m$ in the case of $\tau_s = 0.66s$) while in the second case, for each sample time, tolerance is fixed at $0.15m$.

With the second set of analysis, nonlinear makespan behavior respect to the variations of the number of vehicles running in the carousel has been confirmed (see Fig. 6.6): 20 vehicles is the threshold value after that makespan increases again. Notice that also in this case the maximum makespan corresponds to an average cycle time, 398 SUs/h, that is minor than the estimate one (500 SUs/h). As more, also the influence of vehicles speed on crane performance is confirmed: cranes percentage occupation and makespan, obtained varying vehicles speed, are reported in Fig. 6.7a) and Fig. 6.7b), respectively.

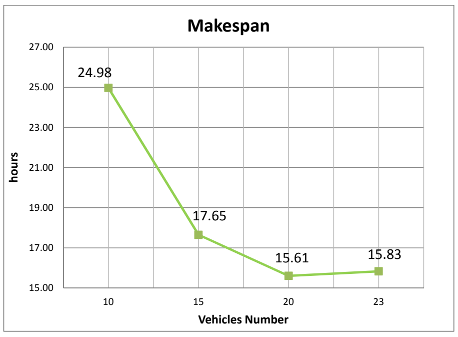
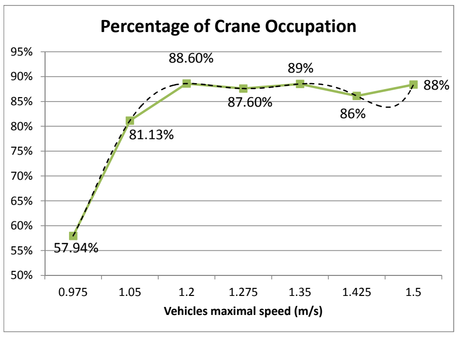
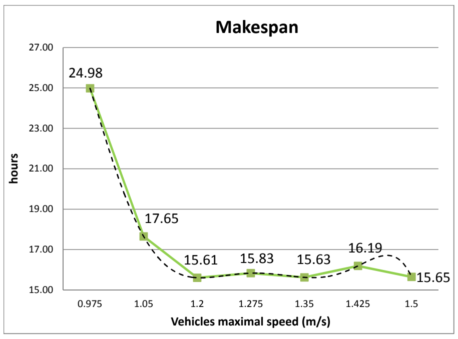


Figure 6.6 Hybrid model variation of makespan, increasing the vehicles number. A variable sample rate $\tau_s = 0.66$s is used.



(a)



(b)

Figure 6.7 Hybrid model variation of (a) cranes percentage occupation and (b) makespan, increasing vehicles speed, with 20 vehicles in the carousel and variable sample rate $\tau_s = 0.66$s.

Finally, it is worth to note that using a variable sample time simulator and setting $\tau_s = 0.5s$ and $\tau_s = 0.66s$ a duration of simulation of 7.97 hours and 7.64 hours respectively, has been obtained. With the same initial warehouse configuration, the same set of missions, the same number of vehicles in the carousel (20) and running the simulator on the same machine, duration of simulation using the CTPN model of Section 6.3.1 is equal to 7.25 hours: even if the use of a hybrid model allows to represent systems behavior with a greater precision, the computational efforts is, practically, the same.

6.4 Conclusions e Future researches

The results of the experimental campaign based on simulation confirm the effectiveness of the CMHPN model. Moreover, a maximum error smaller than 6% between simulation and real data has been obtained. This is a reasonable error in this field.

Current research activity is focusing on the reduction of the simulation time. It seems reasonable to reduce the ratio between real time and simulation time so that the model can be used to make online forward simulations in order to choose between different possibilities to complete a list of missions.

Chapter 7

Identification

System identification is a classical problem in system theory: it consists in determining the mathematical model that describes the behavior of a given system from the observation of the inputs and outputs evolution.

The interest for the identification of DESs usually comes from reverse engineering for (partially) unknown systems, fault diagnosis, or system verification [FS08, BCD09a]. Inputs and/or outputs sequences are observed in a *passive* way during the operation of the system within its environment. Then, the system is assumed to be controlled and the resulting model is a closed-loop model. This makes sense when the main goal of the identification is to evaluate the discrepancy between a desired behavior and the effective behavior.

The methods presented in the literature for the identification of DESs produce a mathematical model expressed as a PN or a finite state automaton model of the system behavior from sequences observed during the system operation. A survey can be found in [LML10] and in [CDFS11].

When the resulting model is a PN, the net structure (places, transitions and arcs) and its initial marking must be identified. Measurable places represent the sensors of the system and transitions represent the observed events. In the identification procedure signals from sensors are considered as outputs, while the observed events are considered as inputs.

At the best of our knowledge, in all the proposed approaches inputs cannot be modified: the identification mode is said to be *passive*. From DES control perspective, another identification mode is also of interest. If it is allowed to force/enable inputs to the actual system, the term *active* identification is used. Active identification could be useful when a supervisory control system must be designed. Indeed, a model of the system to be supervised is not always available.

The goal of the supervision is the coordination of devices, machines, robots. Each device can be modeled as a DES when it is of interest its behavior in terms of controllable and uncontrollable events [RW89, CAD99]. The occurrence of a controllable event can be disabled by an external agent called supervisor, while the occurrence of an uncontrollable event cannot be disabled. However, a plant is composed of devices built by different manufactures, and they do not provide DES models of the devices.

The supervisory control problem is, known the occurred events, to enable/disable controllable events in such a manner that a desired behavior is obtained. In DES control theory it is assumed that a symbol is associated to each event and the behavior of a DES can be described in terms of generated sequences of events/symbols. Consider for instance working cells composed by machines, conveyors and buffers: uncontrollable events can be the end of a working sequence, the arriving of a part at the end of a conveyor, the end of the loading of a part from a buffer; controllable events can be the start of a working sequence, the loading of a part on a conveyor, the loading of a part from the buffer. Desired behaviors can be a correct machine and buffers loading sequence, avoiding buffer overflows, mutual exclusion when using shared resources. Notice that controllable events are enabled by the supervisor, but they are not forced to occur.

The possibility to explore the system behavior by enabling inputs can make the online identification faster. Indeed, the fact that certain outputs have been generated when certain controllable events are enabled adds information to the identification process.

Moreover, if the system is observed in a passive mode, some output events can be seen only after a very long time, since a conflict between a set of events could be solved always in a manner that certain sequences are not generated. Active identification speeds up the identification process making the observed behavior more information rich, since varying input signals different sequences can be observed.

7.1 Literature review

The pioneer approach at the active identification is that proposed in [Ang87] to identify an automaton using examples (behaviors accepted by the system as well as by the conjecture) and counterexamples (behaviors accepted by the system but not of the the conjecture). Two agents are defined: a Learner, that have to identify the unknown system, and a Teacher, that known the behavior of the system. The first one interrogates the Teacher by means of so called *memberships queries*, sending to it some inputs and waiting for the answer (examples). When it has sufficiently information, it builds a hypothesis and, using an *equivalence query*, it asks the Teacher if such a hypothesis correspond to the unknown system. If it is the case, identifications ends else the Teacher send to the Learning a counterexample, i.e. a behavior the hypothesis is not consistent with. In the paper it is shown that the number of membership queries needed to identify the system with a fixed accuracy, depends on the number of states of the current hypothesis. The Angluin's algorithm is at the base of several other works, for example in [SHM11] the Angluin's algorithm is extended to identify a Meanly machine, while in [Jar10] it is used to identify Timed Event System.

A limitation of Angluin's method is that it needs the automaton is reset to the initial state after each query. Rivest and Schapire extend it and provide a polynomial time algorithm for inferring any finite automaton from a single continuous walk (i.e. without return the system in its initial state after each test) on the target automaton [RS, RS94]. They use the concept of *homing sequences* in lieu of the reset: an homing sequence is an input that leads the system in a known state that is used as initial state for the successive interrogation. In [FKR$^+$97] the concept of *local homing sequence* is introduced: this is a sequence of inputs that is guaranteed to orient the learner, but only if the observed output sequence match a particular pattern. Advantage respect the previous ones is that the local homing sequences are shorter and for this reason arriving in a known state can be more frequent than when homing sequences are used.

Even if active learning has been the object of numerous researches, there are very few applications in PN system identification while very common are passive approaches.

In [ELS11] an active learning for Workflow PNs (WPNs) is presented: once again identification is carried out by means of a Learner and a Teacher agent. Membership queries consist in a firing sequence of the WPN transitions; if such a sequence is compatible with the net to identify, the Teacher's answer with another firing sequence that extend the previous one, else it answers *no*. On the base of the Teacher answers the reachability graph of the net is built and then the corresponding WPN is obtained. There are two principal differences between [ELS11] and the approach presented in this dissertation: 1) PN is directly obtained solving the ILPP and 2) the Identifier does not send queries to a Teacher but directly stimulates the system, makes it changes its state. In this way when a new interrogation is executed, system can be in a state different from its initial one.

As said before, principal approaches to PNs identification are passive ones. They can be divided in three groups: 1) on-line identification based on the measurable state observation; 2) I/O based identification; 3) Integer Linear Programming Problem (ILPP) based identification.

Here below they are briefly recalled and summarized in Fig. 7.1:

1) In [MCLM05] Meda-Campaña at el. propose an asymptotic identification approach consisting in compute on-line an IPN model Q_i, describing the behavior of an unknown system Q,

from the measure of its output symbols (the marking of the subset of observable places). Every time a cyclic behavior is detected, the previous computed model Q_{i-1} is updated such that the new (updated) model Q_i acquires more detail of the system than Q_{i-1}.

Differently from the approach proposed in this thesis, the one proposed by Meda-Campaña et al. needs the presence of measurable places since the output of the system is define as the marking of such places. As more a cycle is defined in terms of the observable marking: said φM_i the observable marking at step i, after k steps the system is considered entered in a cycle if $\varphi M_0 = \varphi M_k$. In this dissertation an operative cycle definition based on the observation of repetitions in firing of transitions is proposed.

2) In [EVLLM11], Estrada-Vargas et al. present a black-box methodology to identify a PLC-based controlled DES.

It is based on the observation of inputs (firing transitions) and output (marking of the observable places) of the system: first a sufficiently long I/O sequence is obtained, than it is transformed in a sequence of events related to the changes in the observable marking. The identification result is a safe IPN having a place for each state of the system and a transition for each detected event. Such net is not minimal: to eliminate redundant nodes some transformations are necessary. At this aim in [EVLLM12] a method based on the study of direct and indirect causality matrices is performed to establish the relation between two events, allowing to reduce the number of nodes of the net. Notice that the net does not represent the actual language of the system but the sampled output language of length $k + 1$ of the DES. The parameter k is used to adjust the accuracy of the identified model, similarly as proposed in [KLL05].

The result of the algorithm proposed in this dissertation does not need any simplification because it is already the minimal PN able to describe the observed language; as more it can be not safe (an upperbound of the marking value is supposed to be preliminary known: this allow to distingue cyclical behaviors even in the case of places with marking greater of 1).

3) Identification based on the solving of a Integer Linear Programming Problem (ILPP) is argument of several works: it was introduced for the first time by Giua and Seatzu in [GS05] where authors show that the language of a free-labeled Petri net system can be described by a set of linear constraints. As more they demonstrate that the PN obtained as solution of the system of constraints generates the desired language. Successively the work has been extended to the labeled PN in [CGS07]. However in the both works, the language of the system (firing sequences with maximal length equal to k) is supposed to be known, consequently not a really identification but a synthesis is carried on.

   In [DFM08] ILP is used by Dotoli et al. to identify pre-incidence and post-incidence matrices and the initial marking of an IPN when the number of places and transitions is given and a firing sequence and the corresponding marking are (partially) observable. The difference with [GS05] is that the whole language of the system is not known: a word (a sequence of event belonging to the DES alphabet) of length h are obtained observing the evolution of the system in terms of occurred events and the available output vectors, that correspond to the markings of the measurable places. Any time a new event occurrence is detected, the word w is updated ($w = w\xi_i$, where ξ_i is the last event occurred) then, the corresponding ILP problem is constructed. Solving it the minimal IPN describing the observed language is obtained. Soon after the algorithm start to wait a new event: hence, the algorithm refines recursively the IPN modeling the DES using the on-line solution of the ILP problem. In [DFMU11] the real time identification is extended to systems having unobservable transitions: since the observable part of the system is known the algorithm can detect the occurrence of an event associated to an unobservable transition.

When this occurs a new ILP is built on the base of the last observation and the initial model is updated with the adding of the unobservable transitions.

The active approach described in the following is based on the resolution of an ILPP. As in [DFM08], the language of the DES is considered unknown but resolution of the ILPP does not start after any event occurrence but only after that a cyclical behavior has been detected.

7.2 Thesis contribution

In Fig. 7.2 the abstract model of the proposed algorithm is reported. The block DES represents the system to be identified: it is partially observable and some information about its behavior are preliminary known. The block Identifier represents an agent that can observe the DES output y_k but it can also produce inputs u_k to send to the DES. These inputs enable/disable controllable events of the system, triggering its evolution. Such inputs are generated by Identifier on the basis of previous observations and of additional information about the system. After the observation of a sufficient number of system outputs, a conjecture about its model is produced: it is an approximation of the actual system behavior.

The approach is inspired by the offline approach based on Integer Linear Programming (ILP) proposed in [CGS07] and extended to the real time identification in [DFM08]. Enabling of transitions and their firings is characterized in terms of linear constraints. For selecting among different solutions, a performance index involving the arc weights and the number of tokens in the initial markings is minimized.

In this thesis it is assumed to know the set of conflicting controllable events, i.e. controllable events whose firing disables the firing of the other events in the same conflicting set. A set of conflicting events is represented by a set of transitions having a common input place in a PN.

Paper	Property	Result	Preliminary Information	Kind of System	Identifier's Inputs	Behaviors
[MCLM05]	on-line;	IPN	measurable places	closed loop	observable marking	concur.;
	gray box; asymptotic			cyclical		choice; synch.
[EVLLM11]	off-line;	IPN	measurable places	closed loop	I/O sequence	concur.;
[EVLLM12]	black-box; based on k-accuracy			cyclical		choice; synch.
[GS05]	off-line;	free labeled PN	language L	whatever	firing seq.	concur.;
	gray box;					choice; synch.
[CGS07]	off-line; gray box;	labeled PN	language L	whatever	firing seq.	concur.; choice; synch.
[DFM08]	on-line;	free-labeled PN	alphabet; E	whatever	firing seq.;	concur.;
	gray box;		measurable places		observable marking	choice;
	asymptotic					synch.
[DFMU11]	on-line; gray box;	PN	alphabet E; measurable places	whatever	firing seq.; observable marking	concur.; choice;
	asymptotic					synch.

Figure 7.1 Summary table of principal passive identification approaches.

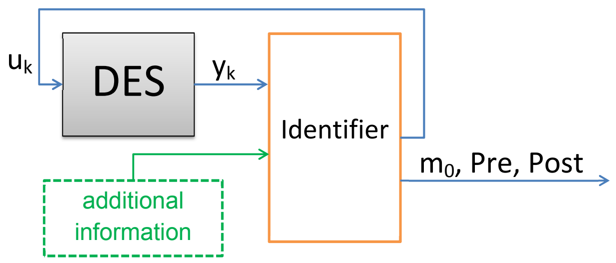


Figure 7.2 Experimental Based Identification of DES.

It is also assumed that if no events are observed for a certain time under the same control inputs, the system is blocked. This is reasonable for a live system where some control inputs are disabled. In this case, the deadlock is induced by the identification procedure. Similarly, it is assumed that a system is entered in a cycle if the same sequence is observed for a certain number of times under the same control inputs. In other words, it must be possible to detect cyclic behavior. The detection of such deadlocks as well as cyclic behaviors induced by the controller allows to obtain additional information speeding up the model identification.

Note that the plant must be controlled in a safety mode otherwise during the identification, plant devices may be damaged. This aspect is not relevant in a software engineering or re-engineering context. However, also during a passive identification process the existence of a controller exciting the devices is needed such that the behavior process plus the controller can be observed. In the problem addressed in this dissertation, such a controller is needed, but some control inputs can be disabled, since they are controllable.

This preliminary work on the topic of active identification of DESs, presents a procedure to obtain a Place Transition (P/T) PN model of the system. Such a procedure is here named Experimental Based Identification (EBI).

7.3 Notations

Let consider a system affected by different types of events then it is possible define an event (discrete) set E whose elements are all these events.

The event set E is partitioned in the set E_c of controllable events and in the set E_{uc} of uncontrollable events. Transitions associated to controllable (uncontrollable) events are drawn like white (black) bars.

Throughout this chapter the following notations will be used:

- Let s be a sequence of events belonging to E. It is said:
    - *empty*, if it has no events, ε is used to denote the empty sequence.
    - *elementary*, if it is made up of just one event: $s = \xi_i$, where $\xi_i \in E$;
    - *composed*, if it is made up of more than one event: $s = \xi_1\xi_2\xi_1\xi_3 \dots \xi_q$;

- Any interval of the system observation, during which system input is kept constant, is said *step*. Each step is denoted by an integer number.

- *enabled*$(\xi_i, k) : \big\{(\xi_i, k), \xi_i \in E_c, k \in \mathbb{N}\big\} \rightarrow \big\{0, 1\big\}$ is the *enabling function* that returns 1 if the event ξ_i is enabled in the *step* k, 0 otherwise.

- The *input* u_k is the set of events $\xi_i \in E_c$ with *enabled*$(\xi_i, k) = 1$ at the step k. For the sake of clarity, a notation without brackets will be used, i.e. assumed $E_c = \{a, b\}$, notation $u_k = a$, $u_k = b$ and $u_k = ab$ will be used instead of $u_k = \{a\}$, $u_k = \{b\}$ and $u_k = \{a, b\}$. Notice that the controllable events are concurrently enabled by Identifier $\mathcal{I}$, so $u_k = ab = ba$ since the input u_k is a set of enabled events.

- The *output sequence* $y_k = out(u_k)$ is the sequence of events belonging to E produced by the system as output to the input u_k.

Definition 7.3.1 (Multiple Sequence)**.** An *input (output) multiple sequence* U (Y) is the succession of the inputs (output sequences) sent to (received from) system $\mathcal{S}$ for a number of consecutive steps:

$$U = u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_l (Y = y_1 \rightarrow y_2 \rightarrow \cdots \rightarrow y_l)$$

The symbol $\rightarrow$ is used to remark that the set of enabled controllable events changes, $y_1 \rightarrow y_2$ means that y_2 follows y_1 when the set of enabled controllable events changes from $u_1 \rightarrow u_2$.

7.3.1 A preliminary result

Using results presented in [DFM08] the following proposition can be proved.

Proposition 7.3.1. *Consider u is given as input to the system when it is under the marking $\boldsymbol{m}'_0$, and let $w = t_1 t_2 \dots t_l$ be the output sequence observed. The system evolution is*

$$\boldsymbol{m}'_0[t_1\rangle \boldsymbol{m}'_1[t_2\rangle \boldsymbol{m}'_2 \dots [t_l\rangle \boldsymbol{m}'_l$$

.

Let $\zeta(u, w)$ be the following set of constraints

$$\begin{cases} \boldsymbol{m}'_j - \boldsymbol{m}'_{j-1} = (\mathbf{Post}' - \mathbf{Pre}') \cdot \boldsymbol{e}_j & (7.1a) \\ \boldsymbol{m}'_{j-1} \geq \mathbf{Pre}' \cdot \boldsymbol{e}_j & (7.1b) \\ -K\boldsymbol{s}_k + \boldsymbol{m}'_{j-1} - \mathbf{Pre}' \cdot \boldsymbol{e}_k \leq -\mathbf{1} & (7.1c) \\ \forall t_k : (t_k \notin w) \wedge \big((t_k \in u) \vee (t_k \notin E_c)\big) & \\ \mathbf{1}^T \cdot \boldsymbol{s}_k \leq m - 1 & (7.1d) \\ \forall t_k : (t_k \notin w) \wedge \big((t_k \in u) \vee (t_k \notin E_c)\big) & \\ \boldsymbol{m}'_j(p_i) = \boldsymbol{m}_O(p_i) \quad \forall p_i \in P_O & (7.1e) \\ \boldsymbol{m}'_j \in \mathbb{N}^m & (7.1f) \\ \mathbf{Pre}', \mathbf{Post}' \in \mathbb{N}^{m \times n} & (7.1g) \\ \boldsymbol{s}_k \in \{0, 1\}^m & (7.1h) \\ j = 1 \ldots l & (7.1i) \end{cases}$$

where $\boldsymbol{m}'_j$, $\mathbf{Pre}'$, $\mathbf{Post}'$ are unknown variables related to a net with m places and n transitions and K is a sufficiently large nonnegative integer to be a bound for the place marking.

If the system $\mathcal{S} = \langle N, \boldsymbol{m}'_0 \rangle$ is a solution of $\zeta(u, w)$, then

$$\boldsymbol{m}'_0[w\rangle$$

i.e. word w is enabled from $\boldsymbol{m}'_0$.

Proof. Since transition t_j is enabled at the marking $\boldsymbol{m}'_{j-1}$, the following condition holds: $\forall t_j \in \sigma$, $\boldsymbol{m}'_{j-1} \geq \mathbf{Pre} \cdot \boldsymbol{e}_j$. Moreover, for each fired transition, state equation $\boldsymbol{m}'_j = \boldsymbol{m}'_{j-1} + \boldsymbol{C}(\cdot, t)$ has to be satisfied, hence, $\boldsymbol{m}'_j - \boldsymbol{m}'_{j-1} = (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{e}_j$.

Transition t_k does not fire because 1) it is not enabled from the marking $\boldsymbol{m}'_{j-1}$ or 2) it is a controllable transition and it is not enabled by Identifier $\mathcal{I}$. In case 1) it exists at least a place p_i s.t. $\boldsymbol{m}'_{j-1} < \mathbf{Pre}(p_i, t_k)$. As shown in [CGS07] satisfaction of this condition correspond to satisfy constraints (7.1c), (7.1d) and (7.1h). In case 2), no constraints can be written because when a transition is not enabled by Identifier $\mathcal{I}$ its firing is prevented even if the marking of the net satisfies (7.1b).

For any observable place p_i, the value of the marking must be equal to the value received as output from the system, i.e. equation (7.1e) must hold.

Equations (7.1f) and (7.1g) in $\zeta(u, w)$ are derived by the definition of marking as well as of pre and post incidence matrices. ■

Note that the number of places of the net to be identified is unknown. As usual, in identification approaches based on mathematical programming tentative values must be used.

7.3.2 Assumptions

The approach described in this chapter works under the following assumptions.

Assumption 1 (Free labeled nets)**.** *There is an isomorphism between the event set E and the transitions set T.* $\diamond$

As consequence of Assumption 1, in this chapter, the name of the event is used to indicate the associated transition.

Assumption 2. *$\mathcal{S}$ is live, bounded and reversible.* $\diamond$

Assumption 2 implies that: i) if a block occurs in $\mathcal{S}$, it is caused by the particular set of enabled/disabled control transitions, imposed by Identifier $\mathcal{I}$; ii) the reachability graph RG of system $\mathcal{S}$ is finite and strongly connected; iii) systems have a cyclic behavior, with cycles of finite length.

Liveness and reversibility assumptions can be removed to extend the approach presented in this chapter to systems having blocking behaviors. In such a case, an opportune constraint set has to be used to represent a dead marking. However, this is out of the topic of this preliminary work.

Assumption 3. *There are not cycles in the net involving only uncontrollable transitions.* $\diamond$

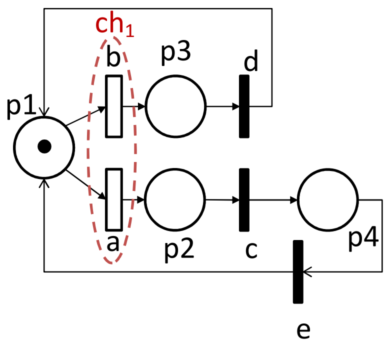


Figure 7.3 A free choice net.

Assumption 3 is usual in the supervisory control theory.

Assumption 4. $\mathcal{S}$ *has only free choices, i.e.*

$\forall p \in P, \mathrm{card}(p^\bullet) \leq 1$ *or* ${}^\bullet(p^\bullet) = \{p\}$*; equivalently*[1]

$$\forall p_1, p_2 \in P, {p_1}^\bullet \cap {p_2}^\bullet \neq \emptyset \Rightarrow \mathrm{card}({p_1}^\bullet) = \mathrm{card}({p_2}^\bullet) = 1$$

In words, it is a PN such that every arc from a place is either a unique outgoing arc or a unique ingoing arc to a transition. If the PN with only free choices is an ordinary PN, then it is called Free Choice net [Mur89](see Fig. 7.3). ◇

7.4 The Algorithm

Calling $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0} \rangle$ the P/T net system modelling the real behavior of the DES, a conjecture model $\mathcal{S}' = \langle \mathcal{N}', \mathbf{m_0}' \rangle$ is computed, on the base of the system observation.

To do it, the following preliminary information about $\mathcal{S}$ are needed:

- The set E of the system events and the set E_c of all the controllable events;

[1]Given a set S, card(S) denotes the cardinality of S.

- The maximal time T_{max} Identifier $\mathcal{I}$ has to wait before to conclude the system is entered in a still marking.

  Definition 7.4.1 (Still Marking)**.** At the *step* k, system $\mathcal{S}$ is said to be in a *Still Marking* under the input u_k if for a time at least equal to T_{max}, no events have occurred. $\diamond$

  In words a still marking represents a deadlock state induced by the fact that some controllable transitions are disabled by the input u_k. It is assumed that the identification procedure starts with $u_k = \emptyset$ and when a still marking has been reached. This is always possible when all controllable transitions are disabled, since there are not cycles involving only uncontrollable transitions. Then, without loss of generality we assume that the initial marking $\boldsymbol{m}_0$ of the observed system is always a still marking.

- The minimum number $R_{cycle} \in \mathbb{N}$ of times an output sequence has to be repeated before Identifier $\mathcal{I}$ can conclude the system is entered in a cycle.

  In this chapter it is assumed that Identifier $\mathcal{I}$ is able to detect when system $\mathcal{S}$ enters in a cycle. For this purpose an operative definition of cycle, based on the observation for R_{cycle} times of the same output multiple sequence, in correspondence of the same input multiple sequence, is introduced in the following:

  Definition 7.4.2 (Cycle)**.** A system is assumed to be in a cycle, starting from the *step* k if one of the two conditions holds.

  1. A sequence of events is repeated consecutively R_{cycle} times in the output sequence y_k. As example, given $R_{cycle} = 2$, if $y_k = abab$, Identifier $\mathcal{I}$ can conclude that the system entered in a cycle since sequence ab is repeated for a number of times equal to R_{cycle}.

2. Starting from the *step* k, a succession Y_C is contained consecutively R_{cycle} times in Y. As for example, given $R_{cycle} = 2$, if $Y = bc \to d \to bc \to d$, then starting from *step* 2 the system entered in a cycle because the subsequence $Y_C = bc \to d$ is repeated consecutively 2 times in Y. $\diamond$

- The set *Choices* consisting of all the possible conflicting set of controllable events. Both controllable and uncontrollable events can be involved in a conflict; it is assumed that all the conflicts involving only controllable events are known and a symbol (e.g. ch_1, ch_2 etc..) is assigned at each conflict. Given the set *Choices*, $eventsOf(\cdot) : Choices \to E$ is the function that for each conflict $ch_i \in Choices$ returns the set of events involved in such a conflict, e.g. in Fig. 7.3 $Choices = \{ch_1\}$ and $eventsOf(ch_1) = \{a, b\}$.

From now on a controllable event that is enabled (disabled) by Identifier $\mathcal{I}$, is said *control enabled (disabled)*; an event that is enabled (disabled) by the current marking of the system is said *state enabled (disabled).* Notice that a controllable event can be, at the same time, control enabled but not state enabled.

EBI algorithm is repeated cyclically: each new execution is called iteration i of the EBI algorithm.

If an output multiple sequence observed at the last observation is enabled for the conjecture $\mathcal{S}'_{i-1}$, obtained at the end of the previous iteration, the algorithm will not update it. Then, the definition of conjecture consistent sequence is given.

Definition 7.4.3 (Conjecture Consistence Sequence)**.** Let Y_i be the output multiple sequence produced by system $\mathcal{S}$ during the i-th iteration when the input multiple sequence U_i is given as input. Let $\mathcal{S}'_{i-1}$ be the conjecture obtained from the $(i-1)$-th algorithm iteration. If sending as input U_i to the conjecture $\mathcal{S}'_{i-1}$, it is produced the output multiple sequence Y_i, conjecture $\mathcal{S}'_{i-1}$ is said to be consistent with reference to the i-th observation. $\diamond$

At each algorithm iteration i, system dynamic is observed and a conjecture $\mathcal{S}'_i$, *consistent* with reference to the i-th observation, is given as result of the iteration.

Each iteration i can be divided in 4 phases:

- Phase 1: System Observation – Identifier $\mathcal{I}$ sends the set of control enabled controllable events as input to system $\mathcal{S}$ and it records the output sequence;

- Phase 2: Conjecture Consistence Checking – The consistence of the last conjecture w.r.t. the current observation is checked to decide if starting a new iteration or continuing with execution of Phase 3 and 4.

- Phase 3: Algebraic Constraint System Construction – A set of linear constraints, describing the observed dynamic of the system during the current iteration, is built;

- Phase 4: ILP Resolution – A consistent conjecture is obtained from the resolution of an ILP problem associated to the set of constraints obtained in Phase 3.

While Phase 1 is always executed at each iteration, Phase 3 and 4 are executed only if the conjecture $\mathcal{S}'_{i-1}$, obtained at the end of the previous iteration, is not consistent w.r.t. the i-th observation, otherwise $\mathcal{S}'_i = \mathcal{S}'_{i-1}$.

EBI algorithm is reported in Fig. 7.4; in the next, each phase will be explained in detail.

7.4.1 Phase 1: System Observation

EBI algorithm remains in Phase 1 until a cycle is detected (notice that for Assumption 2 this always occurs).

From now on, the index i will be used to indicate the i-th algorithm iteration while the index k will be used to indicate the k-th step of the System Observation executed during the i-th iteration.

```
  Requires: $E$, $E_c$, $Choices$, $R_{cycle}$, $T_{max}$.
(* Set of all the possible multiple input sequences construction *)
$\mathcal{U}^{eq}$=ComputeAllPossibleMultipleInputSequences($E_c$,$Choices$)
(* Iteration number initialization *)
i=1
(* Start of identification*)
Repeat
(* Selection of the input multiple sequence for the current step *)
  $\overline{U}_i$=SelectOneInputMultipleSequence($\mathcal{U}^{eq}$,$\mathcal{S}'_{i-1}$);
(* Start of Phase 1: System Observation *)
  $[Y_i, U_i]$=SystemObservation($\overline{U}_i$, $R_{cycle}$, $T_{max}$);
    (* Start of Phase 2: Conjecture Consistence Check *)
  if NotConsistent($Y_i$, $U_i$, $\mathcal{S}'$)
        (* If the current conjecture $\mathcal{S}'_{i-1}$ is not consistent w.r.t. the $i$-th observation*)
        (* Start of Phase 3: Algebraical Constraint Computation *)
        (* Computation of system $\mathcal{A}_i(U_i, Y_i)$ *)
    $\mathcal{A}_i$=ConstraintsComputation($Y_i$, $U_i$, $E$ ,$E_c$, $Choices$);
    (* Computation of system $\mathcal{A}(U, Y)$ *)
    $\mathcal{A}$=ComputeNewSys($\mathcal{A}_i$, $\mathcal{A}$)
    (* Start of Phase 4: ILP Resolution *)
    (* New conjecture $\mathcal{S}'_i$ computation *)
    $S'_i$=SolvingILP($\mathcal{A}$);
  end
(* Iteration number incrementing *)
  i=i+1
Until Stopped
(* Identification ends when no new consistent conjectures are produced *)
```

Figure 7.4 Experimental Based Identification algorithm.

Choice of the input multiple sequence

At each step of the i-th System Observation, a set of controllable events has to be control enabled.

Controllable events can be enabled in more than one way (a single event at time, more events contemporaneously, in different order respect to a previous enabling). Each of these combinations can be seen as a different input multiple sequence.

Let $\mathcal{U}^{eq}$ be the set of all the possible input multiple sequences, obtaining with n_{Ec} controllable events. Its cardinality is

$$N_U = n_{Ec}! + n_s \tag{7.2}$$

where

$$0 \leq n_s \leq \sum_{p=2}^{n_{eq}} \Big[\frac{n_{eq}!}{(n_{eq}-p)!p!} \cdot (n_{Ec} - p + 1)! \cdot max_j(eventsOf(ch_j))\Big]$$

and $n_{eq} = n_{Ec} + n_{ch} - \sum_{j=1}^{n_{ch}} eventsOf(ch_j)$.

Term $n_{Ec}!$ is the number of all the possible input multiple sequences obtaining enabling in succession one controllable event at time, considering all the possible orders of enabling. Term n_s is the number of the input multiple sequences obtained enabling p events at the same time and the others in succession, less the number of forbidden input multiple sequences, i.e. the number of input multiple sequences in which events belonging to the same choice ch_j are enabled at the same time. Indeed in each multiple sequence only one event at time for each conflict can be enabled by Identifier $\mathcal{I}$ to explore the alternatives of the net behavior.

As example, assume $E_c = \{a, b, c\}$, then 13 input multiple sequences are possible: $a \rightarrow b \rightarrow c$, $a \rightarrow c \rightarrow b$, $b \rightarrow a \rightarrow c$, $b \rightarrow c \rightarrow a$, $c \rightarrow a \rightarrow b$,$c \rightarrow b \rightarrow a$, $ab \rightarrow c$, $c \rightarrow ab$, $ac \rightarrow b$, $b \rightarrow ac$, $cb \rightarrow a$, $a \rightarrow cb$, abc. Consider the previous example, assuming that controllable events a and b are involved in the conflict ch_1, only the following 10 multiple sequences could be considered: $a \rightarrow b \rightarrow c$, $a \rightarrow c \rightarrow b$, $b \rightarrow a \rightarrow c$, $b \rightarrow c \rightarrow a$, $c \rightarrow a \rightarrow b$,$c \rightarrow b \rightarrow a$, $ac \rightarrow b$, $b \rightarrow ac$, $cb \rightarrow a$, $a \rightarrow cb$.

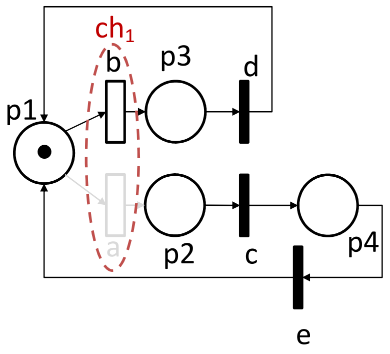


Figure 7.5 Induced Net obtained starting from the one of Fig. 7.3, after the disabling of the controllable event a and the enabling of the controllable event b.

Allowing that just one event for each conflict is enabled at each step of each iteration means that, at each step, conflicts are solved and a new net is obtained, called *Induced Net*. The Induced Net structure is equal to the net of system $\mathcal{S}$ after the clearing of all the disabled controllable transitions. In Fig. 7.5, it is shown the Induced Net obtained from the net of Fig. 7.3, when Identifier $\mathcal{I}$ enables the event b and keeps a disabled. The behavior of system $\mathcal{S}$ under a given input in terms of observed sequences can be explained on the basis of the induced net.

Execution of phase 1

After the choice of the input multiple sequence $\overline{U}_i$, the following two actions are cyclically repeated:

A1: *Step k Induced Net construction*: At each *step* k of the i-th System Observation, the input u_{ik} of $\overline{U}_i$ is control enabled, i.e., if $\overline{U}_i = a \to bc$ at the *step* 1, $u_{i1} = a$ will be enabled while at the *step* 2, $u_{i2} = bc$ will be enabled.

Each sequence u_{ik} is kept enabled during the whole *step* k.

A2: *Step k Induced Net evolution observation*: Identifier $\mathcal{I}$ sends to the system the enabled input u_{ik} and it records the system output y_{ik}.

Each observation step ends either when a cycle is detected, or when a still marking is detected.

If a cycle is detected, Phase 1 is stopped even if the end of $\overline{U}_i$ has not been reached, as example if $\overline{U}_i = a \to bc$ and at *step* 1, with $u_{i1} = a$, a cycle starts, System Observation ends but $u_{i2} = bc$ has not been enabled.

If a still marking is detected a new observation step starts, and actions A1 and A2 are repeated.

If the end of $\overline{U}_i$ has been reached without a cycle has been detected, Identifier $\mathcal{I}$ continues to trig the system, starting again from the first input of the succession, until a cycle is detected. As example, assuming again $\overline{U}_i = a \to bc$, if after the second step no cycles have been detected, *step* 3 starts and $u_{i3} = u_{i1} = a$.

At the end of Phase 1, said L the number of steps executed during System Observation, input (output) multiple sequence $U_i = u_{i1} \to \cdots \to u_{iL}$ ($Y_i = y_{i1} \to \cdots \to y_{iL}$), is obtained as the succession of the inputs (output sequences) sent (received) by Identifier $\mathcal{I}$ at each step of the System Observation.

7.4.2 Phase 2: Conjecture Consistence Checking

When a conjecture $\mathcal{S}'_{i-1} = \langle N'_{i-1}, \boldsymbol{m}'_{0i-1} \rangle$ obtained as solution of the previous iteration of the EBI algorithm is available, a check to determine if it is consistent w.r.t. the current observation is done, If $\mathcal{S}'_{i-1}$ is consistent, Phase 3 and 4 are not executed and a new iteration $i+1$ starts with a System Observation.

7.4.3 Phase 3: Algebraic Linear Constraint System Computation

During Phase 3, a properly set of algebraic constraints, describing the observed system evolution, is formulated.

To do it, first of all the set of constraints presented in Proposition 7.3.1 is written for each step of System Observation.

Let y_{ik} be the observed word when u_{ik} is given as input to system $\mathcal{S}$ and suppose that l is its length. The notation $\boldsymbol{m}'_{ikj}$ is used to indicate the marking of the system after that the j-th event of y_{ik} has occurred, consequently the marking of the system at the beginning of the *step* k is $\boldsymbol{m'}_{ik0}$.

The set of constraints $\zeta(u_{ik}, y_{ik})$ enforces that the conjecture enables y_{ik} under the marking $\boldsymbol{m'}_{ik0}$.

The *congruence marking equation* has to be added to $\zeta(u_{ik}, y_{ik})$:

- For $i = 1$ and $k = 1$
$$\boldsymbol{m}'_{110} = \boldsymbol{m}_0 \tag{7.3}$$
- For $i > 1$ and $k = 1$
$$\boldsymbol{m}'_{ik0} = \boldsymbol{m}'_{(i-1)ql} \tag{7.4}$$
where $\boldsymbol{m}'_{(i-1)ql}$ s.t. $\boldsymbol{m}'_{(i-1)00}[Y_{(i-1)}\rangle \boldsymbol{m}'_{(i-1)ql}$.
- For all $k > 0$
$$\boldsymbol{m}'_{ik0} = \boldsymbol{m}'_{i(k-1)l} \tag{7.5}$$
where $\boldsymbol{m}'_{i(k-1)l}$ s.t. $\boldsymbol{m}'_{i(k-1)0}[y_{i(k-1)}\rangle \boldsymbol{m}'_{i(k-1)l}$.

Equation (7.3) enforces the initial marking of the first iteration to be equal to the initial marking of the unknown system; equation (7.4) enforces the conjecture marking at the beginning of iteration i to be equal to the one it reached at the end of the previous iteration; equation (7.5) enforces the conjecure marking at the beginning of *step* k to be equal to the marking it reached after the firing of the last event in the output sequence $y_{i(k-1)}$ of the previous step.

As more, further constraints, obtained by the additional information and preliminary assumptions, are added to the previous ones:

for each conflict $ch_i \in Choices$

$$\begin{cases} Pre'_i(p, \xi_j) \geq 1 \ \forall \xi_j \in eventOf(ch_i) & \text{(7.6a)} \\ Pre'_i(p, \xi_j) = 0 \ \forall \xi_j \notin eventOf(ch_i) & \text{(7.6b)} \end{cases}$$

where $p \in {}^\bullet(ch_i)$

Equations (7.6a) - (7.6b) enforces that controllable events involved in the same conflict have the same input place p.

Let $\mathcal{A}_i(U_i, Y_i)$ be the constraint set made up of: i) as many repetitions of $\zeta(u_{ik}, y_{ik})$ as the number of step of the current System Observation; ii) the (7.4) only for the first step of the current System Observation (if it is the first iteration, equation (7.3) is used); iii) the repetition of equation (7.5) for all the other steps; iv) equations (7.6a) and (7.6b) repeated for all the controllable conflicts of system $\mathcal{S}$ (only for the first System Observation).

Such a set is used to update the total one, $\mathcal{A}(U, Y)$, obtained in the following way: if it is the first algorithm iteration, $\mathcal{A}(U, Y)= \mathcal{A}_1(U_1, Y_1)$. For all the successive iterations, if it does not exist a conjecture $\mathcal{S}'_{i-1}$ resulting from the previous iteration s.t. it is consistent w.r.t. the current system observation, all the constraints of $\mathcal{A}_i(U_i, Y_i)$ are added to $\mathcal{A}(U, Y)$.

7.4.4 Phase 4: ILP Problem Resolution

Since more than a PN system can satisfy the constraint set $\mathcal{A}(U, Y)$ obtained in the Phase 3, a selection among all these solutions has to be done minimizing a given cost function $f(\boldsymbol{m}_0{}'_i, \mathbf{Pre}'_i, \mathbf{Post}'_i)$. In particular a solution to the identification problem can be computed by solving the following Integer Programming problem

$$\min_{\substack{s.t.\\ \mathcal{A}(U,Y)}} \; f\big(\boldsymbol{m}_0{}'_i, \mathbf{Pre}'_i, \mathbf{Post}'_i\big)\,, \tag{7.7}$$

as it has been proposed in [CGS07].

Different choices can be made for the cost function. If $f(\cdot)$ is chosen linear in the unknowns, then (7.7) becomes an Integer Linear Programming (ILP) problem. In particular if the cost function is chosen as

$$f\big(\boldsymbol{m}_0{}'_i, \mathbf{Pre}'_i, \mathbf{Post}'_i\big) = \mathbf{1_m^T} \cdot \boldsymbol{m}_{\mathbf{0}}{}'_{\mathbf{i}} + \mathbf{1_m^T} \cdot \big(\mathbf{Pre'_i} + \mathbf{Post'_i}\big) \cdot \mathbf{1_n}\,,$$

the solution minimizes the sum of the tokens in the initial marking and of the arc weights [CGS07, FS08].

At the end of this phase a new conjecture $\mathcal{S}'_i = \langle N'_i, \boldsymbol{m}_0{}'_i \rangle$ is obtained, with $N'_i = \big(P, T, \mathbf{Pre}'_i, \mathbf{Post}'_i\big)$.

7.4.5 Stop Condition

Conjecture $\mathcal{S}'_i$, obtained as result of the iteration i is consistent with the output multiple sequences observed until the i-th iteration but it can result consistent also with the other Y_j obtainable starting from still untested $\overline{U}_j$. Let $\mathcal{U}^C_i$ be the set of the input multiple sequences the conjecture $\mathcal{S}'_i$ is consistent with and let N_{iC} be its cardinality.

To obtain such a set the following algorithm can be applied distinguishing between two cases:

Case A: $i = 1$ or $\mathcal{S}'_i \neq \mathcal{S}'_{i-1}$

1. given $\mathcal{S}'_i$, draw its reachability graph RG_i;

2. At each node of RG_i add a self loop, for each controllable event that has been tested to be state disabled under the marking corresponding to such node: as example, suppose that at the beginning of *step* k of the i-th iteration, the marking of $\mathcal{S}'_i$ is m'_{ik0} and the input $u_{(ik} = \xi_j$ is given ; if the corresponding output is $y_{(ik} = \varepsilon$ (i.e. a still state is detected) then ξ_j is control enabled but not state enabled. As consequence a self loop labeled ξ_j and drawn as a dashed arc has to be added to the node of RG_i corresponding to m'_{ik0}.

3. Find the node corresponding to m'_{iql}, that is the marking of $\mathcal{S}'_i$ at the end of i-th iteration. Starting from such a node individuate all the input multiple sequences that bring $\mathcal{S}'_i$ back to m'_{iql}. These input multiple sequences are the ones belonging to $\mathcal{U}^C_i$.

Case B: $i > 1$ and $\mathcal{S}'_i = \mathcal{S}'_{i-1}$

1. Said $\mathcal{U}^C_{i-1}$ the set obtained for $\mathcal{S}'_{i-1}$, and $\overline{U}_i$ the input multiple sequence tested at the iteration i, then

$$\mathcal{U}^C_i = \mathcal{U}^C_{i-1} \bigcup \{\overline{U}_i\}$$

.

2. If new still states have been discovered, add the corresponding dashed arcs to RG_i.

Consequently $\mathcal{U}_i^{eq} = \mathcal{U}^{eq} - \mathcal{U}_i^C$ is the set of input multiple sequences that have to be tested to establish if $\mathcal{S}_i'$ is the model of the system and its cardinality is $N_{Ui} = \mathrm{card}(\mathcal{U}_i^{eq}) = N_U - N_{iC}$.

Then the following stop condition can be defined:

Definition 7.4.4 (Stop Condition)**.** The EBI algorithm stops when the current conjecture $\mathcal{S}_i'$ results to be consistent with all the N_{Ui} successive iterations obtained testing all the input multiple sequences of the set $\mathcal{U}_i^{eq}$.

7.5 Applications

In the following two applications of the EBI algorithm are discussed. In the first one the algorithm is used to identify the model of a lift, in the second a simple version of a handling system is used as object of the identification.

7.5.1 Example 1: the lift

Consider a lift able to move up and down for reaching different levels. It can be moved acting on two controllable signals, from now on called "go up" and "go down".

Initially the lift is stopped at a level: when the command " go up" is sent by the controller, the lift goes up to the next level (if it does not exist the lift remains at the current level), when the command "go down" is sent, the lift goes down to the previous level (if it does not exist the lift remains at the current level).

Such a lift can be modeled as shown in Fig. 7.6 (the meaning of each place and transition is reported in Table 7.7). The set of events is $E : \{u, d, n, p\}$. Its controllable events set is $E_c = \{u, d\}$.

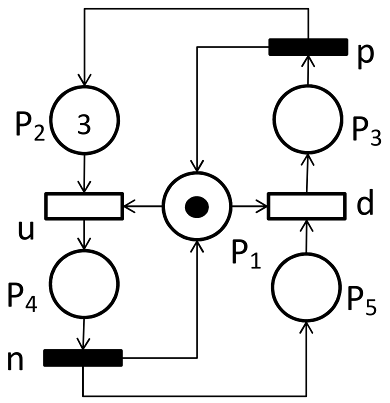


Figure 7.6 Petri Net model of a lift, able to move between 3 levels and starting from the lowest one.

Place	Meaning
P_1	lift at the level
P_2	number of next levels
P_3	number of previous levels
P_4	lift going up
P_5	lift going down
Event	**Meaning**
d	go down
u	go up
n	arrived to the next level
p	arrived to the previous level

Figure 7.7 Meaning of places and transitions of Fig. 7.6.

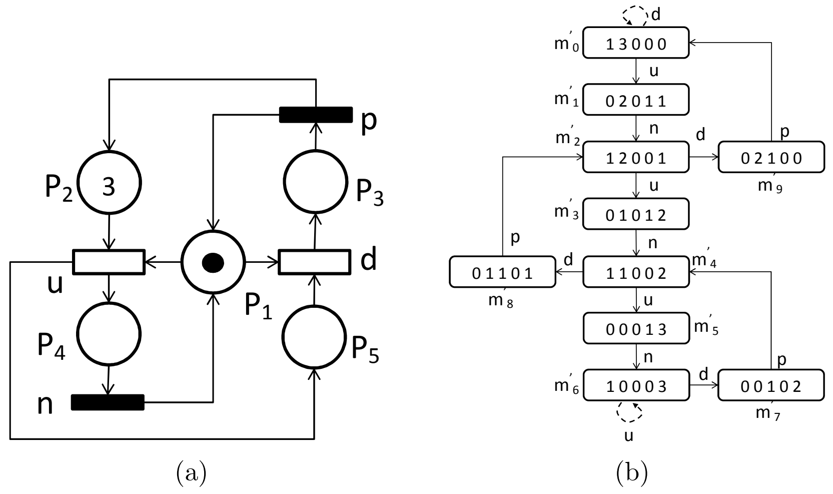


Figure 7.9 (a) Conjecture $\mathcal{S}'_1$ obtained after the first EBI iteration and (b) its reachability graph.

Since when the lift is stopped at a level, in general it is possible to chose if moving it up or down, the corresponding controllable signals are involved in a choice and hence $Choices = \{(u, d)\}$.

Since $n_c = 2$, $n_{ch} = 1$; $n_{eq} = 1$ and $n_s = 0$, hence for the (7.2), the number of all the possible input multiple sequences is $N_U = 2$.

As more, $R_{cycle} = 4$ and $T_{max} = 30s$ are given as input data to the identification problem. The value of R_{cycle} is determined on the base of the largest number of levels the lift can move across and T_{max} on the maximal time the lift takes to arrive to another level.

The set $\mathcal{U}^{eq}$ contains 2 different input multiple sequences: $u \to d$ and $d \to u$. In the table of Fig. 7.8, the iterations executed to identify the system are summarized. For the sake of readable the following compact notations have been used:

- $\underbrace{\xi_1\xi_2\ldots\xi_n\cdots\xi_1\xi_2\ldots\xi_n}_{q\ times}=(\xi_1\xi_2\ldots\xi_n)^q$
- $\underbrace{\xi_1\xi_2 \to \cdots \to \xi_n\cdots\xi_1\xi_2 \to \cdots \to \xi_n}_{q\ times}=(\xi_1\xi_2 \to \cdots \to \xi_n)^q$

i= 1: The selected $\overline{U}_1$ is $u \to d$. At *step* 1 event u is control enabled and the output $y_1 = (un)^3$ is observed. Since se-

i	k	$\overline{U}_i$	u_{ik}	y_{ik}	U_i	Y_i	Y_i'	Comments
1	1	$u \rightarrow d$	u	$(un)^3$		$(un)^3$		still state
	2		d	$(dp)^3$		$(un)^3 \rightarrow (dp)^3$		still state
	3		u	$(un)^3$		$(un)^3 \rightarrow (dp)^3 \rightarrow (un)^3$		still state
	4		d	$(dp)^3$		$[(un)^3 \rightarrow (dp)^3]^2$		still state
	5		u	$(un)^3$		$[(un)^3 \rightarrow (dp)^3]^2 \rightarrow (un)^3$		still state
	6		d	$(dp)^3$		$[(un)^3 \rightarrow (dp)^3]^3$		still state
	7		u	$(un)^3$		$[(un)^3 \rightarrow (dp)^3]^3 \rightarrow (un)^3$		still state
	8		d	$(dp)^3$	$(u \rightarrow d)^4$	$[(un)^3 \rightarrow (dp)^3]^4$	none	cycle

Figure 7.8 System Observation of Example 1.

quence un is repeated only 3 ($< R_{cycle}$) times, no a cycle but a still state is detected.

At *step* 2 event d is control enabled and the corresponding output $y_2 = (dp)^3$ is observed. As at the previous step a still state is reached.

Since all the events composing $\overline{U}_1$ have been tested without a cycle detection, from *step* 3 until the end of iteration the rule $u_i = u_{1+i-3}$ is applied (i.e. $u_3 = u_1$, $u_4 = u_2$ and so on - refer to 7.4.1).

At *step* 8 a cycle is detected since output multiple sequence $(un)^3 \rightarrow (dp)^3$ has been observed for 4 ($= R_{cycle}$) times successively. Consequently the iteration stops.

Since no other conjectures have been made previously, Phase 3 starts and the set of constraints $\mathcal{A}(U, Y)= \mathcal{A}_1(U_1, Y_1)$ is obtained.

The resulting ILP problem is solved in Phase 4: the net of Fig. 7.9(a) is assumed as consistent conjecture $\mathcal{S}'_1$.

The corresponding reachability graph RG_1, completed with control disabled event arcs, is shown in Fig. 7.9(b). W.r.t. the figure, at the end of the iteration the final marking of the system is supposed to be m'_0, consequently the set $\mathcal{U}_2^C$ is $\{u \rightarrow d,\, d \rightarrow u\}$. As consequence $\mathcal{U}_1^{eq} = \emptyset$ and $N_{U_1} = 0$: i.e. identification ends and $\mathcal{S}'_1$ is assumed as the desired model.

7.5.2 Example 2: the handling system

Consider the simple handling system shown in Fig. 7.10: it is made up two vehicles, $v1$ and $v2$, able to move two kind of items from two starting points, P_A and P_B, (indicated with the green flags in the figure) to two destination points, D_A and D_B (indicated with the red flags in the figure).

Each vehicle is dedicated to handling a different kind of item. Items of kind 1 are loaded at the starting point P_A, while items of

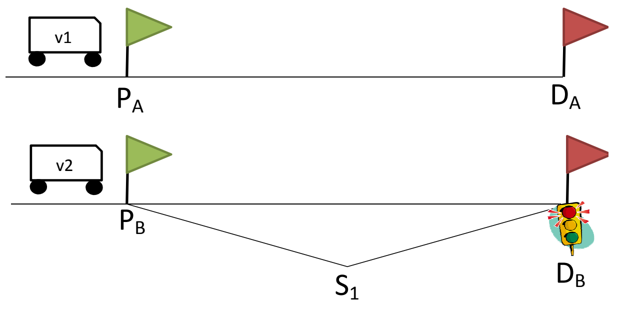


Figure 7.10 A simple handling system.

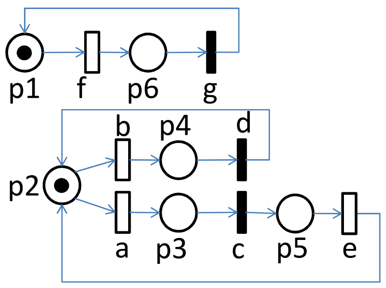


Figure 7.11 PN model of the handling system of Fig. 7.10.

kind 2 are loaded at the starting point P_B.

Items starting from P_B can arrive to their destination point following two alternative routes and vehicle $v2$ can choose between them before starting the handling. As more, along one of the two routes, after it is arrived at the destination, $v2$ has to stop and wait for a starting command.

The PN of Fig. 7.11 models such a system: meaning of each place and transition is reported in Table 7.12

For this net the set of events is $E = \{a, b, c, d, e, f\}$ and the set of controllable events is $Ec = \{a, b, e, f\}$. Since vehicle $v2$ can choose between two mutual exclusive route, one conflict involving controllable events is present, $Choices = \{(a, b)\}$. As more, $R_{cycle} = 2$ and $T_{max} = 120s$ are given as input data to the identification problem.

The set $\mathfrak{U}^{eq}$ contains 58 different input multiple sequences: for the sake of brevity only some of all the possible iterations are reported in the following and are summarized in Table 7.13.

Place	Meaning
p_1	free vehicle at P_A
p_2	free vehicle at P_B
p_3	vehicle going to S_1
p_4	vehicle going to destination D_B along route 1
p_5	vehicle going to destination D_B along route 2
p_6	vehicle going to destination D_A
Event	**Meaning**
a	start from P_B and go along route 1
b	start from P_B and go along route 2
e	return to P_B
f	start from P_A
g	return to P_A

Figure 7.12 Meaning of places and transitions of Fig. 7.6.

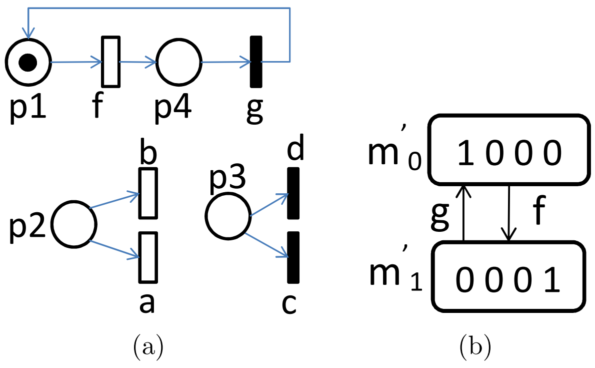


Figure 7.14 (a) Conjecture obtained after the first observation and (b) its reachability graph.

i= 1: The selected $\overline{U}_1$ leads to a cycle detection, starting from the *step* 1. As consequence Phase 1 is ended before the enabling of the other inputs in $\overline{U}_1$ and U_1 is effectively given to the system.

Since no other conjectures have been made previously, Phase 3 starts and the set of constraints $\mathcal{A}(U, Y)= \mathcal{A}_1(U_1, Y_1)$ is obtained.

The resulting ILP problem is solved in Phase 4: the net of Fig. 7.14(a) is assumed as consistent conjecture $\mathcal{S}'_1$.

i	k	$\overline{U}_i$	u_{ik}	y_{ik}	U_i	Y_i	Y'_i	
1	1	$f \to a \to e \to b$	f	$fgfg$	f	$fgfg$	none	cycle
2	1	$b \to f \to e \to a$	b	$bdbd$	b	$bdbd$	ε	cycle
3	1	$a \to e \to f \to b$	a	ac				still state
	2		e	e				still state
	3		f	$fgfg$	$a \to e \to f$	$ac \to e \to fgfg$	$a \to \varepsilon \to \varepsilon$	cycle
4	1	$a \to b \to e \to f$	a	ac				still state
	2		b	ε				still state
	3		e	e				still state
	4		f	$fgfg$	$a \to b \to f$	$ac \to \varepsilon \to e \to fgfg$	$ac \to \varepsilon \to e \to fgfg$	cycle
5	1	$a \to f \to e \to b$	a	ac				still state
	2		f	$fgfg$	$a \to f$	$ac \to fgfg$	$ac \to \varepsilon$	cycle

Figure 7.13 System Observation.

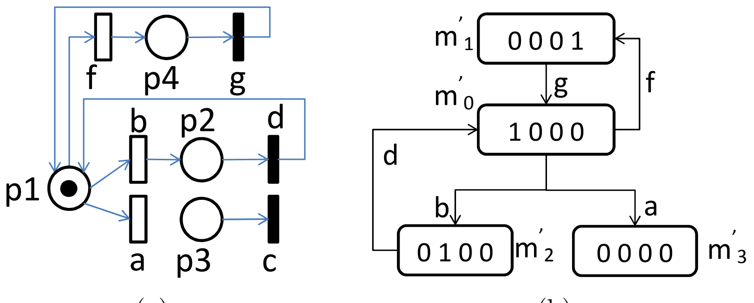


Figure 7.15 (a) Conjecture obtained after the second observation and (b) its reachability graph.

RG_1 is shown in Fig. 7.14(b): at the end of the first iteration $\mathcal{S}'_1$ is returned in its initial marking. On the base of RG_1, the input multiple sequences that return $\mathcal{S}'_1$ in m'_{i00} are $f \rightarrow *$ [2] i.e. all that $\overline{U}_i$ starting with event f. Their number is $N_{1C} = 10$ consequently $\mathcal{S}'_1 = \mathcal{S}$ iif it results consistent with the 48 input multiple sequences of $\mathcal{U}_1^{eq}$.

i= 2: At *step* 1 a cycle is detected and Phase 1 is ended.

During Phase 2, consistence of $\mathcal{S}'_1$ w.r.t. the current observation is tested: given U_2 as input, the output multiple sequence produced by $\mathcal{S}'_1$, Y'_2, is not the same of Y_2, it means $\mathcal{S}'_1$ is not consistent w.r.t. the current observation. As consequence Phase 3 and 4 are execute and the net of Fig. 7.15(a) is assumed as consistent conjecture $\mathcal{S}'_2$.

At the end of the second iteration $m'_{2ql} = m'_{200}$ [indicated as m'_0 in Fig 7.15(b)]: from RG_2, shown in Fig 7.15(b), $\mathcal{U}_2^C$ is made up of $f \rightarrow *$ and $b \rightarrow *$: identification will end if conjecture $\mathcal{S}'_2$ results consistent with all the 36 remaining input multiple sequences of $\mathcal{U}_2^{eq}$.

i= 3: At *step* 1 of the third iteration, $u_{31} = a$; the system output is $y_{31} = ac$. After the firing of c the reaching of a still

[2]Notation $\xi_1 \rightarrow \xi_2\xi_3 \rightarrow \ldots\xi_n \rightarrow *$ indicates all the input multiple sequences having $\xi_1 \rightarrow \xi_2\xi_3 \rightarrow \cdots \rightarrow \xi_n$ as prefix

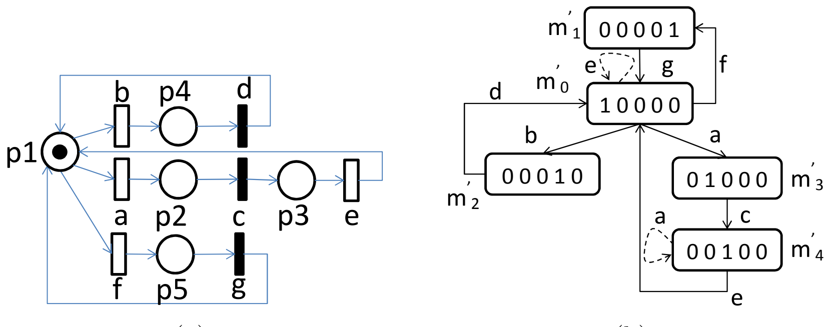


Figure 7.16 (a) Conjecture obtained after the third observation and (b) its reachability graph.

marking is detected, since for a time longer than T_{max}, no other events have occurred. Consequently a new observation step starts: at *step* 2, $u_{32} = e$; the system output is $y_{32} = e$. After the firing of e, another still marking is detected and a new step starts. At *step* 3, $u_{33} = f$; the system output is $y_{33} = fgfg$, then a cycle is detected and System Observation ends.

Since $\mathcal{S}'_2$ is not consistent w.r.t. the current observation, Phase 3 and Phase 4 are executed and the net of Fig. 7.16(a), with the reachability graph of Fig. 7.16(b), is assumed as consistent conjecture $\mathcal{S}'_3$.

Since again $m'_{3ql} = m'_{300}$ [m'_0 in Fig. 7.16(b)], $\mathcal{U}_3^C$ is $\{a \to e \to *; (ae) \to *; b \to *; f \to *; e \to f \to *; e \to b \to *; (be) \to *; (fe) \to *\}$; its cardinality is $N_{3C} = 34$, hence identification will end if conjecture $\mathcal{S}'_3$ results consistent with the remaining 30 input multiple sequences belonging to $\mathcal{U}_3^{eq}$.

i= 4: Iteration 4 ends after 4 steps: in each one of the first three steps a still marking is observed, then a cyclic behavior is detected at *step* 4.

During Phase 2, since Y'_4 is the same of Y_4 it is concluded $\mathcal{S}'_3$ is a consistent conjecture w.r.t. the current observation.

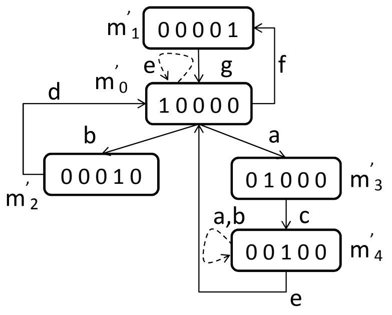


Figure 7.17 Reachability graph of Conjecture obtained after the fourth observation.

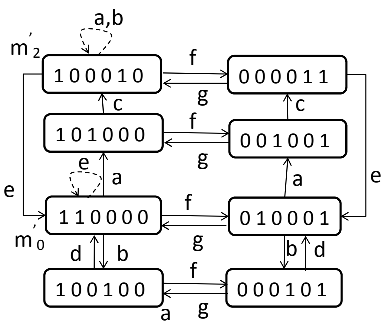


Figure 7.18 Reachability graph of conjecture $\mathcal{S}'_5$.

Consequently Phase 3 and Phase 4 do not start and the net of Fig. 7.16(a) is assumed as consistent conjecture (i.e. $\mathcal{S}'_4 = \mathcal{S}'_3$). Consequently $\mathcal{U}^{eq}_4 = \mathcal{U}^{eq}_4 - \{\overline{U}_4\}$

As more its reachability graph is updated since a new information has been obtained during this iteration: events b is not state enabled w.r.t. marking m'_4 of Fig. 7.16, hence the graph RG_4 is obtained adding the label b to the dashed arc entering in m'_4 (see Fig. 7.17).

i= 5: Iteration 5 ends after 2 steps: at the first step a still state is detected and then a cycle is detected at *step* 2.

Since $\mathcal{S}'_4$ is not a consistent conjecture, a new ILP problem is resolved and the net of Fig. 7.11 is assumed as consistent conjecture $\mathcal{S}'_5$.

At the end of iteration, the final marking is m'_2 (see Fig. 7.18).

Consequently $\mathcal{U}_6^C = \{f \to *; a \to f \to *; (af) \to *; b \to f \to *; (bf) \to *; a \to b \to f \to e; b \to a \to f \to e; e \to a \to f \to b; b \to e \to a \to f;\}$; its cardinality is $N_{5C} = 24$, hence identification will end if $\mathcal{S}'_5$ results consistent with the remaining 34 input multiple sequences of $\mathcal{U}_5^{eq}$.

i>5: For all the following iterations, whatever is the selected input multiple sequence $\overline{U}_i$, the conjecture $\mathcal{S}'_{i-1}$ result to be consistent w.r.t. the current observation.

7.6 Conclusions and Future Researches

In this chapter an active, iterative preliminary approach for the identification of free labeled PN models has been proposed. Respect to the passive identification methods, the presence of controllable events, seen as modifiable inputs, allows a faster identification of all the possible evolutions of the system. A gray-box identification method has been used: conflicting controllable events are assumed to be known as well as the maximum time that must elapse from the enabling of a transition until it fires. Moreover, it is assumed possible to detect if the system is entered in a cyclic behavior. The identification procedure is based on the solving of an Integer Linear Programming Problem.

Future researches are focused principal about how to select the next input multiple sequences in the manner to faster complete the system identification.

In this preliminary approach each $\overline{U}_i$ is chosen in a random way between all the input multiple sequences in $\mathcal{U}_i^{eq}$. But generally speaking, some input multiple sequences can better aid the identification process than others: goal of future works is individuate such a sequence that allows to discover the largest number of yet unknown behaviors, basing such a research on the liveness and reversible property of the system.

As more a stronger stop condition is under investigation: at the moment, a system is considered identified if all the possible tests succeeded; in future works it will be investigated the possibility to reduce the number of iterations.

Finally the extension of the approach to timed PNs is studying: activity of real systems are characterized by a finite duration. i.e. each task needs some time to be completed. Timed PNs allows to model this duration. As more, as shown in [BCCDT11] time information can be used to accelerate the system identification, with respect to the untimed approaches. In particular, exploiting the timing it is possible to determine a set of counterexamples that can be used to improve the net identification.

Chapter 8

Table of Notations

Table 8.1: Table of notations.

Notation	Meaning
$B = (\mathcal{C}_{\mathcal{T}}, T_{in}, T_{out})$	CTPN block
$\boldsymbol{C}$	incidence matrix
$\mathcal{C} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, Cl, Co \rangle$	Colored Petri net
$Choices$	set controllable conflicting
ch_j	j-th choice
Cl	set of colors
$Co : P \cup T \longrightarrow Cl$	color function
$Co(p_i) = \{a_{i,1}, a_{i,2}, ..., a_{i,u_i}\}$	ordered set of possible colors of tokens in p_i
$Co(t_j) = \left\{b_{j,1}, b_{j,2}, ..., b_{j,v_j}\right\}$	ordered set of possible occurrence colors in t_j
ξ	event
D	set of dummy places
$\boldsymbol{\delta} : T^D \to (\mathbb{R}^+)^{n_d}$	firing delay vector
e	internal condition
$\boldsymbol{e}_i$	i-th canonical basis vector
E	set of events
E_C	set of controllable events

Table 8.1: Table of notations (continued)

Table 8.1: Table of notations (continued)

Notation	Meaning
E_{uc}	set of uncontrollable events
$enabled(e_i, k)$	enabling function
$eventsOf(\cdot)$	function that returns the set of events involved in a given conflict
ε	empty sequence
g	control input
$\mathcal{H} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, h, \boldsymbol{\delta}, \boldsymbol{\nu} \rangle$	Hybrid Petri net
$h : P \bigcup T \rightarrow \{D, C\}$	hybrid function
$\mathcal{I}$	identifier
I	feeding speed
$\mathcal{M} = \left\{\mathcal{H}, Cl, Co, \boldsymbol{\nu}\right\}$	Colored Modified Hybrid Petri net
$\boldsymbol{m}$	marking vector
$\boldsymbol{m}_0$	initial marking vector
$\boldsymbol{m}_O$	marking of observable places
$\boldsymbol{m}_{uO}$	marking of unobservable places
$\boldsymbol{m}^C$	marking of continuous places
$\boldsymbol{m}^D$	marking of discrete places
$\boldsymbol{m}\left[\sigma\right\rangle$	enabled sequence σ
$\dot{\boldsymbol{m}}_{p^C}$	balance
$\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$	Place/Transition Petri net
n	number of transitions
n_d	number of discrete transitions
n_c	number of continuous transitions
n_{Ec}	number of controllable events
n_{eq}	number of equivalent events
n_{ch}	number of choises
N_{iC}	cardinality of $\mathcal{U}_i^C$
N_U	cardinality of $\mathcal{U}^{eq}$
$\boldsymbol{\nu} : T^c \rightarrow (\mathbb{R}^+)^{n_c}$	firing speed vector
O	draining speed
P	set of places

Table 8.1: Table of notations (continued)

Table 8.1: Table of notations (continued)

Notation	Meaning
P_O	subset of observable places
P_{uO}	subset of unobservable places
P^D	set of discrete places
P^C	set of continuous places
$\mathbf{Pre}$	pre-incidence matrix
$\mathbf{Post}$	post-incidence matrix
$Pre(p_i;t_j)(hk)$	weight of the arc from place p_i, w.r.t. color $a_{i,h}$, to transition t_j, w.r.t. color $b_{j,k}$
$Post(p_i;t_j)(hk)$	weight of the arc from transition t_j, w.r.t. color $b_{j,k}$, to place p_i, w.r.t. color $a_{i,h}$
$^\bullet p$ ($^\bullet t$)	place p (transition t) pre-set
$p^\bullet$ ($t^\bullet$)	place p (transition t) post-set
$R(\mathcal{N}, \boldsymbol{m}_0)$	set of reachable markings
RG	reachability graph
S	Siphon
s	sequence
$\mathcal{S} = \langle \mathcal{N}, \boldsymbol{m_0} \rangle$	Petri net system
σ	firing sequence
$\boldsymbol{\sigma}$	firing vector
T	set of transitions
T_{in}	set of input transitions
T_{out}	set of output transitions
T^D	set of discrete transitions
T^C	set of continuous transitions
U	input multiple sequence
u_k	input: set of control enabled events at step k
u_i	cardinality of $Co(p_i)$
$\mathcal{U}^{eq}$	set of all the possible input multiple sequences

Table 8.1: Table of notations (continued)

Table 8.1: Table of notations (continued)

Notation	Meaning
$\mathcal{U}_i^C$	set of consistent input multiple sequences
$\mathcal{U}_i^{eq}$	set of the input multiple sequences for the i-th iteration
v_j	cardinality of $Co(t_j)$
w	number of places
w_d	number of discrete places
w_c	number of continuous places
$\boldsymbol{Y}$	P-semiflow
y_k	output sequence
Y	output multiple sequence

Table 8.1: Table of notations

Bibliography

[ABCC05] F. Amato, F. Basile, C. Carbone, and P. Chiacchio, *An approach to control automated warehouse systems*, Control Engineering Practice **13** (2005), no. 10, 1223 – 1241.

[ACH$^+$95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, *The algorithmic analysis of hybrid systems*, Theor. Comput. Sci. **138** (1995), no. 32, 3–24.

[Ang87] D. Angluin, *Learning regular sets from queries and counterexamples*, Information and Computation **75** (1987), no. 2, 87–106.

[BC07] F. Basile and P. Chiacchio, *On the implementation of supervised control of discrete event systems*, IEEE Transactions on Control Systems Technology **15** (2007), no. 4, 725 –739.

[BCC07a] F. Basile, C. Carbone, and P. Chiacchio, *Feedback control logic for backward conflict free choice nets*, IEEE Transaction on Automatic Control **52** (2007), no. 3, 387–400.

[BCC07b] ———, *Simulation and analysis of discrete-event control systems based on Petri nets using PNetLab*, Control Engineering Practice **15** (2007), 241 – 259.

[BCCDT11] F. Basile, P. Chiacchio, J. Coppola, and G. De Tommasi, *Identification of Petri nets using timing information*, 3^{rd} International Workshop on Dependable Control of Discrete Systems (DCDS 11), Saarbrucken, Germany (2011), 154 –161.

[BCCG12] F. Basile, P. Chiacchio, J. Coppola, and D. Gerbasio, *A hybrid Petri nets approach for unmanned aerial vehicles monitoring*, 2012 IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2012), Krakow, Poland (2012).

[BCD09a] F. Basile, P. Chiacchio, and G. De Tommasi, *An efficient approach for online diagnosis of discrete event systems*, IEEE Transaction on Automatic Control **54** (2009), no. 4, 748–759.

[BCD09b] F. Basile, P. Chiacchio, and D. Del Grosso, *A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri net*, Computer Standards & Interfaces **31** (2009), no. 3, 528–538.

[BCD12] F. Basile, P. Chiacchio, and G. De Tommasi, *On $\mathcal{K}$-diagnosability of Petri nets via integer linear programming*, Automatica **48** (2012), 2047–2058.

[BCG07] F. Basile, P. Chiacchio, and A. Giua, *An optimization approach to Petri net monitor design*, IEEE Transaction on Automatic Control **52** (2007), no. 2, 306–311.

[BGM00] F. Balduzzi, A. Giua, and G. Menga, *First-order Hybrid Petri Nets: a model for optimization and control*, IEEE Transactions on Robotics and Automation **16** (2000), no. 4, 382 –399.

[BGS01] F. Balduzzi, A. Giua, and C. Seatzu, *Modelling and simulation of manufacturing systems with first-order* *hybrid Petri nets*, International Journal of Production Research **39** (2001), no. 2, 255 – 282.

[BW84] Y. A. Bozer and J. A. White, *Travel-time models for automated storage/retrieval systems*, IEE Transactions **16** (1984), no. 4, 329–338.

[CAD99] F. Charbonnier, H. Alla, and R. David, *The supervised control of discrete event system*, IEEE Transaction on Control System Technology **7** (1999), no. 2, 175–187.

[CDFS11] M.P. Cabasino, P. Darondeau, M.P. Fanti, and C. Seatzu, *Model identification and synthesis of discrete-event systems*, in Contemporary Issues in Systems Science and Engineering, 2011.

[CGS07] Maria Paola Cabasino, Alessandro Giua, and Carla Seatzu, *Identification of Petri nets from knowledge of their language*, Discrete Event Dynamic Systems **17** (2007), 447–474.

[CL08] Christos G. Cassandras and Stéfane Lafortune, *Introduction to discrete event systems, 2nd ed.*, Springer, 2008.

[CPV99] R. Champagnat, H. Pingaud, and R. Valette, *An extension of high-level Petri nets for modelling batch systems*, IMACS/IEEE International Conference on Circuits, Systems, Communications and Computers (CSCC'99) (1999), 2621–2625.

[DA01] R. David and H. Alla, *On hybrid Petri nets*, Discrete Event Dynamic Systems **11** (2001), 9–40.

[DA05] ______, *Discrete, continuous and Hybrid Petri Nets*, 2005.

[DF05] M. Dotoli and M. Fanti, *A coloured Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: A control perspective.*, International Journal of Computer Integrated Manufacturing **18** (2005), no. 2, 122–136.

[DF07] M. Dotoli and M.P. Fanti, *Deadlock detection and avoidance strategies for automated storage and retrieval systems*, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **37** (2007), no. 4, 541 –552.

[DFM08] M. Dotoli, M.P. Fanti, and A.M. Mangini, *Real time identification of discrete event systems using Petri nets*, Automatica **44** (2008), no. 5, 1209 – 1219.

[DFMU11] M. Dotoli, M.P. Fanti, A.M. Mangini, and W. Ukovich, *Identification of the unobservable behaviour of industrial automation systems by Petri nets*, Control Engineering Practice **19** (2011), no. 9, 958 – 966.

[DHP$^+$93] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat, *Practice of Petri Nets in Manufacturing*, Chapman and Hall, 1993.

[DK98] I. Demongodin and N.T. Koussoulas, *Differential Petri nets: representing continuous systems in a discrete-event world*, IEEE Transactions on Automatic Control **43** (1998), no. 4, 573 –579.

[DPP09] M.A. Drighiciu, A.P. Petrisor, and M. Popescu, *A Petri Nets approach for hybrid systems modeling*, International Journal of Circuits, Systems and Signal Processing (2009).

[ECM95] J. Ezpeleta, J.M. Colom, and J. Martinez, *A Petri net based deadlock prevention policy for flexible man-*

ufacturing systems, IEEE Transactions on Robotics and Automation **11** (1995), no. 2, 173 –184.

[ELS11] J. Esparza, M. Leucker, and M. Schlund, *Learning Workflow Petri Nets*, Fundamenta Informaticae **113** (2011), no. 3-4, 205–228.

[Eur03] European Federation of Material Handling, *Fem 9.851 - performance data of storage and retrieval machines: cycle times*, 2003.

[EVLLM11] A.P. Estrada-Vargas, J.-J. Lesage, and E. Lopez-Mellado, *Stepwise identification of automated discrete manufacturing systems*, 2011 IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA), September 2011, pp. 1 –8.

[EVLLM12] ______, *Identification of industrial automation systems: Building compact and expressive Petri net models from observable behavior*, "2012 American Control Conference (ACC'12), Montréal : Canada (2012), June 2012.

[FCSS99a] K. Feldmann, A. Colombo, C. Schnur, and T. Stöckel, *Specification, design and implementation of logic controllers based on colored Petri net models and the standard IEC 1131 part I: Specification and design*, IEEE Transaction on Control System Technology **7** (1999), no. 6, 657–665.

[FCSS99b] ______, *Specification, design and implementation of logic controllers based on colored Petri net models and the standard IEC 1131 part II: Design and implementation*, IEEE Transaction on Control System Technology **7** (1999), no. 6, 666–674.

[FKR+97] Y. Freund, M. Kearns, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie, *Efficient learning of typical* *finite automata from random walks*, Information and Computation **138** (1997), no. 1, 23–48.

[FS08] M.P. Fanti and C. Seatzu, *Fault diagnosis and identification of discrete event systems using Petri nets*, 9^{th} International Workshop on Discrete Event Systems (WODES'08), Goteborg, Sweden (2008), 432 –435.

[FZ04] M.P. Fanti and MengChu Zhou, *Deadlock control methods in automated manufacturing systems*, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans **34** (2004), no. 1, 5–22.

[GAS05] Latefa Ghomri, Hassane Alla, and Zaki Sari, *Structural and Hierarchical Translation of Hybrid Petri Nets in Hybrid Automata*, IMACS05 (2005).

[GGM10] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis, *Research on warehouse design and performance evaluation: A comprehensive review*, European Journal of Operational Research **203** (2010), no. 3, 539 – 549.

[GHS77] S. C. Graves, W. H. Hausman, and L. B. Schwarz, *Storage-retrieval interleaving in automatic warehousing systems*, Management Science **23** (1977), no. 9, 935–945.

[GS05] A. Giua and C. Seatzu, *Identification of free-labeled Petri nets via integer programming*, Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, dec. 2005, pp. 7639 – 7644.

[GU98] A. Giua and E. Usai, *Modeling hybrid system by high-level Petri nets*, API - JESA **32** (1998), no. 9, 9–10.

[GZNL08] V. Giordano, J.B. Zhang, D. Naso, and F. Lewis, *Integrated supervisory and operational control of a warehouse with a matrix-based approach*, IEEE Transaction on Automation Science and Engineering **5** (2008), no. 1, 53–70.

[HC99] S. Hsieh and Y.F. Chen, *Agvsimnet: A Petri-net-based AGVS simulation system*, The International Journal of Advanced Manufacturing Technology **15** (1999), 851–861.

[HCL07] Shan-Jun He, Fei Cheng, and Jian Luo, *Modeling and implementing of an automated warehouse via Colored Timed Petri Nets; a behavior perspective*, IEEE International Conference on Control and Automation, 2007. ICCA 2007. (2007), 2823 –2828.

[HHC98] S. Hsieh, J.S. Hwang, and H.C. Chou, *A Petri net based structure for AS/RS operation modeling*, Int. Journal of Production Research **36** (1998), no. 12, 3323–3346.

[HK98] S. Hsieh and M.Y. Kang, *Developing AGVS Petri net control models from flowpath*, Journal of Manufacturing Systems **17** (1998), no. 4, 237–249.

[HL09] H. Hu and Z. Li, *Local and global deadlock prevention policies for resource allocation systems using partially generated reachability graphs*, Computers & Industrial Engineering **57** (2009), 1168–1181.

[HMSW87] M. H. Han, L.F. McGinnis, J. S. Shieh, and J. A. White, *On sequencing retrievals in an automated storage/retrieval system*, IEE Transactions **19** (1987), no. 3, 56–66.

[HZL12] H. Hu, M.C. Zhou, and Zhiwu Li, *Liveness and ratio-enforcing supervision of automated manufacturing systems using Petri nets*, IEEE Transactions on

Systems, Man, and Cybernetics-Part A: Systems and Humans **42** (2012), no. 2, 392–403.

[Jar10] D. E. Jarvis, *An identification technique for timed event systems*, Discrete Event Systems, 2010. WODES'10. 10th International Workshop on (Berlin, Germany), vol. 10, September 2010, pp. 191–196.

[Jen95] K. Jensen, *Colored Petri nets. basic concepts, analysis methods and pratical use. volume 1*, Monographs on Theoretical Computer Science, Springer Verlag, New York, 1995.

[KLL05] Stephane Klein, Lothar Litz, and Jean-Jacques Lesage, *Fault detection of discrete event systems using an identification approach*, 16th IFAC world Congress, Praha, Tchèque, République, 2005.

[LdSO96] S.G. Lee, R. de Souza, and E.K. Ong, *Simulation modelling of a narrow aisle automated storage and retrieval system (AS/RS) serviced by rail-guided vehicles*, Computers in Industry **30** (1996), no. 3, 241 – 253.

[LML10] Ana Paula Estrada-Vargas Ernesto Lopez-Mellado and Jean-Jacques Lesage, *A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems*, Mathematical Problems in Engineering (2010).

[MCLM05] M. E. Meda-Campana and E. Lopez-Mellado, *Identification of concurrent discrete event systems using Petri nets*, Proceedings of IMACS world congress, 2005, pp. 11–15.

[Mur89] T. Murata, *Petri Nets: Properties, analysis and applications*, Proceedings of IEEE **77** (1989), no. 4, 541–580.

[PL95] S. Pettersson and B. Lennartson, *Hybrid modelling focused on Hybrid Petri Nets*, in 2nd European Workshop on Real-time and Hybrid systems (1995), 303–309.

[pne] *PNetLab*, by Automatic Group of Salerno, `http://www.automatica.unisa.it/PnetLab.html`.

[Ros02] Elszbieta Roszkowska, *Undirected colored Petri net for modelling and supervisory control of AGV systems*, Proceedings of the 6-th International Workshop on Discrete Event Systems (WODES'02) (2002).

[RR08] E. Roszkowska and S.A. Reveliotis, *On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems*, IEEE Transactions on Automatic Control **53** (2008), no. 7, 1689 –1695.

[RRS$^+$00] B. Rouwenhorst, B. Reuter, V. Stockrahm, G.J. van Houtum, R.J. Mantel, and W.H.M. Zijm, *Warehouse design and control: Framework and literature review*, European Journal of Op. Research **122** (2000), no. 3, 515 – 533.

[RS] R.L. Rivest and R.E. Schapire, *Inference of finite automata using homing sequences.*

[RS94] R. L. Rivest and R. E. Schapire, *Diversity-based inference of finite automata*, Journal of the ACM (JACM) **41** (1994), no. 3, 555–589.

[RW89] P.J. Ramadge and W.M. Wonham, *The control of discrete event systems*, Proc. of IEEE **77** (1989), no. 1, 637–659.

[SHM11] Bernhard Steffen, Falk Howar, and Maik Merten, *Introduction to active automata learning from a practical perspective*, Formal Methods for Eternal Networked Software Systems **6659** (2011), 256–296.

[TL97] D.A. Tacconi and F.L. Lewis, *A new matrix model for discrete event systems: application to simulation*, IEEE Control Systems Magazine **17** (1997), 62–71.

[TS93] E. Teruel and M. Silva, *Liveness and Home States in Equal Conflict Systems*, vol. 691, pp. 415–432, Springer, 1993.

[TTV06] George J. Tsinarakis, Nikos C. Tsourveloudis, and Kimon P. Valavanis, *Modeling, analysis, synthesis, and performance evaluation of multioperational production systems with hybrid timed Petri nets*, IEEE Transaction on Automation Science and Engineering **3** (2006), no. 1, 29–46.

[Van99] J.P. Van den Berg, *A literature survey on planning and control of warehousing systems*, IIE Transactions **31** (1999), 1–13.

[WCCZ10] N. Q. Wu, F. Chu, C. B. Chu, and M. C. Zhou, *Hybrid Petri net modeling and schedulability analysis of high fusion point oil transportation under tank grouping strategy for crude oil operations in refinery*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **40** (2010), no. 2, 159–175.

[WCZ09] N. Q. Wu, F. Chu, and M. C. Zhou, *Short-term schedulability analysis of multiple distiller crude oil operations in refinery with oil residency time constraint*, IEEE Transactions on Systems, Man, and Cybernetics, Part C **39** (2009), no. 1, 1–16.

[WZC08] N. Q. Wu, M. C. Zhou, and F. Chu, *A Petri net-based heuristic algorithm for realizability of target refining schedule for oil refinery*, IEEE Transactions on Automation Science and Engineering **5** (2008), no. 4, 661–676.

[XH11] Jingjing Xue; and Dawei Hu, *Modeling of logistics warehousing system based on Timed Petri Net*, 11th International Conference of Chinese Transportation Professionals (ICCTP). Nanjing, China. (2011).

[XWW$^+$07] Xiaowei Xu, Zhiyan Wang, Yanyan Wang, Xiaoye Cao, Yinghong Liang, and Yaohua Wu, *A novel modeling design method for automated storage and retrieval system based on Petri nets*, IEEE International Conference on Automation and Logistics, (2007).