Implementing a Multiple Specification Regulation Controller: A Case of Study*

R. Campos
Cinvestav del IPN
L. Mateos Sur 590
Guadalajara, Jal, México.

Abstract — This paper presents the implementation of a controller for an advanced Manufacturing Cell (MC) that uses a decision making machine to track several model specifications. It follows an incremental approach, where the individual controllers (which are designed for isolated specifications) are used to obtain the global controller. The global controller is the combination of all specifications working in concurrence. Such concurrence / combinatorial approach leads to efficient algorithms and computation towards optimal performance of the global controller.

Keywords: Manufacturing Cells, Discrete Event Systems, Regulation Control.

1 Introduction

The problem of controlling a manufacturing cell has been solved using different approaches. For instance supervision control [11], regulation control [9], and monitor control [6]. In supervisory control theory (SCT), the system and the specification models are given in the form of formal languages. Unfortunately, supervisory control theory solves this problem using all the words of the system language. If several specifications must work in concurrence, supervisory control theory must explore the synchronous product of all the specifications models, leading to a state explosion problem [12].

In regulation control theory (RCT), the system and the specification (called reference model) are given in the form of Petri nets (PN), in such a way that the specification is a high-abstraction view of the system. That is, the specification shows “what to do” instead of “how to do”. When several specifications are given, the states produced by individual specifications evolving in concurrence must be computed, leading to a computational complex problem. Fortunately, the approach presented in a companion paper [10] allows to build a global controller using an incremental approach, avoiding the computational problem.

2 Interpreted Petri Nets

This section introduces the PN and interpreted PN (IPN) basic concepts and notation used through this paper. Fine PN references are [7] and [2]. Some aspects of IPN can be found in [4].

Definition 1 A Petri Net structure Q is a bipartite digraph represented by the 4-tuple Q = (P, T, I, O) where:

- P = {p₁, p₂, ..., pₙ} is a finite set of vertices called places,
- T = {t₁, t₂, ..., tₘ} is a finite set of vertices called transitions,
- I : P × T → Z⁺ is a function representing the weighted arcs going from places to transitions,
- O : P × T → Z⁺ is a function representing the weighted arcs going from transitions to places, where Z⁺ is the set of nonnegative integers.

Typical conventions are: the symbol *t_j denotes the set of all places p_i such that I(p_i, t_j) ≠ 0 while t_j* denotes the set of all places p_i such that O(p_i, t_j) ≠ 0. Analogously, *p_i denotes the set of all transitions t_j such that O(p_i, t_j) ≠ 0 and p_i* the set of all transitions

*Supported by NSF-CONACyT DNI-0219195 grant from USA. 0-7803-8566-7/04/$20.00 © 2004 IEEE.
t_j such that I(p_i,t_j) ≠ 0. Pictorially, places are represented by circles, transitions are represented by rectangles, and arcs are depicted as arrows.

The pre-incidence matrix of Q is $C^- = [c^-_{ij}]$, where $c^-_{ij} = I(p_i,t_j)$; the post-incidence matrix of G is $C^+ = [c^+_{ij}]$, where $c^+_{ij} = O(p_i,t_j)$; and the incidence matrix of Q is $C = C^+ - C^-$. The marking function $M : P \rightarrow Z^+$ is a mapping from each place to the nonnegative integers representing the number of tokens (depicted as dots) residing inside each place. The marking of a PN is usually expressed as an n-entry vector.

**Definition 2** A Petri Net system or Petri Net (PN) is the pair $N = (Q,M_0)$, where $Q$ is a PN structure and $M_0$ is an initial token distribution.

In a PN system, a transition $t_j$ is enabled at marking $M_k$ if $\forall p_i \in P, M_k(p_i) \geq I(p_i,t_j)$; an enabled transition $t_j$ can be fired reaching a new marking $M_{k+1}$ which can be computed as $M_{k+1} = M_k + C v_k$, where $v_k(i) = 0$, $i \neq j$, $v_k(j) = 1$, this equation is called the PN state equation. The reachability set of a PN is the set of all possible reachable marking from $M_0$ firing only enabled transitions; this set is denoted by $R(Q, M_0)$.

This work uses Interpreted Petri Nets (IPN) [4], an extension to PN, since they allow to associate input and output signals to PN models. Formally IPN are defined as follows.

**Definition 3** An Interpreted Petri Net (IPN) is the 4-tuple $(N, \Sigma, \lambda, \varphi)$ where:

- $N = (Q,M_0)$ is a PN system.
- $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the input alphabet of the net, where $\alpha_i$ is an input symbol.
- $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ is a labelling function of transitions with the following constraint: $\forall t_j, t_k \in T, j \neq k$, if $\forall p_i, I(p_i, t_j) = I(p_i, t_k) = 0$ and both $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$. In this case $\varepsilon$ represents an internal system event.
- $\varphi : R(Q, M_0) \rightarrow (Z^+)^r$ is an output function, that associates to each marking in $R(Q, M_0)$ an output vector. Here $q$ is the number of outputs.

Remarks:

1. In this work, $(Q,M_0)$ will be used instead of $(N,\Sigma,\lambda,\varphi)$ to emphasize the fact that there is an initial marking in an IPN.
2. Attention is focused on the case when function $\varphi$ is a $q \times n$ matrix, where $q$ is the number of places representing measurable states in the DES and $n$ is the number of places in the model $(Q,M_0)$. Each column of this matrix is an elementary or null vector. If the output symbol $i$ is present (turned on) every time that $M(p_i) \geq 1$, then $\varphi(i,j) = 1$, otherwise $\varphi(i,j) = 0$.

3. Equivalent transitions are not allowed, i.e. it is assumed that $\forall t_i, t_j$ such that $t_i \neq t_j, \lambda(t_i) = \lambda(t_j)$, it holds that $C(\cdot,i) \neq C(\cdot,j)$. This is not a major constraint because those transitions are redundant.
4. A Strongly Connected State Machine (SCSM) is a PN (IPN) in which each of its transitions fulfills $|t_j| = 1 = |t_j^*|$, i.e., it has only one input place and one output place [2].

A transition $t_j \in T$ of an IPN is enabled at marking $M_k$ if $\forall p_i \in P, M_k(p_i) \geq I(p_i,t_j)$. If $\lambda(t_j) = \alpha_i \neq \varepsilon$ is present and $t_j$ is enabled, then $t_j$ must fire. If $\lambda(t_j) = \varepsilon$ and $t_j$ is enabled then $t_j$ can be fired. When an enabled transition $t_j$ is fired in a marking $M_k$, then a new marking $M_{k+1}$ is reached. This fact is represented as $M_k \xrightarrow{t_j} M_{k+1}$ and $M_{k+1}$ can be computed using the dynamic part of the state equation:

$$M_{k+1} = M_k + C v_k$$

where $C$ and $v_k$ are defined as in PN and $y_k \in (Z^+)^q$ is the $k-$th output vector of the IPN.

IPN can be classified according to some characteristics. For instance, if the resulting graph is strongly connected, then the IPN becomes a strongly connected one. When the number of tokens inside each place is one or zero at any reachable marking, the IPN is called a binary one. If each transition have just one input place and one output place, then the IPN is named a State Machine.

Now, the following definitions relate the input and output symbol sequences with the firing transition sequences and input or output languages.

**Definition 4** A firing transition sequence of an IPN $(Q,M_0)$ is a transition sequence $\sigma = t_1 t_2 \ldots t_k \ldots$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_w \xrightarrow{t_k} \ldots$.

**Definition 5** The set $\mathcal{L}(Q,M_0)$ of all firing transition sequences is called the firing language of $(Q,M_0)$, $\mathcal{L}(Q,M_0) = \{\sigma | \sigma = t_1 t_2 \ldots t_k \ldots \land M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_w \xrightarrow{t_k} \ldots \}.$

**Definition 6** The set $\mathcal{L}_{in}(Q,M_0)$ of all input sequences is called the input language of $(Q,M_0)$, $\mathcal{L}_{in}(Q,M_0) = \{\lambda(t_1) \lambda(t_2) \ldots \lambda(t_k) | t_1 t_2 \ldots t_k \in \mathcal{L}(Q,M_0)\}$.

**Definition 7** The set $\mathcal{L}_{out}(Q,M_0)$ of all output sequences is called the output language of $(Q,M_0)$, $\mathcal{L}_{out}(Q,M_0) = \{\varphi(M_0) \varphi(M_1) \ldots \varphi(M_w) | M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_w \xrightarrow{t_k} \ldots, t_1 t_2 \ldots t_k \in \mathcal{L}(Q,M_0)\}.$

Then two more languages can be defined.
Definition 8 Let $L$ be a language and $\omega \in L$ a word of $L$. The prefix set of $\omega$ is $\omega = \{ \omega' | \exists v \text{ such that } \omega v = \omega \}$. In a similar way, the prefix set of a language $L$ is $L = \{ z | z \in L \lor \omega \in L \}$.

Definition 9 Let $L$ be a language. The mid set of $L$ is $L_{mid} = \{ \omega | \exists z \in L, v \in L, z \text{ could be empty strings} \}$.

Definition 10 Let $\sigma = t_1 t_2 \ldots$ be a firing transition sequence. The Parikh vector $\vec{\sigma} : T \rightarrow (\mathbb{Z}^+)^n$ of $\sigma$ maps every transition $t \in T$ to the number of occurrences of $t$ in $\sigma$.

According to functions $\lambda$ and $\varphi$, transitions and places of an IPN $(Q, M_0)$ can be classified as follows.

Definition 11 If $\lambda(t_i) \neq \varepsilon$ the transition $t_i$ is said to be manipulated. Otherwise it is nonmanipulated. A place $p_i \in P$ is said to be measurable if the $i$ - $\varepsilon$ column vector of $\varphi$ is not null, i.e. $\varphi(\bullet, i) \neq 0$. Otherwise it is nonmeasurable.

3 Trace Equivalent Petri Nets

The following relations are used to define the trace equivalence property.

Definition 12 Let $(Q, M_0)$ be an IPN and $R(Q, M_0)$ be its reachability graph. $\forall M_i, M_j \in R(Q, M_0), M_i$ is $T(Q, M_0)$ related with $M_j$, $M_i \sim_{T(Q, M_0)} M_j$, iff there exist $t_q \in T$ (the set of transitions of $(Q, M_0)$) such that $M_i \xrightarrow{t_q} M_j$.

Definition 13 Let $(Q, M_0)$ be an IPN and $R(Q, M_0)$ be its reachability graph. $\forall M_i, M_j \in R(Q, M_0), M_i$ is $T(Q, M_0)^*$ related with $M_j$, $M_i \sim_{T(Q, M_0)^*} M_j$, iff there exist $t_q t_r \ldots t_s \in T$ (the set of transitions of $(Q, M_0)$) such that $M_i \xrightarrow{t_q} M_j \xrightarrow{t_r} \ldots \xrightarrow{t_s} M_j$.

Definition 14 Let $(S_f, M_0)$ and $(R_m, M_0)$ be two IPN and $R(S_f, M_0), R(R_m, M_0)$ be their reachability sets, respectively. It is said that $(S_f, M_0)$ is trace equivalent with $(R_m, M_0)$ if there exists a function $\Pi : R(R_m, M_0) \rightarrow R(S_f, M_0)$ such that:

$$\forall t_j \in T(S_f, M_0)^*, \exists \Pi \subseteq \Pi_0 \sim_{T(R_m, M_0)}$$

In other words, if the diagram depicted in figure 1 is an inclusion.

Now, a constraint can be imposed to the trace equivalence concept. It deals with including the constraint $\varphi_{R_m}(M_j) = \varphi_{S_f}(M_j)$ when $\Pi(M_j) = M_j$. Roughly speaking, it means that when we only take into account the output of markings $\Pi(M_j)$, then the output language of both nets, $(R_m, M_0)$ and $(S_f, M_0)$, must be equal.

Definition 15 Let $(S_f, M_0)$ and $(R_m, M_0)$ be two IPN, and $R(S_f, M_0), R(R_m, M_0)$ be their reachability sets, respectively. $(S_f, M_0)$ is output-trace equivalent with $(R_m, M_0)$ if $(S_f, M_0)$ is trace equivalent with $(R_m, M_0)$ and

$$\forall M_j \in R(R_m, M_0), \varphi_{R_m}(M_j) = \varphi_{S_f}(\Pi(M_j))$$

A characterization of output-trace equivalence property, focuses in the case when the specifications can be represented by binary strongly connected state machines (binary SCSM) and function $\Pi$ is a linear function, is presented in a companion paper [10]. The next equations are for representing the system and specification models.

$$S_f = \begin{cases} M_{n+1} = M_n + C \tilde{u}_n \\ \tilde{y}_n = \varphi_{S_f}(M_n) \end{cases}$$

$$R_m = \begin{cases} \tilde{M}_{k+1} = \tilde{M}_k + Q \tilde{x}_k \\ \tilde{y}_k = \varphi_{R_m}(\tilde{M}_k) \end{cases}$$

Corollary 1 Let $(S_f, M_0)$ and $(R_m, \tilde{M}_0)$ be two IPN represented by the equation (4), where $(R_m, \tilde{M}_0)$ is a binary SCSM. Assume that there exists a linear function $\Pi : R(R_m, \tilde{M}_0) \rightarrow R(S_f, M_0)$, such that:

1. $\Pi \tilde{M}_0 = M_0$.
2. $\forall t_j \in T, \exists \omega_j \in L(S_f, \Pi(\tilde{M}_j))$ such that $\Pi Q \tilde{t}_j = \tilde{C} \omega_j$.
3. $\varphi_{R_m} = \varphi_{S_f} \Pi$.

then $(S_f, M_0)$ is output-trace equivalent with $(R_m, \tilde{M}_0)$.

Definition 16 Let $(S_f, M_0)$ and $(R_m, \tilde{M}_0)$ be two IPN, where $(S_f, M_0)$ is output-trace equivalent with $(R_m, \tilde{M}_0)$ and $\Pi : R(R_m, \tilde{M}_0) \rightarrow R(S_f, M_0)$ is a linear function. The net

$$R_m = \begin{cases} \Pi \tilde{M}_{k+1} = \Pi \tilde{M}_k + \Pi Q \tilde{x}_k \\ \tilde{y}_k = \varphi_{S_f} \Pi \tilde{M}_k \end{cases}$$
is named the projection of \((R_m, \tilde{M}_0)\) over the places of \((S_f, M_0)\).

4 Multiple Specification Regulation Control

This is the classical definition of controllability presented in [8].

Definition 17 Let \((S_f, M_0)\) be an IPN and \(K \subseteq \mathcal{L}(S_f, M_0)\) be the specification language. The language \(K\) is controllable with respect to \((S_f, M_0)\) if \(\forall t_k \in T_{NM}; (i.e., \lambda(t_k) = \varepsilon)\) it holds that \(\tilde{R}_k \cap \mathcal{L}(S_f, M_0) \subseteq \tilde{K}\).

The multiple specification regulation control problem is stated in the following definition.

Definition 18 Let \(\{R_f, M_0\}, \ldots, (R_l, M_0)\) be a set of binary SCSM, where \((S_f, M_0)\) is output-trace equivalent with each \((R_i, M_0)\). Let \((S_f, M_0)\) be an IPN. The sets \(R(S_f, M_0)\) and \(R(R_i, M_0)\) are the reachability sets of \((S_f, M_0)\) and \((R_i, M_0)\) respectively.

The full information regulation control problem consists in finding out a partial function (controller) \(H_i : \mathcal{R}(S_f, M_0) \times \mathcal{R}(R_i, M_0) \times \tilde{T} \to \mathcal{L}_{mid}(S_f, M_0)\) where \(H_i(\Pi(M_j), M_j, t_k) = w_k\) and \(w_k\) is controllable in \((S_f, M_0)\) and \((R_i, M_0)\) respectively.

The multiple specification regulation control problem consists in finding out a partial function (controller) \(H_G : \mathcal{R}(S_f, M_0) \times \mathcal{R}(R_i, M_0) \times \tilde{T} \to \mathcal{L}_{mid}(S_f, M_0)\) where \(H_G(\Pi(M_j), \tilde{M}_j, t_k) = w_k\) and \(w_k\) is controllable in \((S_f, \Pi(M_j))\).

Figure 2 presents the proposed control scheme for implementing the full information regulation control problem.

Results of previous section can be used to characterize when the full information output regulation problem has a solution.

Theorem 2 Let \((S_f, M_0)\) and \((R_m, \tilde{M}_0)\) be two IPN represented by the equation (4), where \((R_m, \tilde{M}_0)\) is a binary SCSM. Assume that there exists a linear function \(\Pi : R(R_m, \tilde{M}_0) \to R(S_f, M_0)\), such that:

1. \(\Pi \tilde{M}_0 = M_0\).
2. \(\forall t_j \in \tilde{T}, \exists w_j \in \mathcal{L}(S_f, \Pi(M_j))\) such that \(\Pi Q \tilde{t}_j = C \tilde{w}_j\) and \(w_j\) is controllable with respect to \((S_f, \Pi(M_j))\).
3. \(\varphi_{R_m} = \varphi_{S_f} \circ \Pi\).

where \(\circ\) is the composition function operator. Then

the full information output regulation problem has a solution.

Notice that finding out function \(\Pi\) and sequences \(w_j\) are enough to compute function \(H\). Next section is devoted to compute function \(\Pi\) and sequences \(w_j\).

5 Controller design

In a companion paper [10], the following algorithms are proposed to solve the full information regulation control problem.

Computation of a linear function \(\Pi\).

1. Solve the following linear programming problems:

\[
\begin{align*}
LPP = \min \quad & \sum_{t_j} \Pi_{t_j} + \sum_{n=1}^{m} \sum_{k=1}^{l} w_m(n) \\
\text{s.t.} \quad & \Pi \cdot M_0 = M_0 \\
& \forall t_j \in \tilde{T}, \Pi \cdot Q \tilde{t}_j = C \tilde{w}_m \\
& \varphi \cdot \Pi = \varphi
\end{align*}
\]

The solution of this problem is the function \(\Pi\) and the firing vectors \(\tilde{w}_m\) (one vector for each transition of \((R_m, \tilde{M}_0)\)).

If the previous LLP problem has no solution, then there exist no linear function \(\Pi\) that solves this problem. When a LLP problem has a solution, then function \(\Pi\) and vectors \(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_m\) are computed. These vectors are used to compute fireable sequences \(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_m\) as follows.

Since \(\Pi(\tilde{M}_0) = M_0\), the system and reference model produce the same output. Assume that \(\tilde{t}_j\) is enabled in \(R(R_m, \tilde{M}_0)\) and \(\tilde{M}_m \xrightarrow{\tilde{t}_j} \tilde{M}_{m+1}\), then the Parikh vector \(\tilde{w}_j\) is computed. Afterwards, this vector is converted into a fireable sequence \(\sigma_{w_j}\). In order to obtain this word \(\sigma_{w_j}\), such that \(M_n \xrightarrow{\sigma_{w_j}} M_{n+1}\), and \(\varphi_{S_f}(M_{k+1}) = \varphi_{R_m}(\tilde{M}_{k+1})\), any search algorithm can be applied to partially expand \(R(S_f, M_n)\). We uses a
depth search algorithm constrained to the fact to the Parikh vector of \( \sigma_{u_i} \) should be equal to \( w_i \). Thus, we have the next algorithm.

Computation of controller \( H \).

Let \( M_k \xrightarrow{t_i} M_{k+1} \) and \( M_k \xrightarrow{w_i} M_{k+1} \) be the markings actual of both the reference model and system respectively, which fulfilled that \( \varphi_{S_j}(M_{k+1}) = \varphi_{R_m}(M_{k+1}) \) and \( w_i = t_0 t_1 \ldots t_z \).

1. Then \( M_k \xrightarrow{t_i} M_{k'} \xrightarrow{t_i} \ldots M_{k''} \xrightarrow{t_i} M_{k+1} \), and \( H \) can be computed as:

\[
H(M_k, \tilde{M}_{k+1}, \lambda(t_i)) = \lambda(t_0) \\
H(M_k', \tilde{M}_{k+1}, \lambda(t_i)) = \lambda(t_0) \\
\vdots \\
H(M_k'', \tilde{M}_{k+1}, \lambda(t_i)) = \lambda(t_0) \\
H(M_{k+1}, \tilde{M}_{k+1}, \lambda(t_i)) = \lambda(t_0) \\
H(M_{k+1}, \tilde{M}_{k+1}, \lambda(t_i)) = \lambda(t_0)
\]

The solution to the multiple specification regulation control problem is tested in the following algorithm. It assumes that functions \( \Pi_z \) and sequences \( \sigma_{w_i} \) were previously computed using the algorithm 5 and a partial expansion algorithm.

Solving the Multiple Specification Problem

1. Compute the \( \Pi_1 R^m_1 \) and \( \Pi_2 R^m_2 \). Compute the sets \( S_1 \) and \( S_2 \) of places belonging to the p-semiflows of \( \Pi_1 R^m_1 \) and \( \Pi_2 R^m_2 \) respectively, using equation (5). They are very easy to compute, since \( R^m_1 \) and \( R^m_2 \) are SCSM IPN. Places of \( \Pi_2 R^m_1 \) and \( \Pi_2 R^m_2 \) also belong to \((S_f, M_0)\). Finally, compute \( M_{GC} = [\tilde{M}_0] \).

2. The set of places \( P \) of \((S_f, M_0)\) is computed as follows \( P = \{P_1 \cup P_{R_1} \cup P_{R_2} \cup P_C\} \), where \( P_1 = \{p_i | p_i \in S_1 \cap S_2\} \), \( P_{R_1} = \{p_i | p_i \in S_1 - P_1\} \), \( P_{R_2} = \{p_i | p_i \in S_2 - P_1\} \) and \( P_C = P - (P_1 \cup P_{R_1} \cup P_{R_2}) \).

3. A general function \( \Pi_G \) of \( n \times q_1 \) and \( \Pi_2 \) of \( n \times q_2 \) is computed using \( \Pi_1 \) of \( n \times q_1 \) and \( \Pi_2 \) of \( n \times q_2 \):

\[
\Pi_G(i, j) = \begin{cases} 
\Pi_1(i, j) & \text{if } p_i \in P_1 \cup P_{R_1} \cup P_C \text{ and } 1 \leq j \leq q_1 \\
\Pi_2(i, j) & \text{if } p_i \in P_1 \cup P_{R_2} \text{ and } q_1 + 1 \leq j \leq q_2 \\
0 & \text{otherwise} 
\end{cases}
\]

4. If \( \Pi_G(M_{GC}) \leq M_0 \) (the initial marking of the system), then the Multiple Specification Problem has solution using the previous computed sequences, i.e. the previous computed \( \sigma_{w_i} \) can be used.

Previous results can be tested in a similar way to theorem 2.

6 Example: The IMH Cell

An example of a regulation controller for an isolated specification was presented in [1]. Now, this example is extended to the case when two specifications must be tracked in concurrence. This example was implemented in the IMH cell at the UTA's ARRI [3], using the software named LabWindows.

6.1 IMH Cell Description

The IMH cell at UTA's ARRI is composed by three robots, three conveyors, ten sensors, and two machines. The figure 3 depicts the layout for the IMH cell. The robot defined as \( R_1 \), a CRS robot, can perform four different tasks: two tasks relate to picking up parts type A and B from the input parts area to be placed over conveyor denominated B1. The other two tasks are related to picking up final products A and B from conveyor B3 to the output area. The Puma robot \( R_2 \), performs three different tasks: picks up parts A from conveyor B1 to M1 (machine one), picks up parts B from conveyor B1 to conveyor B2, or picks up parts A from M1 to be placed over conveyor B2. The Adept robot \( R_3 \), also performs three different tasks: picks up parts A from conveyor B3, to conveyor B2, picks up parts B from conveyor B2 to M2 (machine two), or picks up parts B from M2 to be placed over conveyor B3. Machines M1 and M2 are simulated by activating valve-air cylinders. The outputs form the cell, measured by sensors, are arranged in as vectors \( v \) and \( r \), that measure the ready jobs and ready resources, respectively.

6.2 IMH Cell modelling

A PN model for the IMH cell can be derived using the matrices presented in [3]. The job sequencing matrix \( F_v \) captures the job relationships; the resource assignment matrix \( F_r \), captures the material requirement for each job; and the input part matrix \( F_u \) captures the cell
incoming parts. In these matrices, $F_o(i,j) = 1$ means that job $j$ is performed before job $i$ is done; $F_r(i,j) = 1$ means that resource $j$ is required to perform job $i$; and $F_i(i,j) = 1$ means that incoming part $j$ is ready to be processed by the job $i$.

Three additional matrices are posed in a similar way: start equation matrix $S_e$, capturing the enabling conditions for starting jobs; resource release matrix $S_r$, indicating the resources that must be released when an activity is finished; and finally the output part matrix $F_o$, indicating when a part is finished and must be moved to an output store. Such a model results controllable. Thus previous controller scheme can be used to control the IMH cell.

Figure 4 depicts the $PN$ graph. The Robot $R_i$ performs different tasks $U_j$, called $R_iU_j$ activities, that are represented as upper labels for the places in the net. The lower label for the place determines its index. Thus, when a token is posed in the place $p_3$, upper label $R_1U_1$, this means that the robot $R_1$ is performing its job $U_1$, i.e., picking up parts type $A$ and $B$ from the input parts area. In a similar way for all of the remaining robots and all the machines. The bold labels $A, B, C, D, E$, and $F$ at the bottom of the places defines the output signals that will be used in the reference models. Finally, the labels $X_i$ over the transitions determines its index.

### 6.3 IMH cell implementation

A LabWindows implementation for the IMH cell Discrete Event Controller (DEC) was developed. Figure 5 depicts a block diagram. LabWindows text based programming environment works on ANSI C platform. It allows multiple thread processes to interact simultaneously in supervisory applications. In this implementation, this multi-thread capability is used to communicate between different resources at the same time. The process this work is interested in are jobs for different robots, machining jobs and transferring parts using conveyors, i.e. sequence jobs implemented in manufacturing processes.

In figure 5 it is presented inside the dashed lines, the

DEC for the IMH cell implemented on a PC using LabWindows. The DEC is responsible for firing the transitions. Once the transitions to be fired are done, the main program shifts to reads sensors for keeping a track of the jobs being performed by the robots and the machines. Then it communicates with the other five implemented threads for the three robots and two machines as shown in figure 5.

This PC-based IMH controller has three serial ports that interact with the three robots of the IMH cell, as well as a DAQ card. The DAQ card receives discrete signals from capacitive proximity sensors, which sense parts within the IMH cell, and also sends discrete signals to the machines to initialize jobs. Once the job is done by the resources, the program function reads and sets signals from and to the resources sensors.

Using the capability of LabWindows to read ANSI C programming language, the algorithms in section 5 was implemented as a stand-alone module. Then the scheme depicted in figure 2 is used to put all the LabWindows programs together.

### 6.4 Multiple Specification

Two independent specification were proposed: Specification A and Specification B depicted in figure 6. The
specification A enforces the system to fabricates only A products, while the specification B enforces the system to fabricates only B products. There exists a controller for each specification [10]. Individual II functions for each specification are depicted in figure 7.

It is easy to see, staring at figure 4, that the two specifications in running concurrence are not allowed in the IMH cell, since there are only one robot R1, R2 and R3. Thus, the there exist two approaches: either, to add extra resources, in this case extras robots R1, R2, and R3, or to serialize the specifications [10]. The obvious solution in this case is to serialize the specifications. Then the previous functions II1 and II2, are involved in computing a new II function that is the solution for the serialization specification. The new function is depicted in figure 8. The serialized specification is in fact, allowed in the system. Figure 9 depicts the Gantt diagram for the states of the Robot R1 during the execution of the final specification.

7 Conclusions

This work addressed the regulation control problem when a system must track several specifications working in concurrence. It was showed that the algorithms used to solve this problem can be implemented in a real FMS. The IMH cell at the UTA's ARRI was the platform in which our controller was tested. It was implemented using LabWindows text base programming environment and ANSI C code. The results, depicted as Gantt diagrams, shows that the IMH cell works as the reference model specifies.
References


