Deterministic λ-free Petri net Languages and their Application to the Supervisory Control of Discrete Event Dynamic Systems

Ramavarapu S. Sreenivas*
Department of General Engineering & CSL
University of Illinois at Urbana-Champaign
Urbana, IL 61801
E-mail: sree@deds.ge.uiuc.edu

August 14, 1993

Abstract

In this paper we introduce a family of languages called *Deterministic λ-free Petri net languages* (DPNLs) [5]. We show that the *controllability* of a DPNL $K \subseteq \Sigma^*$ with respect to a DPNL $L \subseteq \Sigma^*$ is decidable. That is, it is possible to decide if (i) $K \subseteq L$, and (ii) $K\Sigma_u \cap L \subseteq K$, where $\Sigma = \Sigma_u \cup \Sigma_c$ and $\Sigma_u \cap \Sigma_c = \emptyset$. We also show that this family of languages strictly includes the family of *Free-labeled Petri net languages* (FLPNLs), another family of languages where the controllability of one language with respect to another is decidable [9]. Essentially this paper is an extension of reference [9].

1 Introduction

We assume familiarity with supervisory control of the untimed behaviors of *Discrete Event Dynamic Systems* (DEDS). For a detailed treatment the reader is referred to the original papers by Ramadge and Wonham [8, 7]. In particular, we consider the *forbidden string problem* [8], where the objective is to synthesize a supervisor that prevents the occurrence of certain strings in the closed-loop behavior.

We concern ourselves with infinite-state systems and modeling formalisms that can represent such systems. Apart from the purely speculative motivation, the practical stimulus for a study of such systems is that often large, finite-state systems can be approximated by compact infinite-state systems. For any representation of an infinite-state system to be meaningful it is imperative that the representation has a finite description. This requirement brings with it an issue that is unique to finite representations of infinite-state systems: *undecidability* or *uncomputability*. Loosely, when a particular property is uncomputable for a specific modeling formalism it implies that there is no *single* mechanized procedure that establishes the absence or presence of this property for an *arbitrary* instance.

A frequently encountered phenomenon in discrete state models is that an increase in modeling power is always accompanied by a decrease in decision power. A desirable feature of any modeling formalism is that it is restrictive enough to decide all the properties of interest while at the same time general enough to represent a large class of systems. We assume that deciding the controllability (cf. section 5, [8]) of a language in a specific modeling formalism with respect to another language in the same formalism is essential for establishing viability. Recently it was shown that languages represented as *free-labeled Petri nets* (FLPNs) are restrictive enough to decide the property of controllability while at the same time general enough to represent a non-trivial class of infinite-state systems [9]. In this note we introduce a family of languages called *deterministic λ-free Petri net languages* (DPNLs) and we show that this family is restrictive enough to decide the property of controllability. Since the family of DPNLs strictly include the family of languages represented by FLPNs, in a sense this paper can be considered as an extension of reference [9].

In the next section we introduce the class of *deterministic λ-free Petri net languages* (DPNLs) and show that the controllability of one DPNL K with respect to another DPNL L is decidable. We conclude with discussion on future research topics in section 3.

*This research was supported in part by grant RES-BRD-IC-SREENIVAS-1-2-68317 from the UIUC Campus Research Board

CH 3381-1/93/$01.00 ©1993 IEEE

2 Deterministic λ-free Petri net Languages

In this section we assume familiarity with *Petri nets* (PNs) and Petri net languages (PNLs) and only those terms that are crucial to the the material in this section are defined. For a detailed treatment of this subject the reader is referred to Peterson's book [6] or Hack's technical report [1].

A PN $N = (\Pi, T, \Phi, \mathbf{m}^0)$ is an ordered 4-tuple, where $\Pi = \{p_1, p_2, \ldots, p_n\}$ is a set of n *places*, $T = \{t_1, t_2, \ldots, t_m\}$ is a collection of m *transitions*, $\Phi \subseteq (\Pi \times T) \cup (T \times \Pi)$ is a set of *arcs*[1] , $\mathbf{m}^0{:}\Pi \rightarrow \mathcal{N}$ is the *initial marking function* (or the *initial marking*), and $\mathcal{N}$ is the set of non-negative integers. The state of a PN is given by the *marking* $\mathbf{m}{:}\Pi \rightarrow \mathcal{N}$ which indicates the distribution of *tokens* in each place. We note that since the value of the marking is unbounded, finite PNs can represent infinite-state systems. For a given marking $\mathbf{m}$ a transition $t \in T$ is said to be *enabled* if

$$\forall\, p \in {}^\bullet t,\ \mathbf{m}(p) \geq 1, \text{ where } {}^\bullet x := \{y \mid (y, x) \in \Phi\}.$$

For a given marking $\mathbf{m}$ the set of enabled transitions is denoted by the symbol $T_e(\mathbf{m})$. An enabled transition $t \in T_e(\mathbf{m})$ can *fire*, which changes the marking $\mathbf{m}$ to $\hat{\mathbf{m}}$ according to the equation

$$\hat{\mathbf{m}}(p) = \mathbf{m}(p) - \text{card}(p^\bullet \cap \{t\}) + \text{card}({}^\bullet p \cap \{t\}),$$

where $x^\bullet := \{y \mid (x, y) \in \Phi\}$ and the symbol card($\bullet$) is used to denote the cardinality of the set argument. In this note we do not consider simultaneous firing of multiple transitions.

A string of transitions $t_1 t_2 \cdots t_k$, where $t_i \in T$ ($i \in 1, 2, \ldots, k$) is said to be a *valid firing sequence* starting from the marking $\mathbf{m}$, if,

- the transition t_1 is enabled under the marking $\mathbf{m}$, and
- for $i \in \{1, 2, \ldots, k\text{-}1\}$ the firing of the transition t_i produces a marking under which the transition t_{i+1} is enabled.

Given an initial marking $\mathbf{m}^0$ the set of *reachable markings* for $\mathbf{m}^0$ denoted by $\Re(N, \mathbf{m}^0)$, is defined as the set of markings generated by all valid firing sequences starting with marking $\mathbf{m}^0$ in the PN N. Given any arbitrary $\mathbf{m} \in \mathcal{N}^n$, it is of interest to know if $\mathbf{m} \in \Re(N, \mathbf{m}^0)$. This is known as the *reachability problem* (cf. chapter 5, [6]), which has been shown to be decidable [4, 2]. This will play a crucial role in all the results in this section.

A *labeled PN* is a 3-tuple $P = (N, \Sigma, \alpha)$, where $N = (\Pi, T, \Phi, \mathbf{m}^0)$ is a PN, Σ is a finite symbol set, and $\alpha{:}T \rightarrow \Sigma \cup \{\lambda\}$ is a labeling function that identifies a particular element of the symbol set Σ or the *null symbol* λ with each transition. If the labeling function $\alpha{:}T \rightarrow \Sigma$, is such that the null symbol λ is not allocated to any transition, then the labeled PN is said to be a *λ-free labeled PN*. If the labeling function $\alpha{:}T \rightarrow \Sigma$ is a bijection then the labeled PN is said to be a *free-labeled PN*.

A valid firing sequence $t_1 t_2 \cdots t_k$ of the PN N defines a string ω where $\omega = \alpha(t_1)\alpha(t_2)\cdots\alpha(t_k) \in \Sigma^*$. Note that in general, the length of ω, denoted by $|\,\omega\,|$ satisfies the inequality $|\,\omega\,| \leq k$. For λ-free, or free-labeled PNs we have $|\,\omega\,| = k$. It is easy to see that a labeled PN $P = (N, \Sigma, \alpha)$ effectively defines a language $L(P) \subseteq \Sigma^*$, where

$$L(P) = \{\omega \in \Sigma^* \mid \omega = \alpha(t_1)\alpha(t_2)\cdots\alpha(t_k) \text{ for some valid firing sequence } t_1 t_2 \cdots t_k \text{ of the PN } N \text{ starting from } \mathbf{m}^0\}.$$

A language $K \subseteq \Sigma^*$ is a *PN language* (PNL) if there exists a labeled PN $P = (N, \Sigma, \alpha)$ such that $L(P) = K$.

A λ-free labeled PN P is said to be a *deterministic λ-free PN* (DPN) if $\forall\, \mathbf{m} \in \Re(N, \mathbf{m}^0)$, $\forall\, t_1, t_2 \in T_e(\mathbf{m}^0)$, $\alpha(t_1) = \alpha(t_2) \Leftrightarrow t_1 = t_2$. That is, a deterministic λ-free PN $P = (N, \Sigma, \alpha)$ is a λ-free labeled PN such that for any reachable marking $\mathbf{m} \in \Re(\text{N}, \mathbf{m}^0)$, no two transitions in the set $T_e(\mathbf{m})$ are assigned to the same symbol by the labeling function α. The family of *deterministic λ-free PN languages* (DPNLs) is the family of languages generated by DPNs. Likewise, the family of *free-labeled PN languages* (FLPNLs) is defined as the family of languages generated by FLPNs.

It can be shown that the family of DPNLs strictly includes the family of *regular languages* (cf. chapter 2, [3]). This observation follows from the fact that every regular language has a deterministic finite-state automaton that generates it (cf. theorems 2.5.1 and 2.3.1, [3]), and the fact that any finite-state automaton can be converted into an equivalent labeled *state machine PN* (cf. [6]) with a labeling function such that no two transitions that share an input place have the same associated symbol. Also, it can be shown that the family of FLPNLs are contained in the family of DPNLs. This containment follows by definition. At the end of this section we show by example that this containment is proper. It is known that containment is decidable for the family of DPNLs (cf. section 6, [5]).

[1] In this note we restrict our attention to *Ordinary PNs*. This is implicitly assumed when we suppose $\Phi \subseteq (\Pi \times T) \cup (T \times \Pi)$. However, all results in this paper can be extended to *General PNs* (cf. section 5.3, [6]).

Theorem 2.1 *[5] Containment (and hence also the equivalence) of DPNLs is reducible to the reachability problem.*

It is worthwhile to ponder over one of the implications of theorem 2.1. In formal language theory it has always been of interest to investigate the equivalence of deterministic and non-deterministic formalisms, where the notions of determinism and non-determinism have their usual intuitive interpretation of the state-transition relation being single-valued or multiple-valued. In the context of finite-state automata it is well-known that the family of languages accepted/generated by deterministic finite-state automata is equivalent to the family of languages accepted/generated by non-deterministic finite-state automata (cf. theorem 2.3.1, chapter 2, [3]). For context-free languages it is known that the family of languages accepted/generated by deterministic context-free grammars (or deterministic push-down automata) is strictly contained in the family of languages accepted/generated by non-deterministic context-free grammars (or non-deterministic push-down automata) (cf. theorem 3.6.1, chapter 3, [3]). For PNLs Hack has shown that containment is undecidable (cf. theorem 8.2, [1]), while theorem 2.1 states that containment is decidable for DPNLs as the reachability problem is decidable. Thus implying the strict containment of the family of DPNLs in the family of PNLs. That is, there are some PNLs that do not have a deterministic generator/acceptor.

The literature on the supervisory control of DEDS assume that the plant and supervisor representations are deterministic. For certain paradigms that finitely represent infinite-state systems the assumption on the existence of a deterministic model cannot be made with impunity. The implications of these observations to supervisory control is an interesting future research issue.

We return to the main result of this paper, namely, the controllability of a DPNL K with respect to another DPNL L is decidable. Essentially the proof of theorem 3.4 in reference [9] , where the controllability of a FLPNL K with respect to another FLPNL L is shown to be decidable, is a proof of the current result, *mutatis mutandis*.

Theorem 2.2 *Given two DPNLs K, $L \subseteq \Sigma^*$, and any arbitrary partition of the symbol set $\Sigma = \Sigma_u \cup \Sigma_c$, the controllability of K with respect to L is decidable. That is, we can decide if*

1. $K \subseteq L$, *and*
2. $K\Sigma_u \cap L \subseteq K$.

Let $\Sigma = \{a, b\}$ and let $K = \{ a^n b^m \mid n \geq m \geq 0\}$ be a prefix-closed language on Σ. It is straightforward to show that K is not regular. Hence it has no finite-state generator/recognizer. We show that K is not a FLPNL. We establish this result for ordinary PNs (cf. section 5.3, [6]) by contradiction. The proof for general labeled PNs[2] is more involved but is a straightforward extension of the following argument. Let us assume that we have a FLPN that generates K . Any FLPN that generates K should only have two transitions as $\text{card}(\Sigma) = 2$. We use the symbols associated with these transitions to unambiguously identify the transitions. That is, we have two transitions "a" and "b". We first note that initially a can fire an arbitrary number of times, thus implying the input place set of a is contained in the output place set. For ordinary PNs this implies that the token load of each element in the input place set of a is constant. Also observe that the first firing of b (after an arbitrary, non-zero firings of a) should disable the firing of a, a little thought should convince the reader that this implies the two transitions must share a common input place whose token load is zero after the first firing of b. But this would mean that after the first firing of b both a and b are disabled. But to recognize K it should be possible to fire b at least as many times as a had fired in the past, and since a can fire an arbitrary number of times initially, it follows that there cannot be a FLPN that generates K. Figure 1 shows a DPN that generates K. It is easy to see that the transitions t_1 and t_3 are never simultaneously enabled. Hence K is a DPNL.

To complete the example we consider the supervisory control problem in reference [10]. Let the *plant language* $L = \{a^*b^*\}$, where $\Sigma = \{a, b\}$ and $\Sigma_c = \{b\}$. The desired closed-loop behavior is K, where K is as defined above. As shown in figure 1 K is a DPNL, figure 2 shows that L is also a DPNL. In principle, there exists a *single* mechanized procedure that can decide the controllability of *any* DPNL K with respect to *any* DPNL L. For the example under consideration it is easy to show that K is controllable with respect to L (cf. [10]). Figure 3 shows a DPN based solution to the problem. Figure 3 suggests that the controllable symbol b be permitted in the plant only when the token load of place p_3 is non-zero.

3 Conclusions

In this paper we assume that deciding the controllability (cf. section 5, [8]) of a language in a specific modeling formalism with respect to another language within the same formalism is essential for establishing

[2]Labeled PNs with weighted arcs.

viability. We introduce a family of languages called *deterministic λ-free Petri net languages* (DPNLs) [5] and show that the controllability of one DPNL K with respect to another DPNL L is decidable. This is a direct consequence of the decidability of the *Petri net reachability problem* [4, 2]. We establish the non-triviality of the family of DPNLs by example. The note also contains some observations on the properties of DPNLs and their implications to the issue of supervisory control of DEDS.

The literature on the supervisory control of DEDS assume that the plant and supervisor representations are deterministic. For certain paradigms that finitely represent infinite-state systems the assumption on the existence of a deterministic model cannot be made with impunity. This is the case with PN languages (cf. theorem 2.2). The implications of this observation to supervisory control is an interesting future research issue.

References

[1] M.H.T. Hack. Petri net languages. Technical Report 159, Massachusetts Institute of Technology, March 1976.

[2] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc. of the 14th Annual Symposium of the Theory of Computing*, pages 267–281, 1982.

[3] H.R. Lewis and C.H. Papadimtriou. *Elements of the theory of computation.* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[4] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, pages 441–460, 1984.

[5] E. Pelz. Closure properties of Deterministic Petri nets. In *STACS 87*, pages 371–382, 1987. Volume 247 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

[6] J.L. Peterson. *Petri net theory and the modeling of systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[7] P.J. Ramadge and W.M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 25(5):1202–1218, September 1987.

[8] P.J. Ramadge and W.M. Wonham. Supervisory control of class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.

[9] R.S. Sreenivas. A note on deciding the controllability of a language K with respect to a language L. *IEEE Transactions on Automatic Control*, 38(4):658–662, April 1993.

[10] R.S. Sreenivas and B.H. Krogh. On petri net models for infinite state supervisors. *IEEE Trans. on Automatic Control*, 37(2):274–277, Feb. 1992.

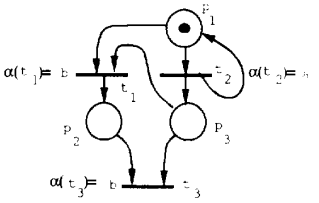

Figure 1: A DPN that generates the language K.

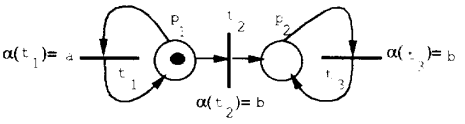

Figure 2: A DPN that generates the language L.

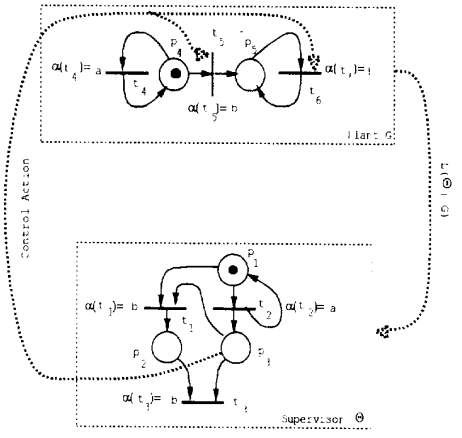

Figure 3: A DPN-based solution to the supervisory control problem of section 2.