Some Results on Petri Net Languages

STÉPHANE LAFORTUNE AND HYUCK YOO

Abstract—**We compare various classes of Petri net languages. We present a new constructive proof of the equivalence, from the point of view of descriptive power, of "general" Petri nets and "restricted" Petri nets (no multiple arcs nor self-loops are allowed in the latter class). We also comment on the descriptive power of Petri nets versus that of finitely recursive processes.**

I. Introduction

Finite-state machines (FSM), Petri nets (PN), and finitely recursive processes (FRP) are three classes of logical discrete event models that are currently being used for the modeling, analysis, and control of discrete event systems (DES). Ramadge and Wonham have proposed a control theory for DES based on controlled state machines and formal languages [8]. Petri nets have been widely used as a modeling tool for DES (see the references in [1]); Krogh *et al.* have discussed the control of PN in [6] and [4]. More recently, Inan and Varaiya have extended the work of Hoare [3] and proposed the FRP model for DES [5]. Each of these models has its own merits and drawbacks. From the point of view of descriptive power, or language complexity, it is well known that FRP are more general than PN, which in turn are more general than FSM.

It must be remembered when discussing the language complexity of PN that there are many kinds of PN as well as there are many different ways of defining Petri net languages (PNL). Specifically, the class of restricted PN (RPN) does not allow multiple arcs and self-loops in the PN structure; the class of general PN (GPN) allows for such occurrences. Concerning PNL, Peterson presents four different ways of defining the language generated by a PN and three different labelings of the transitions, resulting in 12 types of PNL [7]!

The purpose of this note is to clarify the relationships between some important classes of PNL, in particular, concerning languages generated by GPN and RPN. PNL have been studied in [7] and [2]. In [7], GPN and RPN are proved equivalent for languages that allow ϵ-labeling of transitions (ϵ is the empty string) but not for "ϵ-free" languages. Hack considered many properties of PNL in [2]. However, his proof technique for the problem that we address in Section III is not constructive but indirect, and we have found that it is incomplete concerning the elimination of "bounded closed subnets with multiple arcs" [2, p. 72]. In view of these observations, we provide in Section III a new constructive proof of the equivalence of GPN and RPN for languages that do not allow ϵ-labeling of transitions, thus generalizing existing results on this topic. Finally, in Section IV, we compare PNL to the sets of traces generated by FRP. We extend the proof of Fact 3.7 in [5] to handle GPN and noninjective labeling of transitions.

II. Classes of Petri Net Languages

For the sake of brevity, we assume that the reader is familiar with the definition and operation (i.e., firing) of a PN (see, e.g., [7]). Our notation and terminology are as follows. A PN structure is a five-tuple $C = (P, T, A, I, O)$ that describes the bipartite graph constituting the PN. P is the set of places, T the set of transitions, and A the set of directed arcs between T and P. Let 2^A denote the power set of A. $I: T \times P \to 2^A$ and $O: T \times P \to 2^A$ are functions that describe connections from places to transitions for I, and from transitions to places for O. It is convenient to index the elements of P and T by integers $1 \leq j \leq |P|$ and $1 \leq i \leq |T|$, respectively.

A PN structure C must satisfy the following assumptions: i) there should be no isolated node in the bipartite graph; ii) an arc connects only one (t_i, p_j) pair, i.e., given any $a \in A$, there exists a unique pair (t_i, p_j) such that $a \in I(t_i, p_j)$ or $a \in O(t_i, p_j)$.

An "ordinary PN"[1] satisfies $|I(t_i, p_j)| \leq 1$ and $|O(t_i, p_j)| \leq 1$ $\forall t_i \in T$, $\forall p_j \in P$, i.e., it has at most one arc from a given place to a given transition and one arc from a given transition to a given place (no "multiple arcs"). Let

$$I(t_i) := \{p_j \in P | I(t_i, p_j) \neq \emptyset\} \tag{2.1}$$

$$O(t_i) := \{p_j \in P | O(t_i, p_j) \neq \emptyset\}. \tag{2.2}$$

A "self-loop free" PN satisfies $I(t_i) \cap O(t_i) = \emptyset$ $\forall t_i \in T$, i.e., there are no directed cycles of length two in the bipartite graph. RPN are self-loop free ordinary PN. The class of GPN allows both multiple arcs and self-loops and is the one usually meant when one simply says "Petri net." It is convenient to denote by $\mathcal{P}$ the class of GPN, and by $\mathcal{P}^{NS}$ the class of self-loop free GPN, and by $\mathcal{P}^R$ the class of RPN.

Using notation analogous to that employed for FSM, we define a PN to be a five-tuple $N = (C, \Sigma, \sigma, \mu_0, F)$ where

- C is a PN structure (P, T, A, I, O);
- Σ is the alphabet of symbols for transition labeling; we require that $\epsilon \notin \Sigma$ (this is the so-called ϵ-free labeling case);
- $\sigma: T \to \Sigma$ is the transition labeling function, which need not be injective;
- $\mu_0 \in \mathbb{N}^{|P|}$ is the initial marking vector of the places, i.e., the initial number of tokens in each place;
- $F \subset \mathbb{N}^{|P|}$ is the set of "final" markings that represent "completed" tasks (F need not be finite).

Let $\delta_N: \mathbb{N}^{|P|} \times T \to \mathbb{N}^{|P|}$ be the (partial) marking transition function corresponding to the admissible firings of N. We extend this function to $\delta_N: \mathbb{N}^{|P|} \times T^* \to \mathbb{N}^{|P|}$ in the usual manner. Here, T^* is the Kleene closure of T; also, $\delta_N(\mu, \epsilon) = \mu$. Similarly, we extend the transition labeling function σ to $\sigma: T^* \to \Sigma^*$; thus $\sigma(s) = \epsilon$ iff $s = \epsilon$.

We will consider two languages for a given PN N (corresponding to the P-type and L-type languages in [7]):

$$L(N) = \{\sigma(s) \in \Sigma^* | s \in T^* \wedge \delta_N(\mu_0, s) \text{ is defined}\} \tag{2.3}$$

$$L_m(N) = \{\sigma(s) \in \Sigma^* | s \in T^* \wedge \delta_N(\mu_0, s) \in F\}. \tag{2.4}$$

Next, define the classes of languages

$$\mathcal{L} := \{K \subset \Sigma^* | \exists N \in \mathcal{P}(K = L(N))\} \tag{2.5}$$

$$\mathcal{L}_m := \{K \subset \Sigma^* | \exists N \in \mathcal{P}(K = L_m(N))\} \tag{2.6}$$

and similarly the classes $\mathcal{L}^{NS}$, $\mathcal{L}_m^{NS}$, $\mathcal{L}^R$, $\mathcal{L}_m^R$ by substituting $\mathcal{P}$ by $\mathcal{P}^{NS}$ and $\mathcal{P}^R$, respectively, in the above. Clearly, $\mathcal{L}^R \subseteq \mathcal{L}^{NS} \subseteq \mathcal{L}$ and $\mathcal{L}_m^R \subseteq \mathcal{L}_m^{NS} \subseteq \mathcal{L}_m$. If $\epsilon \in \Sigma$, i.e., if transitions can be labeled with the empty string, then Peterson's construction [7, pp. 126–128] (for reachability analysis) actually proves the reverse inclusions. We collect this and other results from [7] in the following proposition. (Recall that a language is ***regular*** iff it is generated by an FSM.)

Proposition 2.1 [7]:

i) $\mathcal{L} \subset \mathcal{L}_m$.

ii) $\mathcal{R} \subset \mathcal{L}_m$ and $\bar{\mathcal{R}} \subset \mathcal{L}$, where $\mathcal{R}$ denotes the class of regular languages and $\bar{\mathcal{R}}$ that of closed regular languages.

iii) If the problem formulation is changed to allow $\epsilon \in \Sigma$, then $\mathcal{L}^R = \mathcal{L}$ and $\mathcal{L}_m^R = \mathcal{L}_m$. □

Remark 2.1:

i) When σ is required to be injective (the so-called "free-labeling" case), the properties of $\mathcal{L}$ and $\mathcal{L}_m$ are not as interesting as when this constraint is relaxed. Indeed, ii) in the above theorem is no longer true. This is why we do not require this assumption.

ii) It follows from definitions (2.3) and (2.5) that $\forall L \in \mathcal{L}$, $\epsilon \in L$. For the corresponding languages in $\mathcal{L}_m$, it simply means that $\mu_0 \in F$. It is still the case that $\epsilon \notin \Sigma$. □

Manuscript received February 17, 1989; revised July 14, 1989. This work was supported in part by the National Science Foundation under Grant ECS-8707671.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122.

IEEE Log Number 8933388.

[1] The terminology is from [2].

0018-9286/90/0400-0482$01.00

III. A New Proof for Petri Net Languages

We now complement the results of [7] and [2] by presenting a new constructive proof of $\mathcal{L}_m^R = \mathcal{L}_m$ and $\mathcal{L}^R = \mathcal{L}$ for the case of ϵ-free languages (i.e., $\epsilon \notin \Sigma$). To make the constructive procedure easier to follow, we first prove that $\mathcal{L}_m^{NS} = \mathcal{L}_m^R$ in Section III-A, and then we show how to extend that result to $\mathcal{L}_m = \mathcal{L}_m^R$ in Section III-B (similarly for $\mathcal{L} = \mathcal{L}^R$).

A. Self-Loop Free Case

Let $N = ((P, T, A, I, O), \Sigma, \sigma, \mu_0, F) \in \mathcal{P}^{NS}$. The objective is to build $N' = ((P', T', A', I', O'), \Sigma, \sigma', \mu_0', F') \in \mathcal{P}^R$ such that $L_m(N') = L_m(N)$ and $L(N') = L(N)$. This will prove that $\mathcal{L}_m^{NS} = \mathcal{L}_m^R$ and $\mathcal{L}^{NS} = \mathcal{L}^R$. The construction procedure whose steps are described below, replaces each place $p_j \in P$ by a set of places $P_j' \subseteq P'$, and each transition $t_i \in T$ by a set of transitions $T_i' \subseteq T'$. (Recall that $I(t_i) \cap O(t_i) = \varnothing \; \forall t_i \in T$.)

Step 1: For each $p_j \in P$, define

$$|P_j'| := \max\left\{\max_{t_i \in T}\left|O(t_i, p_j)\right|, \max_{t_i \in T}\left|I(t_i, p_j)\right|\right\}. \tag{3.1}$$

Index each of the $|P_j'|$ places with k, $1 \leq k \leq |P_j'|$, and form the disjoint union $P' := \cup_{p_j \in P} P_j'$. Thus, for each place p_j in N, the number of corresponding places p_{jk}' in N' is equal to the maximum number of incoming/outgoing multiple arcs to/from p_j. □

Before going to Step 2, we define the set $F(X, n)$, where X is a set whose elements are indexed by positive integers, $1 \leq i \leq |X|$, and where $n \in \mathbb{N}$. $F(X, n) \subseteq X^n$ and it is constructed as follows. i) Build n-tuples from the indexed elements of X, with unity step, with $|X| + 1 \equiv 1$ ("wrap-around"), and with x_1 the first element of the first n-tuple, x_2 the first element of the second n-tuple, etc. ii) The last n-tuple to build is the one whose first component is $x_{|X|}$. Thus, the n-tuples look like $(x_1, \cdots, x_n), (x_2, \cdots, x_{n+1}), \cdots, (x_{|X|}, \cdots, x_{|X|+n})$. By convention, $F(X, 0) = \varnothing$. When two tuples of $F(X, n)$ contain the same elements, we say that they are equivalent. When $F(X, n)$ contains equivalent tuples, we delete all but one of these tuples. This arises if $|X| = n$. In all cases, each element in X appears in the same number of n-tuples in $F(X, n)$.

Example 3.1:

$$F(\{x_1, x_2, x_3\}, 3) = \{(x_1, x_2, x_3)\}$$

$$F(\{x_1, x_2, x_3, x_4, x_5\}, 3) = \{(x_1, x_2, x_3), (x_2, x_3, x_4), (x_3, x_4, x_5), (x_4, x_5, x_1), (x_5, x_1, x_2)\}. \quad \square$$

Step 2: For each $t_i \in T$, define[2]

$$T_i' := [\mathbf{X}_{p_j \in I(t_i)} F(P_j', \left|I(t_i, p_j)\right|)]\mathbf{X}[\mathbf{X}_{p_j \in O(t_i)} F(P_j', \left|O(t_i, p_j)\right|)]. \tag{3.2}$$

Let

$$\sum_{p_j \in I(t_i)} \left|I(t_i, p_j)\right| = q_i \text{ and } \sum_{p_j \in O(t_i)} \left|O(t_i, p_j)\right| = r_i.$$

Then $T_i' \subset (P')^{q_i + r_i}$. Again, one should label each of these transitions distinctly and then form the disjoint union $\cup_{t_i \in T} T_i' =: T'$. The labeling function σ' is consistent with the original σ: if $t_{ij}' \in T_i'$, then $\sigma'(t_{ij}') = \sigma(t_i)$. (If $t_{ij}' \in T_i'$, we say that t_{ij}' and t_i are *corresponding* transitions.) □

Step 3: The remainder of the structure C' of N', namely, the connections between places and transitions, is done by examining the tuples in each T_i' of (3.2). Let $t_{ij}' \in T_i'$. $I'(t_{ij}')$ corresponds to the first q_i components of t_{ij}' since, from (3.2), these components are places in subsets P_j' where $p_j \in I(t_i)$; for each p_j, rather than using multiple arcs, $|I(t_i, p_j)|$ different places of P_j' are connected to t_{ij}'. Similarly, $O'(t_{ij}')$ corresponds to the remaining r_i components of t_{ij}' since these components are places in subsets P_j' where this time $p_j \in O(t_i)$. It results from this construction that the original connections of C are preserved in C' for all corresponding transitions, but avoiding this time multiple arcs due to the multiplicity of places and transitions. □

For a given marking μ of N, define the set of markings of N'

$$M'(\mu) := \left\{\mu' \in N^{|P'|} \,\middle|\, \sum_{p_{ik}' \in P_i'} \mu'(p_{ik}') = \mu(p_i) \, \forall p_i \in P\right\}. \tag{3.3}$$

If $\mu' \in M'(\mu)$, then we write $\mu = M'^{-1}(\mu')$.

Step 4: We take $M'(\mu_0)$ to be the set of admissible initial markings of N'. In a similar manner, we define

$$F' := \{\mu' \in N^{|P'|} \mid \exists \mu \in F \text{ s.t.} (\mu' \in M'(\mu))\}. \tag{3.4}$$

□

Example 3.2: Fig. 1 illustrates the application of Steps 1-3 to a PN with multiple arcs. For example, $|T_1'| = |F(P_1', 1)| \cdot |F(P_2', 2)| \cdot |F(P_3', 1)| = 1 \cdot 3 \cdot 2 = 6$. The first transition in T_1', say t_{11}', corresponds to the tuple $((p_{11}'), (p_{21}', p_{22}', p_{31}'))$, where we have separated input and output places. The rest of the structure of N' is straightforward to derive. Observe that $|T_2'| = 1$ since $|P_2'| = |I(t_2, p_2)|$ and $|P_3'| = |O(t_2, p_3)|$. □

Lemma 3.1: Let μ be the current marking of N and let any $\mu' \in M'(\mu)$ be that of N'. If $t_{ij}' \in T_i'$ is enabled in N', then $t_i \in T$, the corresponding transition in N, is enabled in N. (We shall say: (N, μ) is consistent with respect to (N', μ'), or simply (N, μ) is (N', μ')-consistent.)

Proof: Follows from the construction procedure. □

We can also *define* the notion that (N', μ') is (N, μ)-consistent if, whenever $\mu = M'^{-1}(\mu')$, all the enabled t_i at μ in N possess a corresponding enabled t_{ij}' at μ' in N'. Observe, however, that (N', μ') is not (N, μ)-consistent for all $\mu' \in M'(\mu)$. For example, in Fig. 1, take $\mu(p_2) = 3$ and $\mu'(p_{21}') = 3$ (0 tokens everywhere else). Then T_2' is not enabled in N' even though t_2 is in N.

Let us say that P_j' is filled uniformly if the tokens it contains can be mapped to a substring of $(p_{j1}' \cdots p_{j|P_j'|}')^*$. If all the P_j' in a marking μ' are filled uniformly, then we say that μ' is a ***uniform marking***. In other words, a set of places P_j' is filled uniformly if P_j' is filled in the order of the indexing of the places p_{jk}' it contains (with "wrap-around"), with some arbitrary starting place p_{js}' and where the last filled place is $p_{j(n-1)}'$. Let the next-to-be-filled place be p_{jn}'. We define the "next-to-be-filled tuple" by T_i' in P_j' to be the tuple in $F(P_j', |O(t_i, p_j)|)$ that is equivalent (i.e., equal up to reordering) to the tuple $(p_{jn}', \cdots, p_{j(n+\Delta)}')$ where $\Delta = |O(t_i, p_j)| - 1$. Let the above p_{js}' be the oldest-filled place in P_j'. Similarly, we define the "oldest-filled tuple" in P_j' for T_i' to be the tuple in $F(P_j', |I(t_i, p_j)|)$ that is equivalent to the tuple $(p_{js}', \cdots, p_{j(s+\delta)}')$ where $\delta = |I(t_i, p_j)| - 1$. (Example 3.3 below illustrates these two concepts.) From the definition of $F(\cdot, \cdot)$, the next-to-be-filled and oldest-filled tuples are well defined.

The above definition of uniform marking together with Step 3 of the construction procedure leads to the following lemma.

Lemma 3.2: If marking μ' in N' is uniform, then (N', μ') is $(N, M'^{-1}(\mu'))$-consistent. □

Theorem 3.1:

i) $\forall \mu_0' \in M'(\mu_0)$, $L(N') \subseteq L(N)$ and $L_m(N') \subseteq L_m(N)$.

ii) $\exists \mu_0' \in M'(\mu_0)$ such that $L(N') = L(N)$ and $L_m(N') = L_m(N)$.

Proof: i) By Lemma 3.1, (N, μ_0) is (N', μ_0')-consistent $\forall \mu_0' \in M'(\mu_0)$. We now argue that if (N, μ) is (N', μ')-consistent and if one fires t_{ij}' in N' and t_i in N, resulting in new markings μ_{new}' and μ_{new}, then $\mu_{\text{new}}' \in M'(\mu_{\text{new}})$ and (N, μ_{new}) is (N', μ_{new})-consistent. This is because all the transitions in a set T_i' in N' respect the original connections of t_i (recall Step 3 above). Thus, the firing of any $t_{ij}' \in T_i'$ in N' consumes and generates the *same* number of tokens as the firing of t_i in N; moreover, these tokens are removed from or deposited into *corresponding places* (P_i' in N' corresponds to p_i in N). Consequently, at each step of the firing sequence $t_{s_1 j_1}' \cdots t_{s_n j_n}'$ in N', t_{s_k} can be fired in N because it is enabled, and thus $t_{s_1} \cdots t_{s_n}$ is a firing sequence of N. From the definition of σ', it follows that $L(N') \subseteq L(N)$, which in turn implies, from the definition of F', that $L_m(N') \subseteq L_m(N)$.

[2] $\mathbf{X}$ denotes Cartesian product.

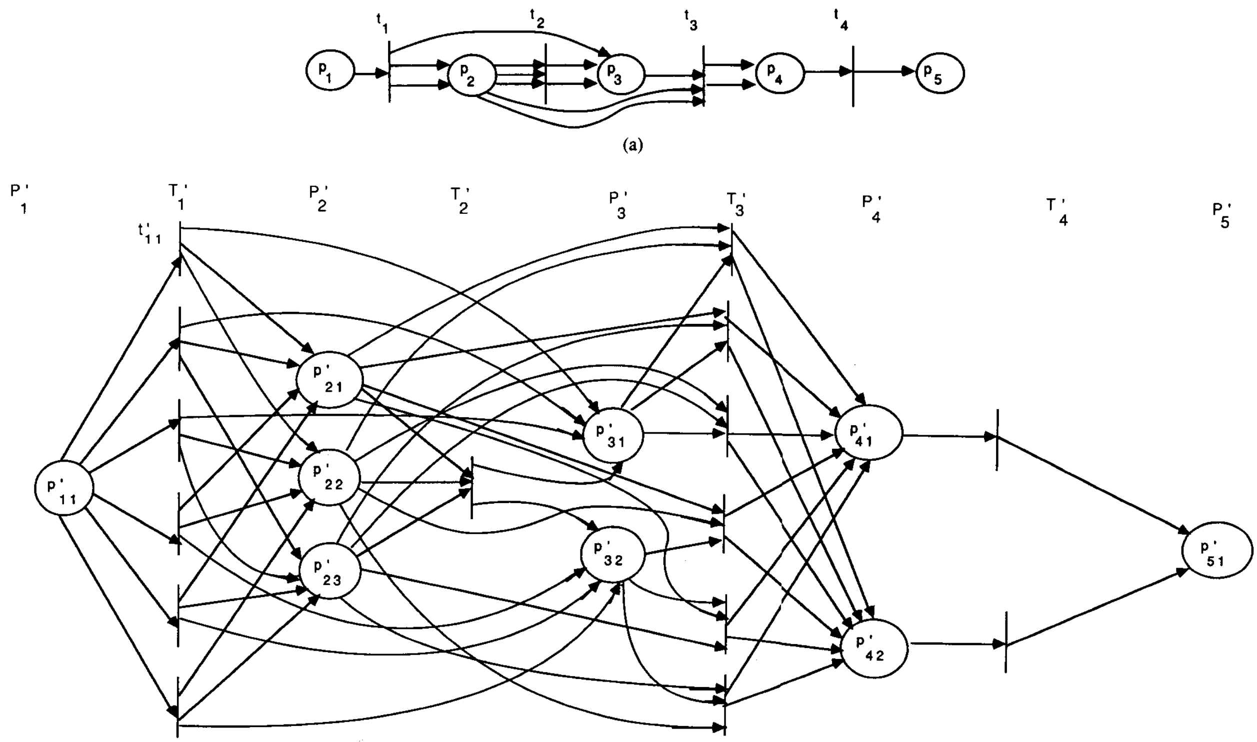

Fig. 1. (a) General Petri net N. (b) Restricted Petri net N'.

ii) We now show that every string in $L(N)$ can be generated by N' by proper choice of μ_0', provided that given a firing of t_i in N, one fires the "proper" $t_{ik}' \in T_i'$ in N'.

Choose the μ_0' that fills each P_j' uniformly with starting place p_{j1}' for all p_j such that $\mu_0(p_j) > 0$. Thus, this μ_0' is uniform and as a consequence of Lemma 3.2, (N', μ_0') is (N, μ_0)-consistent. Next, assume that (N', μ') is (N, μ)-consistent and that μ' is uniform. We argue that if one fires t_i in N, with $\mu_{\text{new}} = \delta_N(\mu, t_i)$, then $\exists t_{ik}' \in T_i'$ such that $\mu_{\text{new}}' = \delta_{N'}'(\mu', t_{ik}')$ is uniform and (N', μ_{new}') is (N, μ_{new})-consistent. This inductive argument will prove part ii) of the theorem. Its correctness is established as follows. The proper t_{ik}' to fire in N' is the transition that, when fired, removes the tokens from the "oldest-filled" tuples for T_i' in $F(P_j', |I(t_i, p_j)|)$, $p_j \in I(t_i)$, and that fills the "next-to-be-filled" tuples by T_i' in $F(P_l', |O(t_i, p_l)|)$, $p_l \in O(t_i)$. The existence of t_{ik}' is guaranteed from the construction of T_i' in Step 2; namely, from (3.2), there exists a transition in T_i' for each possible combination of consecutive input and output places from the appropriate P_j' and P_l' subsets. By definition of "oldest-filled" and uniform marking, and since $\mu' \in M'(\mu)$, t_{ik}' is necessarily enabled in N' whenever t_i is enabled in N. Firing t_{ik}' results in marking μ_{new}' which is uniform because the rule oldest-filled/next-to-be-filled clearly preserves the uniformity of the markings when transitions are fired in N'. Moreover, by Lemma 3.2, (N', μ_{new}') will be (N, μ_{new})-consistent.

In conclusion, any firing sequence $t_{s_1}, \cdots, t_{s_n}$ in N possesses a corresponding firing sequence $t_{s_1k_1}' \cdots t_{s_nk_n}'$ in N', which completes the proof. □

Example 3.3: Consider Fig. 1(b). If $\mu'(P_2') = [0\ 1\ 1]$ then the oldest-filled tuple in $F(P_2', |I(t_2, p_2)|)$ for T_2' is $(1, 2, 3)$ [meaning $(p_{21}', p_{22}', p_{23}')$] [because $(1, 2, 3)$ is equivalent to $(2, 3, 1)$], the oldest-filled tuple for T_3' in $F(P_2', |I(t_3, p_2)|)$ in $(2, 3)$, while the next-to-be-filled tuple by T_1' in $F(P_2', |O(t_1, p_2)|)$ is $(1, 2)$. □

B. General Case

Let $S(t_i) = I(t_i) \cap O(t_i)$. Then define sets of arcs $I^s(\cdot, \cdot)$, $O^s(\cdot, \cdot)$, $I^{ns}(\cdot, \cdot)$, $O^{ns}(\cdot, \cdot)$, $I^{ns}(\cdot)$, and $O^{ns}(\cdot)$ (all initialized to $\varnothing$) as follows.

- If $p_j \in S(t_i)$, then $I^s(t_i, p_j) = I(t_i, p_j)$ and $O^s(t_i, p_j) = O(t_i, p_j)$.
- If $p_j \notin S(t_i)$, then $I^{ns}(t_i, p_j) = I(t_i, p_i)$ and $O^{ns}(t_i, p_j) = O(t_i, p_j)$.
- $I^{ns}(t_i) = I(t_i) - S(t_i)$ and $O^{ns}(t_i) = O(t_i) - S(t_i)$.

The construction procedure of Section III-A is modified as follows.

Step 1:

$$|P_j'| = \max \left[\max_{t_i \in T} \left| I^{ns}(t_i, p_j) \right|, \max_{t_i \in T} \left| O^{ns}(t_i, p_j) \right|, \max_{t_i \in T} \left(\left| I^s(t_i, p_j) \right| + \left| O^s(t_i, p_j) \right| \right) \right]. \tag{3.5}$$

□

Step 2:

$$T_i' = [\mathbf{X}_{p_j \in I^{ns}(t_i)} F(P_j', \left| I^{ns}(t_i, p_j) \right|)]\mathbf{X} \cdot [\mathbf{X}_{p_j \in O^{ns}(t_i)} F(P_j', \left| O^{ns}(t_i, p_j) \right|)]\mathbf{X} \cdot \left[\mathbf{X}_{p_j \in S(t_i)} \left[\bigcup_{f_{ij}(k) \in F(P_j', |O^s(t_i, p_j)|)} [\{f_{ij}(k)\} \mathbf{X} F(P_j'(i, k), \left| I^s(t_i, p_j) \right|)] \right] \right] \tag{3.6}$$

where

$$P_j'(i, k) := \{p_{jl}' \in P_j' | p_{jl}' \text{ is not a component of the tuple } f_{ij}(k)\}. \tag{3.7}$$

Let

$$\sum_{p_j \in I^{ns}(t_i)} \left| I^{ns}(t_i, p_j) \right| = q_i \quad \text{and} \quad \sum_{p_j \in O^{ns}(t_i)} \left| O^{ns}(t_i, p_j) \right| = r_i. \ \square$$

Step 3: The connections between places and transitions in C' are again inferred from the tuples in T_i', except that this time these tuples have a more complicated form. The first q_i components of $t_{ij}' \in T_i'$ are input places to t_{ij}' since they correspond to $I^{ns}(t_i)$ in (3.6). The next r_i components are output places from t_{ij}' since they correspond to $O^{ns}(t_i)$. Then output and input places alternate according to the third term in (3.6). For each $p_j \in S(t_i)$, we have $|O^s(t_i, p_j)|$ components that are output places from t_{ij}' followed by $|I^s(t_i, p_j)|$ components that are input places to t_{ij}'.

Step 4: Unchanged. □

Remark 3.1: The intuition behind the third term in (3.6) is the following. Given $|P_j'|$ when $p_j \in S(t_i)$, each $f_{ij}(k)$ in (3.6) represents a different way of filling $|O^s(t_i, p_j)|$ places in P_j', which corresponds to the firing of t_i. For each $f_{ij}(k)$, we consider all the possible ways the remaining places (i.e., the set $P_j'(i, k)$) can be combined to provide the necessary $|I^s(t_i, p_j)|$ tokens for the firing of t_i. (Note that from (3.5), there are enough remaining places.) Thus, self-loops are eliminated while the original connections in C are preserved. □

The above construction preserves the two important properties of the structure C' of Section III-A: i) preservation of the connections of t_i for all $t_{ij}' \in T_i'$; ii) presence of enough t_{ij}' to represent all the possible combinations of consecutive input and output places from the appropriate P_j' subsets. Consequently, the proof of Theorem 3.1 applies *mutatis mutandis* to the present case.

In conclusion, we have proved the following result.

Theorem 3.2: $\mathcal{L}_m^R = \mathcal{L}_m$ and $\mathcal{L}^R = \mathcal{L}$. □

IV. Discussion on Petri Nets and Finitely Recursive Processes

In [5], Inan and Varaiya introduce the FRP formalism for the modeling of DES. The language complexity of FRP is in terms of the sets of traces that these processes can generate. Let A be the fixed set of events for the class of FRP, which we denote $\mathfrak{F}$. We can write

$$\bar{\mathcal{L}}^{\mathfrak{F}} := \{K \subset A^* \mid \exists X \in \mathfrak{F}(K = \operatorname{tr} X)\} \tag{4.1}$$

to represent the class of *closed* languages that $\mathfrak{F}$ can generate ($\operatorname{tr} X$ denotes the set of traces of process X). $\bar{\mathcal{L}}^{\mathfrak{F}}$ is compared to $\mathcal{L}$ in Fact 3.7 in [5]. However, two implicit assumptions on the structure of the PN generating $\mathcal{L}$ are made in the proof of that fact: i) there are no multiple arcs in the PN structure; and ii) the transition labeling function σ is injective (the free-labeling case).

In view of Theorem 3.2, Assumption i) is not restrictive. However, as was mentioned in Remark 2.1, Assumption ii) does make a difference. As one cannot relax that assumption in the proof of Fact 3.7 in [5] without affecting the synchronous composition defining the "master process" there, we suggest the following approach. We use the symbol change function of Hoare to define the new class of processes

$$\mathfrak{F}^{sc} := \{Y \mid \exists X \in \mathfrak{F} \wedge f_{sc}: A \to \Sigma(Y = f_{sc}(X))\} \tag{4.2}$$

where f_{sc} is as defined in [3, p. 55], i.e.,

$$\operatorname{tr} Y = \{s \in \Sigma^* \mid \exists t \in \operatorname{tr} X(s = f_{sc}(t))\} \tag{4.3}$$

where $f_{sc}(\sigma_1 \cdots \sigma_n) := f_{sc}(\sigma_1) \cdots f_{sc}(\sigma_n)$ for $\sigma_i \in A$, $1 \le i \le n$. Observe that the function f_{sc} is not necessarily injective and that $\epsilon \notin \Sigma$. For our purposes, it is sufficient to consider FRP with constant event functions; also, the particular way in which the termination function of $Y = f_{sc}(X)$ is defined is irrelevant here.

Given a PN $N = (C, \Sigma, \sigma, \mu_0, F)$, we proceed as follows to build an FRP Z such that $\operatorname{tr} Z = L(N)$. We follow the proof of Fact 3.7 where $A = T$, the set of transitions of N, in order to build the "master process" $Z' = Y_1 \| \cdots \| Y_{|P|} \in \mathfrak{F}$ (cf. [5]). Then we build $Z = f_{sc}(Z')$ where $f_{sc} \equiv \sigma$, the transition labeling function of N. Thus, $\operatorname{tr} Z = L(N)$, which proves the following result.

Proposition 4.1: $\mathcal{L} \subseteq \bar{\mathcal{L}}^{\mathfrak{F}^{sc}}$. □

V. Conclusion

$\mathcal{L}_m$ and $\mathcal{L}$ are important classes of PNL. Because they allow noninjective transition labeling functions, they are more general and possess more interesting properties than "free-labeling" languages (cf. Remark 2.1). Moreover, they avoid ϵ-labeling of transitions. Unlike the situation in finite automata theory, where both deterministic and nondeterministic automata generate the same class of languages (namely, $\mathfrak{R}$), allowing ϵ-labeling may affect the properties of classes of PNL. In this context, Theorem 3.2 is an interesting result since it shows that the language complexity equivalence of GPN and RPN is also true for ϵ-free languages. Theorem 3.2 also helped to establish Proposition 4.1. In general, the simplified structure of RPN as compared to GPN may prove helpful in the study of PN.

Acknowledgment

It is a pleasure to acknowledge many discussions with B. Krogh on the subject of Petri net languages.

References

[1] S. Drees *et al.* "Bibliography of Petri nets," in *Advances in Petri Nets 1987*, G. Groos and J. Hartmanis, Eds. New York: Springer-Verlag, 1987, pp. 309–341.
[2] M. Hack, "Petri net languages," Tech. Rep. MIT/LCS/TM-159, Lab. Comput. Sci., M.I.T., Mar. 1976.
[3] C. A. R. Hoare, *Communicating Sequential Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[4] L. E. Holloway and B. H. Krogh, "Efficient synthesis of control logic for a class of discrete event systems," Carnegie-Mellon Univ., Pittsburgh, PA; also in *IEEE Trans. Automat. Contr.*, to be published.
[5] K. Inan and P. Varaiya, "Finitely recursive process models for discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 626–639, July 1988.
[6] B. H. Krogh, "Controlled Petri nets and maximally permissive feedback logic," in *Proc. 25th Allerton Conf.*, Sept. 1987.
[7] J. L. Peterson, *Petri Net Theory and the Modelling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[8] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81–98, Jan. 1989.

Simplifying H_∞ Controller Synthesis Via Classical Feedback System Structure

HARRY S. HVOSTOV

***Abstract*—Recently, H_∞-optimal controller synthesis problem has been shown to lead to the solution of two algebraic Riccati equations. This note explores the properties of this solution in the framework of the classical cascade controller feedback structure. In addition to allowing effective tradeoff between significant closed-loop transfer functions, this system structure is shown to substantially simplify controller computation in the practically important case of open-loop stable plant and weights in that at most one Riccati equation needs to be solved.**

I. Introduction

The original solution to the H_∞ norm minimization problem obtained by solving an equivalent model matching problem is summarized in [5] along with the relevant factorization theory. Complete state-space solutions for all stabilizing controllers which simultaneously minimize the H_∞ norm of the appropriate closed-loop transfer function are given in [1]. There it is shown that under relatively mild assumptions on the plant the controller can be computed via a set of explicit state-space formulas which depend upon the solution of two associated algebraic Riccati equations.

In a typical H_∞-optimal controller synthesis problem the plant would incorporate additional frequency dependent weights which are selected

Manuscript received February 24, 1989; revised May 12, 1989.
The author is with Storage Systems Advanced Development Group, Digital Equipment Corporation, Colorado Springs, CO 80919.
IEEE Log Number 8933389.

0018-9286/90/0400-0485$01.00