International Journal of Control
Vol. 78, No. 12, 15 August 2005, 949–967

Taylor & Francis
Taylor & Francis Group

Signed real measure of regular languages for discrete event supervisory control

A. RAY*

Department of Mechanical Engineering, The Pennsylvania State University, University Park, PA 16802, USA

(*Received 9 October 2004; in final form 27 May 2005*)

This paper reviews, expands, and clarifies the underlying concepts of a signed real measure of regular languages, which has been used as a novel tool for synthesis of discrete event supervisory control systems. The language measure is constructed upon the principles of automata theory and real analysis. It allows total ordering of a set of partially ordered sublanguages of a regular language for quantitative evaluation of the supervised behaviour of deterministic finite state automata (DFSA) under different supervisors. In the setting of the language measure, a supervisor's performance is superior if the supervised plant is more likely to terminate at a good marked state and/or less likely to terminate at a bad marked state. The computational complexity of the language measure algorithm is polynomial in the number of DFSA states.

1. Introduction

Discrete event systems belong to a special class of dynamical systems. The states of a discrete event system may take discrete (or symbolic) values and change only at (possibly asynchronous) discrete instants of time, in contrast to the familiar continuously varying dynamical systems of the physical world, which can be modelled by differential or difference equations. The dynamics of many human-engineered systems evolve asynchronously in time via complex interactions of various discrete-valued events with continuously varying physical processes. The relatively young discipline of discrete event systems has undergone rapid growth over the last three decades with the evolution of human engineered complex systems, such as integrated control and communication systems, distributed sensing and monitoring of large-scale engineering systems, manufacturing and production systems, software fault management, and military Command, Control, Computer, Communication, Intelligence, Surveillance, and Reconnaissance (C^4ISR) systems.

The discipline of discrete event systems was initiated with simulation of human-engineered processes about four decades ago in the middle of nineteen sixties. The art of discrete event simulation emerged with the development of a simulation software package, called GPSS, that was followed by numerous other software simulation tools, such as SIMSCRIPT II.5, SLAM II, and SIMAN (Law and Kelton 1991). Shortly thereafter, computer scientists and control theorists entered the field and brought in theoretical concepts of languages and automata in modelling discrete event systems. In the late nineteen sixties, Arbib (see Kalman *et al*. 1969) showed how algebraic methods could be used to explore the structure of finite automata to model dynamical systems. Around that time, computer scientists focused on formal languages, automata theory, and computational complexity for application of language-theoretic concepts (e.g., regular expressions and context-free grammars) in software development including design of compilers and text processors (Yu 1997, Hopcroft *et al*. 2001). In the late nineteen seventies and early nineteen eighties, Ho and co-workers introduced the concept of finite perturbation in discrete event systems for modelling and analysis of human-engineered systems (Ho and Cao 1991). So far, no concrete theoretical concept and mathematical tools had been available for analysis and synthesis of discrete event control systems.

The concept of discrete event supervisory (DES) control was first introduced in the seminal paper of

*Email: axr2@psu.edu

International Journal of Control
ISSN 0020–7179 print/ISSN 1366–5820 online © 2005 Taylor & Francis Group Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/00207170500202447

Ramadge and Wonham (1987) and this important paradigm has been subsequently extended by other researchers (for example, see citations in Kumar and Garg (1995) and Cassandras and Lafortune (1999), and the October 2000 issue of Part B of IEEE Transactions on Systems, Man, and Cybernetics). These efforts have led to the evolution of a new discipline in decision and Control, called Supervisory Control Theory (SCT), that requires partitioning the discrete-event behaviour of a physical process, called the plant, into legal and illegal categories. The legal behaviour of plant dynamics is modelled by a deterministic finite-state automaton, abbreviated as DFSA in the sequel. The DFSA model is equivalent to a regular language that is built upon an alphabet of finitely many events; the event alphabet is partitioned into subsets of controllable events (that can be disabled) and uncontrollable events (that cannot be disabled). Based on the regular language of an unsupervised plant, SCT synthesizes a DES controller as another regular language, having the common alphabet with the plant language, that guarantees restricted legal behaviour of the supervised plant based on the desired specifications. Instead of continuously handling numerical data, DES controllers are designed to process event strings to disable certain controllable events in the physical plant. A number of algorithms for DES control synthesis have evolved based on the automata theory and formal languages relying on the disciplines of Computer Science and Control Science. In general, a supervised plant DFSA is synthesized as a parallel composition of the unsupervised plant DFSA and a supervisor DFSA (Cassandras and Lafortune 1999). The supervised plant DFSA yields a sublanguage of the unsupervised plant language, which enables restricted legal behaviour of the supervised plant (Ramadge and Wonham 1987, Kumar and Garg 1995, Cassandras and Lafortune 1999). These concepts have been extended to several practical applications, including hierarchical Command, Control, Communication, and Intelligence (C^3I) systems (Phoha *et al.* 2002). Apparently, there have been no quantitative methods for evaluating the performance of supervisory controllers and establishing thresholds for their performance.

The concept of permissiveness has been used in DES control literature (Kumar and Garg 1995, Cassandras and Lafortune 1999) to facilitate qualitative comparison of DES controllers under the language controllability condition. Design of maximally permissive DES controllers has been proposed by several researchers based on different assumptions. However, maximal permissiveness does not imply best performance of the supervised plant from the perspective of achieving plant operational objectives. For example, in the travelling salesman problem, a maximally permissive supervisor may not yield the least expensive way of visiting the scheduled cities and returning to the starting point because no quantitative measure of performance is addressed in this type of supervisor design.

The above argument evinces the need for a signed real measure of regular languages, which can be used for quantitative evaluation and comparison of different supervisors for a physical plant, instead of relying on permissiveness as the (qualitative) performance index. Construction of the proposed language measure follows Myhill–Nerode Theorem (Martin 1997, Hopcroft *et al.* 2001), which states that a regular language can be partitioned into finitely many right-invariant equivalence classes. In other words, a state-based partitioning of the (unsupervised) plant language yields equivalence classes of finite-length event strings. Each marked state is characterized by a signed real value that is chosen based on the designer's perception of the state's impact on the system performance. Conceptually similar to conditional probability, each event is assigned a cost based on the state at which it is generated. This procedure permits a string of events, terminating on a good (bad) marked state, to have a positive (negative) measure. A supervisor can be designed in this setting such that the supervisor attempts to eliminate as many bad strings as possible and retain as many good strings as possible. Different supervisors may achieve this goal in different ways and generate a partially ordered set of supervised sublanguages. The language measure then creates a total ordering on the performance of the supervised sublanguages, which provides a precise quantitative comparison of the controlled plant behaviour under different supervisors. This feature is formally stated as follows.

Given that the relation $\subseteq$ induces a partial ordering on a set of supervised sublanguages $\{L(S^j/G), j = 1, \ldots, N\}$ of the plant language $L(G)$ under supervisors whose languages are $\{L(S^j), j = 1, \ldots, N\}$, the language measure μ induces a total ordering $\leq$ on $\{\mu(L(S^j/G))\}$. In other words, the range of the set function μ is totally ordered while its domain could be partially ordered.

The above problem was first addressed by Wang and Ray (2004) who proposed a signed measure of regular languages; an alternative approach was proposed by Ray and Phoha (2003) who constructed a vector space of formal languages and defined a metric based on the total variation measure of the language.

This paper reviews, clarifies and expands the contents of previous publications (Ray and Phoha 2003, Wang and Ray 2004) from the perspectives of discrete-event supervisory control within a unified framework and also introduces new concepts and ramifications of the language measure and its parameter identification. Systematic procedures for computation of the language

measure are developed in this paper and they are illustrated with an engineering example. The major objective here is rigorous formulation and systematic construction of a real signed measure of regular languages, based on the fundamental principles of automata theory and real analysis. The quantitative tools are readily applicable to analysis and synthesis of discrete-event supervisory control algorithms. Specifically, performance indices of supervisors can be defined in terms of the language measure.

The signed real measure for a DFSA, presented in this paper, is constructed based on assignment of an event cost matrix and a characteristic vector. Two techniques for language measure computation have been recently reported. While the first technique (Wang and Ray 2004) leads to a system of linear equations whose (closed form) solution yields the language measure vector, the second technique (Ray and Phoha 2003) is a recursive procedure with finite iterations. A sufficient condition for finiteness of the signed measure has been established in both cases; and an upper bound is established for the max norm of the language measure vector.

In order to induce total ordering on the measure of different sublanguages of a plant language under different supervisors, it is implicit that same strings in different sublanguages must be assigned the same measure. This is accomplished by a quantitative tool that requires a systematic procedure to assign a characteristic vector and an event cost matrix. The clarifications and extensions presented in this paper are intended to enhance development of a systematic analytical tool for synthesizing discrete-event supervisory control. For example, Ray *et al.* (2004) have proposed unconstrained optimal control of regular languages where a state-based optimal control policy is obtained by selectively disabling controllable events to maximize the measure of the supervised plant language.

The paper is organized in eight sections including the present introductory section and two appendices. Section 2 briefly describes the language measure and introduces the notations. Section 3 presents the procedure by which the performance of different supervisors can be compared based on a common quantitative tool. It also discusses two methods for computing language measure. Section 4 addresses issues regarding physical interpretation of the event cost used in the language measure. Section 5 presents a recursive algorithm for identification of the language parameters (i.e., elements of the event cost matrix). Section 6 illustrates the usage of the language measure for construction of metric spaces of formal languages and synthesis of optimal discrete-event supervisors. Section 7 presents an application of the language measure on the discrete-event model of a twin-engine unmanned aircraft (Ray and Phoha 2003, Ray *et al.* 2004). The paper is summarized and concluded in §8 along with recommendations for future research. Appendix I provides pertinent mathematical background of measure theory as needed in the main body of the paper. Appendix II establishes a sufficient condition for absolute convergence of the language measure.

2. Language measure concept

This section first introduces the signed real measure of regular languages, originally reported in (Ray and Phoha 2003, Wang and Ray 2004). Then, the underlying concepts of language measure are clarified in the context of discrete event supervisory (DES) control.

Let $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$ be a trim (i.e., accessible and co-accessible) finite-state automaton model (Ramadge and Wonham 1987, Cassandras and Lafortune 1999) that represents the discrete-event dynamics of a physical plant, where $Q = \{q_k : k \in \mathcal{I}_Q\}$ is the set of states and $\mathcal{I}_Q \equiv \{1, 2, \ldots, n\}$ is the index set of states; the automaton starts with the initial state q_i; the alphabet of events is $\Sigma = \{\sigma_k : k \in \mathcal{I}_\Sigma\}$, and $\mathcal{I}_\Sigma \equiv \{1, 2, \ldots, \ell\}$ is the index set of events; $\delta : Q \times \Sigma \to Q$ is the (possibly partial) function of state transitions; and $Q_m \equiv \{q_{m_1}, q_{m_2}, \ldots, q_{m_r}\} \subseteq Q$ is the set of marked (i.e., accepted) states with $q_{m_k} = q_j$ for some $j \in \mathcal{I}_Q$.

Let Σ^* be the Kleene closure of Σ, i.e., the set of all finite-length strings made of the events belonging to Σ as well as the empty string ϵ that is viewed as the identity of the monoid Σ^* under the operation of string concatenation, i.e., $\epsilon s = s = s\epsilon$. The extension $\hat{\delta} : Q \times \Sigma^* \to Q$ is defined recursively in the usual sense (Martin 1997, Hopcroft *et al.* 2001). For DES control (Ramadge and Wonham 1987), the event alphabet Σ is partitioned into sets, Σ_c and $\Sigma - \Sigma_c$ of controllable and uncontrollable events, respectively, where each event in Σ_c and no event in $\Sigma - \Sigma_c$ can be disabled by the supervisor.

Definition 1: The language $L(G_i)$ generated by a DFSA G_i initialized at the state $q_i \in Q$ is defined as

$$L(G_i) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) \in Q\}. \tag{1}$$

Since the state transition function δ is allowed to be a partial function, $L(G_i) \subseteq \Sigma^*$ following Definition 1; if δ is a total function, then the generated language $L(G_i) = \Sigma^*$.

Definition 2: Given a DFSA plant model G_i, having the set of controllable events $\Sigma_c \subseteq \Sigma$, let S and $\widetilde{S}$ be two controllable supervisors (i.e., each of S and $\widetilde{S}$ is represented by an event disabling mapping $L(G_i) \to 2^{\Sigma_c}$). Let the languages of the plant supervised by S and $\widetilde{S}$

be denoted as $L(S/G_i)$ and $L(\widetilde{S}/G_i)$, respectively. Then, S is said to be less permissive (or more restrictive) than $\widetilde{S}$, denoted as $S \preceq \widetilde{S}$, if the following condition holds.

$$S \preceq \widetilde{S} \quad \text{if } L(S/G_i) \subseteq L(\widetilde{S}/G_i). \tag{2}$$

In other words, S may disable a larger set of controllable events than $\widetilde{S}$ following the execution of an event string $s \in \Sigma^*$.

Definition 3: The language $L_m(G_i)$ marked by a DFSA G_i, initialized at the state $q_i \in Q$, is defined as

$$L_m(G_i) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) \in Q_m\}. \tag{3}$$

Definition 4: For every $q_i, q_k \in Q$, let $L_{i,k}$ denote the set of all strings that, starting from the state q_i, terminate at the state q_k, i.e.,

$$L_{i,k} = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) = q_k\}. \tag{4}$$

In order to obtain a quantitative measure of the marked language, the set Q_m of marked states is partitioned into Q_m^+ and Q_m^-, i.e., $Q_m = Q_m^+ \cup Q_m^-$ and $Q_m^+ \cap Q_m^- = \emptyset$. The positive set Q_m^+ contains all good marked states that one would desire to reach, and the negative set Q_m^- contains all bad marked states that one would not want to terminate on, although it may not always be possible to completely avoid the bad states while attempting to reach the good states. From this perspective, each marked state is characterized by an assigned real value that is chosen based on the designer's perception of the state's impact on the system performance.

Definition 5: The characteristic function $\chi: Q \to [-1, 1]$ assigns a signed real weight to a state-based sublanguage $L_{i,j}$, having each of its strings terminating on the same state q_j, and is defined as

$$\forall q_j \in Q, \quad \chi(q_j) \in \begin{cases} [-1, 0), & q_j \in Q_m^- \\ \{0\}, & q_j \notin Q_m \\ (0, 1], & q_j \in Q_m^+. \end{cases} \tag{5}$$

The state weighting vector, denoted by $\mathbf{X} = [\chi_1 \ \chi_2 \ \dots \ \chi_n]^T$, is called the **X**-vector, where $\chi_j \equiv \chi(q_j)$. That is, the jth element χ_j of **X**-vector is the weight assigned to the corresponding state q_j.

In general, the marked language $L_m(G_i)$ consists of both good and bad strings, which start from the initial state q_i, respectively lead to Q_m^+ and Q_m^-. Denoting the set difference operation by "–", any event string belonging to the language $L^0(G_i) \equiv L(G_i) - L_m(G_i)$ leads to one of the non-marked states belonging to $Q - Q_m$ and $L^0(G_i)$ does not contain any one of the good or bad strings. Partitioning Q_m into the positive set Q_m^+ and the negative set Q_m^- leads to partitioning of the marked language $L_m(G_i)$ into a positive language $L_m^+(G_i)$ and a negative language $L_m^-(G_i)$. Based on the equivalence classes defined in the Myhill–Nerode Theorem (Hopcroft *et al.* 2001), the regular languages $L(G_i)$ and $L_m(G_i)$ can be expressed as

$$L(G_i) = \bigcup_{k \in \mathcal{I}_Q} L_{i,k} \tag{6}$$

$$L_m(G_i) = L_m^+(G_i) \bigcup L_m^-(G_i) \tag{7}$$

where the sublanguage $L_{i,k} \subseteq L(G_i)$ is uniquely labelled by the state q_k, $k \in \mathcal{I}_Q$ and $L_{i,k} \cap L_{i,j} = \emptyset \ \ \forall k \neq j$; and $L_m^+(G_i) \equiv \cup_{q_k \in Q_m^+} L_{i,k}$ and $L_m^-(G_i) \equiv \cup_{q_k \in Q_m^-} L_{i,k}$ are good and bad sublanguages of $L_m(G_i)$, respectively. Then, the null sublanguage $L^0(G_i) = \cup_{q_k \notin Q_m} L_{i,k}$ and $L(G_i) = L^0(G_i) \cup L_m^+(G_i) \cup L_m^-(G_i)$.

Now a signed real measure is constructed as μ^i: $2^{L(G_i)} \to \mathbf{R} \equiv (-\infty, \infty)$ on the σ-algebra $M = 2^{L(G_i)}$. (Appendix I provides details of measure-theoretic definitions and results.) With this choice of σ-algebra, every singleton set made of an event string $s \in L(G_i)$ is a measurable set, which allows its quantitative evaluation based on the above state-based decomposition of $L(G_i)$ into null (i.e., $L^0(G_i)$), positive (i.e., $L_m^+(G_i)$), and negative (i.e., $L_m^-(G_i)$) sublanguages.

Conceptually similar to the conditional probability, each event is assigned a cost based on the state at which it is generated.

Definition 6: The event cost of the DFSA G_i is defined as a (possibly partial) function $\tilde{\pi}: \Sigma^* \times Q \to [0, 1]$ such that $\forall q_i \in Q$, $\forall \sigma_j \in \Sigma$, $\forall s \in \Sigma^*$,

$$\begin{aligned} &\tilde{\pi}[\sigma_j, q_i] = 0 \text{ if } \delta(q_i, \sigma_j) \text{ is undefined; } \tilde{\pi}[\epsilon, q_i] = 1; \\ &\tilde{\pi}[\sigma_j, q_i] \equiv \tilde{\pi}_{ij} \in [0, 1); \sum_{j \in \mathcal{I}_\Sigma} \tilde{\pi}_{ij} < 1; \\ &\tilde{\pi}[\sigma_j s, q_i] = \tilde{\pi}[\sigma_j, q_i]\, \tilde{\pi}[s, \delta(q_i, \sigma_j)]. \end{aligned} \tag{8}$$

A simple application of the induction principle to the last part of Definition 6 shows $\tilde{\pi}[st, q_j] = \tilde{\pi}[s, q_j]\, \tilde{\pi}[t, \hat{\delta}(q_j, s)]$. The condition $\sum_{k \in \mathcal{I}_Q} \tilde{\pi}_{jk} < 1$ provides a sufficient condition for the existence of the real signed measure as discussed in § 3 and Appendix II. Additional comments on the physical interpretation of the event cost are provided in § 4.

The $n \times \ell$ event cost matrix is defined as

$$\widetilde{\mathbf{\Pi}} = \begin{bmatrix} \tilde{\pi}_{11} & \tilde{\pi}_{12} & \dots & \tilde{\pi}_{1\ell} \\ \tilde{\pi}_{21} & \tilde{\pi}_{22} & \dots & \tilde{\pi}_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\pi}_{n1} & \tilde{\pi}_{n2} & \dots & \tilde{\pi}_{n\ell} \end{bmatrix}. \tag{9}$$

Definition 7: The state transition cost, $\pi: Q \times Q \to [0, 1)$, of the DFSA G_i is defined as follows.

$$\forall i, j \in \mathcal{I}_Q,$$

$$\pi_{ij} = \begin{cases} \sum_{\sigma \in \Sigma} \tilde{\pi}[\sigma, q_i], & \text{if } \delta(q_i, \sigma) = q_j \\ 0 & \text{if } \{\delta(q_i, \sigma) = q_j\} = \emptyset. \end{cases} \tag{10}$$

The $n \times n$ state transition cost matrix is defined as

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} & \dots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \dots & \pi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n1} & \pi_{n2} & \dots & \pi_{nn} \end{bmatrix} \tag{11}$$

and is referred to as the $\mathbf{\Pi}$-matrix in the sequel.

Definition 8: Given a DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$ the cost ν^i of a sublanguage $K \subseteq L(G_i)$ is defined as the sum of the event cost $\tilde{\pi}$ of individual strings belonging to K.

$$\nu^i(K) = \sum_{s \in K} \tilde{\pi}[s, q_i]. \tag{12}$$

Definition 9: For a given DFSA G_i, the signed real measure of every singleton string set $\{s\} \in L_{i,j} \subseteq L(G_i)$ is defined as $\mu^i(\{s\}) \equiv \tilde{\pi}(s, q_i)\chi_j$ implying that

$$\forall s \in L_{i,j}, \quad \mu^i(\{s\}) \begin{cases} = 0, & q_j \notin Q_m \\ > 0, & q_j \in Q_m^+ \\ < 0, & q_j \in Q_m^- \end{cases} \tag{13}$$

Thus an event string terminating on a good (bad) marked state has a positive (negative) measure and one terminating on a non-marked state has zero measure. It follows from Definition 9 that the signed measure of the sublanguage $L_{i,j} \subseteq L(G_i)$ of all events, starting at q_i and terminating at q_j, is

$$\mu^i(L_{i,j}) = \left(\sum_{s \in L_{i,j}} \tilde{\pi}[s, q_i] \right) \chi_j \tag{14}$$

Definition 10: The signed real measure of the language of a DFSA G_i initialized at a state $q_i \in Q$, is defined as

$$\mu_i \equiv \mu^i(L(G_i)) = \sum_{j \in \mathcal{I}_Q} \mu^i(L_{i,j}). \tag{15}$$

The language measure vector, denoted as $\boldsymbol{\mu} = [\mu_1 \ \mu_2 \dots \mu_n]^T$, is called the $\boldsymbol{\mu}$-vector.

Remark 1: $\mu^i(L_m(G_i)) = \mu_i \ \ \forall i \in \mathcal{I}_Q$ because $\chi_k = 0$ $\forall q_k \in Q - Q_m$.

It follows from Definition 10 that $\mu^i(L_{i,j}) = \nu^i(L_{i,j})\chi_j$. Under the condition of $\sum_k \tilde{\pi}_{jk} < 1$ in Definition 6, convergence of the signed real language measure μ^i has been proved in (Ray and Phoha 2003, Wang and Ray 2004). The total variation measure $|\mu^i|$ of μ^i has also been shown to be finite for every $i \in \mathcal{I}_Q$ (Ray and Phoha 2003).

In the above setting, the role of the language measure in DES control synthesis is explained below.

A discrete-event non-marking supervisor S restricts the marked behaviour of an unsupervised (i.e., uncontrolled) plant G_i such that $L_m(S/G_i) \subseteq L_m(G_i)$. The unsupervised marked language $L_m(G_i)$ consists of good strings leading to Q_m^+ and bad strings leading to Q_m^-. A supervised (i.e., controlled) language $L_m(S/G_i)$ based on a given specification of the supervisor S may disable some of the bad strings and keep some of the good strings enabled. Different supervisors $S_j: j \in \{1, 2, \dots, n_s\}$ for a DFSA G_i achieve this goal in different ways and generate a partially ordered set of supervised sublanguages $\{L_m(S_j/G_i): j \in \{1, 2, \dots, n_s\}\}$. The real signed measure μ^i provides a precise quantitative comparison of the controlled plant behaviour under different supervisors because the set $\{\mu^i(L_m(S_j/G_i)): j \in \{1, 2, \dots, n_s\}\}$ is totally ordered.

In order to realize the above goal, the performance of different supervisors has to be evaluated based on a common quantitative tool. Let $G \equiv \langle Q^G, \Sigma, \delta^G, q_1^G, Q_m^G \rangle$ denote the unsupervised plant and $S \equiv \langle Q^S, \Sigma, \delta^S, q_1^S, Q_m^S \rangle$ denote the supervisor with respective languages $L(G)$ and $L(S)$ and the corresponding marked languages $L_m(G)$ and $L_m(S)$.

Let $\mathbb{G} \equiv \langle Q, \Sigma, \delta, q_1, Q_m \rangle$ where $Q = Q^G \times Q^S$, $q_1 = (q_1^G, q_1^S)$, $Q_m = \{(p, \tilde{p}) | p \in Q_m^G \text{ and } \tilde{p} \in Q_m^S\}$ and the transition function δ is defined by the formula: $\forall p \in Q^G$, $\tilde{p} \in Q^S$, and $\sigma \in \Sigma$

$$\delta((p, \tilde{p}), \sigma) = \left(\delta^G(p, \sigma), \delta^S(\tilde{p}, \sigma) \right). \tag{16}$$

Then, the marked language $L_m(\mathbb{G})$ of the automaton $\mathbb{G}$ is $L_m(G) \cap L_m(S)$ because $\mathbb{G}$ is a parallel composition (Ramadge and Wonham 1987, Cassandras and Lafortune 1999) of the automata G and S that have the common alphabet Σ. Then, it follows that the extension $\hat{\delta}$ satisfies the condition

$$\forall s \in \Sigma^*, \ \hat{\delta}((p, \tilde{p}), s) = (\hat{\delta}^G(p, s), \hat{\delta}^S(\tilde{p}, s)) \tag{17}$$

whenever $\hat{\delta}^G(p, s)$ and $\hat{\delta}^S(\tilde{p}, s)$ are defined.

The unsupervised plant language $L(G)$ is partitioned by $L_{1,j}^G$, $1 \le j \le n^G$ where $|Q^G| = n^G$. Similarly, the supervisor language $L(S)$ is partitioned by $L_{1,k}^S$, $1 \le k \le n^S$ where $|Q^S| = n^S$. With this construction, each of the sublanguages $L_{1,j}^G$ is further partitioned by $L_{1,j}^G \cap L_{1,k}^S$. Thus, for any $q_j^G \in Q_m^G$, the set of strings, which is retained in $L_m(G) \cap L_m(S)$, is given by $L_{1,j}^G \cap (\cup_{q_k^S \in Q_m^S} L_{1,k}^S)$. In this setting, the goal is to synthesize a supervisor that will retain many strings that terminate on some state in Q_m^{G+} while discarding many strings

that terminate on Q_m^{G-}. It will yield a relatively high measure and hence good performance.

The above construction shows how the event cost and characteristic function assigned to the unsupervised plant can be used as a quantitative tool with which the performance of different supervisors can be evaluated and compared. The following procedure indicates how this can be accomplished explicitly.

Definition 11: Let G, S and $\mathbb{G}$ be defined as above. Let G represent the unsupervised plant and $\tilde{\pi}^G$ be the event cost function and χ^G be the characteristic function. Then, for the DFSA $\mathbb{G}$ which represents the language of the supervised plant, the event cost function $\tilde{\pi}$ is defined as

$$\tilde{\pi}\left[\sigma, \left(q_i^G, q_j^S\right)\right] = \tilde{\pi}^G\left[\sigma, q_i^G\right] \quad \forall \sigma \in \Sigma \text{ and } \forall i,j \text{ s.t. } 1 \le i \le n^G, 1 \le j \le n^S. \tag{18}$$

The χ-vector for the DFSA $\mathbb{G}$ is defined as

$$\chi\left(\left(q_i^G, q_j^S\right)\right) = \chi^G(q_i^G)\mathcal{J}\left(q_j^S\right) \tag{19}$$

where $\mathcal{J}(\cdot)$ is the indicator function defined as

$$\mathcal{J}(p) = \begin{cases} 1 & p \in Q_m^S \\ 0 & p \notin Q_m^S. \end{cases} \tag{20}$$

Let $s \in L_{((q_1^G, q_1^S), (q_j^G, q_k^S))}$, i.e., the set of all strings starting at the state $(q_1^G, q_1^S) \in Q \equiv Q^G \times Q^S$ and terminating at (q_j^G, q_k^S). If $q_j^G = \hat{\delta}^G(q_1, s)$, it follows from Definition 9 that $\mu(\{s\}) = \tilde{\pi}^G[s, q_1^G]\chi^G(q_j^G)$ for the unsupervised (i.e., uncontrolled) plant. Following equations (18) and (19), the measure of the supervised (i.e., controlled) plant becomes

$$\begin{aligned} \mu_1(\{s\}) &= \tilde{\pi}\left[s, \left(q_1^G, q_1^S\right)\right] \chi\left(\left(q_j^G, q_k^S\right)\right) \\ &= \tilde{\pi}^G\left[s, q_1^G\right] \chi^G\left(q_j^G\right) \mathcal{J}\left(q_k^S\right). \end{aligned} \tag{21}$$

In other words, if no event in the string s is disabled by the supervisor, then $\mu_1(\{s\})$ in the supervised plant automaton $\mathbb{G}$ remains the same as in the unsupervised plant automaton G; otherwise, $\mu_1(\{s\}) = 0$. Thus, Definition 11 guarantees that the same strings in different supervised sublanguages of the unsupervised plant language $L(G_i)$ are assigned the same measure. Hence, the performance of different supervisors can be compared with a common quantitative tool.

Finally to conclude this section, it should be noted that while the domain (i.e., $2^{L(G_i)}$) of the language measure μ^i is partially ordered, its range which is a subset of $\mathbf{R}$ becomes totally ordered. The set $L(G_i)$ with the σ-algebra, $2^{L(G_i)}$, forms a measurable space. In principle, any measure μ can be defined on this measurable space to form a measure space (i.e., the triple $\langle L(G_i), 2^{L(G_i)}, \mu^i \rangle$). The choice of the signed language measure, as given by Definitions 9 and 10, has been motivated by the fact that it bears a physical significance and hence is qualified to serve as a performance measure for DES controller synthesis. Moreover, defining the measure in this way also leads to simple computational procedures as discussed in the next section and further elaborated in § 4.

3. Language measure computation

Various methods of obtaining regular expressions for DFSAs are reported in Martin (1997) and Hopcroft *et al.* (2001). While computing the measure of a given DFSA, the same event may have different significance when emanating from different states. This requires assigning (possibly) different costs to the same event defined on different states. Therefore, it is necessary to obtain a regular expression which explicitly yields the state-based event sequences. In order to compute the language measure, it is convenient to transform the procedures of evaluating regular expression from symbolic equations to algebraic ones. The following two methods (Ray and Phoha 2003, Wang and Ray 2004) are presented, in detail, for language measure computation.

3.1. *Method I: closed form solution*

This section presents a closed-form method to compute the language measure via inversion of a square operator.

Definition 12: Let $L_i \equiv L(G_i)$, $i \in \mathcal{I}_Q$, denote the regular expression representing the language of a DFSA $G_i = \langle Q, \Sigma, \delta, q_i, Q_m \rangle$, where q_i is the initial state.

Definition 13: Let σ_j^k denote the set of event(s) $\sigma \in \Sigma$ that is defined on the state q_j and leads to the state $q_k \in Q$, where $j, k \in \mathcal{I}_Q$, i.e., $\delta(q_j, \sigma) = q_k$, $\forall \sigma \in \sigma_j^k \subseteq \Sigma$.

Then, given a DFSA $G_i = \langle Q, \Sigma, \delta, q_i, Q_m \rangle$ the procedure to obtain the system equation by a set of regular expressions L_i of the language $L(G_i)$, $i \in \mathcal{I}_Q$, is as follows:

$$\forall q_i \in Q, \quad L_i = \sum_{j \in \mathcal{I}_Q} R_{i,j} + \epsilon, \tag{22}$$

where the operator $\sum$ indicates the sum of regular expressions (equivalently, union of regular languages); and $R_{i,j}$ is defined as follows.

If there exists $\sigma \in \Sigma$ such that $\delta(q_i, \sigma) = q_j \in Q$ for $j \in \{1, \ldots, n\}$, then $R_{i,j} = \sigma_i^j L_j$, otherwise, $R_{i,j} = \emptyset$.

The set of symbolic equations may be written as

$$L_i = \sum_j \sigma_i^j L_j + \epsilon. \tag{23}$$

The above system of symbolic equations can be solved using a result given below, which is illustrated through an example.

Lemma 1: *Let u, v be two known regular expressions and r be an unknown regular expression that satisfies the following algebraic identity*:

$$r = ur + v. \tag{24}$$

Then, the following relations are true.

(1) $r = u^*v$ *is a solution to equation* (24).
(2) *If* $\epsilon \notin u$, *then* $r = u^*v$ *is the unique solution to equation* (24).

Proof: The proof of Lemma 1, which is also known as Arden's relation, is given in Yu (1997) and Ray and Phoha (2003). □

Example 1: In this example, shown in figure 1, the alphabet is $\Sigma = \{a, b\}$; the set of states is $Q = \{1, 2, 3\}$; the initial state is 1; and the only marked state is 2. Let the set of linear algebraic equations representing the transitions at each state of the DFSA be as follows:

$$\left.\begin{aligned} L_1 &= a_1^1 L_1 + b_1^2 L_2 + \epsilon \\ L_2 &= a_2^1 L_1 + b_2^3 L_3 + \epsilon \\ L_3 &= a_3^1 L_1 + b_3^2 L_2 + \epsilon \end{aligned}\right\} \tag{25}$$

where the 'forcing' term ϵ is introduced on the right side of each equation. For example, by application of Lemma 1, the regular expression for the language $L(G_1)$ is given as

$$L_1 = (a_1^1)^* b_1^2 (a_2^1 (a_1^1)^* b_1^2 + b_2^3 a_3^1 (a_1^1)^* b_1^2 + b_2^3 b_3^2)^* + \epsilon.$$

Instead of obtaining regular expressions, the language measure can be directly computed by transforming this set of equations into a system of linear equations based on the following result.

Theorem 1: *Following Definition* 10, *the language measure of the symbolic equation* (23) *is given by*

$$\mu_i = \sum_j \pi_{ij} \mu_j + \chi_i. \tag{26}$$

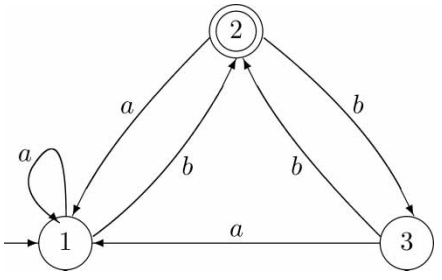

Figure 1. Finite state machine for Example 1.

Proof: Following equation (22) and Definition 5

$$\forall i \in \mathcal{I}_Q, \quad \mu^i(\epsilon) = \chi_i. \tag{27}$$

Therefore, each element of the vector $\mathbf{X} = [\chi_1 \ \chi_2 \ \cdots \ \chi_n]^T$ is the forcing function in equations (23) and (24). Starting from the state q_i, the measure of the language $L_i \equiv L(G_i)$ (see Definition 12)

$$\begin{aligned} \mu_i = \mu^i(L_i) &= \mu^i\left(\sum_j \sigma_i^j L_j + \epsilon\right) \\ &= \mu^i\left(\sum_j \sigma_i^j L_j\right) + \mu^i(\epsilon) \\ &= \sum_j \mu^i(\sigma_i^j L_j) + \nu^i(\varepsilon)\chi_i \\ &= \sum_j \pi(\sigma_i^j)\mu^j(L_j) + \chi_i \\ &= \sum_j \pi_{ij}\mu_j + \chi_i. \end{aligned}$$

The third equality in the above derivation follows from the fact that $\epsilon \cap \sigma_i^j L_j = \emptyset$. It is also true that

$$\forall j \neq k, \qquad \sigma_i^j L_j \bigcap \sigma_i^k L_k = \emptyset \tag{28}$$

since each string in $\sigma_i^j L_j$ starts with an event in σ_i^j while each string in $\sigma_i^k L_k$ starts from an event in σ_i^k and $\sigma_i^j \cap \sigma_i^k = \emptyset \ \forall j \neq k$ because G_i is a DFSA. This justifies the fourth equality. The fifth equality follows from Definition 8 and the fact that $\mu^i(L_{i,j}) = \nu^i(L_{i,j})\chi(q_j)$; therefore, by Definitions 7 and 13, $\mu^i(\sigma_i^j L_j) = \pi[q_i, q_j]\,\mu^j(L_j) = \pi_{ij}\mu_j$. □

In vector notation, equation (26) in Theorem 1 is expressed as

$$\boldsymbol{\mu} = \boldsymbol{\Pi}\boldsymbol{\mu} + \mathbf{X}$$

whose solution is given by

$$\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Pi})^{-1}\mathbf{X} \tag{29}$$

provided that the matrix $\mathbf{I} - \boldsymbol{\Pi}$ is invertible. The following important result guarantees the existence of $\boldsymbol{\mu}$.

Theorem 2: *Given DFSAs* $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$, *with the state transition cost matrix* $\boldsymbol{\Pi}$, *the matrix* $(\mathbf{I} - \boldsymbol{\Pi})$ *is an invertible bounded linear operator and* $\boldsymbol{\mu} \in \mathbf{R}^n$.

Proof: It follows from Definitions 6 and 7 that the induced max norm $\|\boldsymbol{\Pi}\|_\infty \equiv \max_i \sum_j \pi_{ij} = 1 - \theta$ where $\theta \in (0, 1)$. Then $(\mathbf{I} - \boldsymbol{\Pi})$ is invertible and is a bounded linear operator and $\|(\mathbf{I} - \boldsymbol{\Pi})^{-1}\|_\infty \leq \theta^{-1}$ (Naylor and

Sell 1982). Then, it follows from equation (29) that $\boldsymbol{\mu} \in \mathbf{R}^n$. □

Corollary 1 (to Theorem 2)**:** *The language measure vector* $\boldsymbol{\mu}$ *is bounded as* $\|\boldsymbol{\mu}\|_\infty \leq \theta^{-1}$ *where* $\theta \equiv (1 - \|\boldsymbol{\Pi}\|_\infty)$.

Proof: The proof follows by applying the norm inequality property and Theorem 2 to equation (29) and the fact that the max norm $\|\mathrm{X}\|_\infty \leq 1$ by Definition 5. □

Alternatively, sufficient conditions for convergence of μ can be obtained based on the properties of nonnegative matrices that are given in Appendix II. Therefore, Definitions 6 and 7 provide a sufficient condition for the language measure μ of the DFSA G_i to be finite. A closed-form algorithm to compute a language measure based on the above procedure is presented below.

Algorithm 1: Closed-form computation of the language measure

(1) For a given $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$, specify the characteristic vector $\mathbf{X}$ (see Definition 5) and determine the event cost matrix $\widetilde{\boldsymbol{\Pi}}$ (see Definition 6) via experimentation or simulation, as described later in § 5).
(2) Generate the $\boldsymbol{\Pi}$-matrix (Definition 7).
(3) Compute the language measure vector $\boldsymbol{\mu} \leftarrow (\mathbf{I} - \boldsymbol{\Pi})^{-1}\mathbf{X}$ using Gaussian elimination.
(4) Obtain μ_i, the ith element of $\boldsymbol{\mu}$-vector, which is the measure of the generated language of the DFSA G_i.

The jth element of the ith row of the $(\mathbf{I} - \boldsymbol{\Pi})^{-1}$ matrix, denoted as ν_i^j, is the language measure of the DFSA with the same state transition function δ as G_i and having the following properties: (i) the initial state is q_i; (ii) q_j is the only marked state; and (iii) the χ-value of q_j is equal to 1. Thus, $\mu_i \equiv \mu(L(G_i))$ is given by $\mu_i = \sum_j \nu_i^j \chi_j$. Numerical evaluation of the language measure of the automaton G_i requires Gaussian elimination of the single variable μ_i involving the real invertible matrix $(\mathbf{I} - \boldsymbol{\Pi})$. Therefore, the computational complexity of the language measure algorithm is polynomial in the number of states.

3.2. *Method II: recursive solution*

This section presents a second method to compute the language measure using a recursive procedure based on Kleene's theorem (Martin 1997) which states that the marked (i.e., accepted) language of a DFSA is regular. It also yields an algorithm to recursively construct the regular expression of its language instead of the closed form solution in Method I.

Definition 14: Given q_i, $q_k \in Q$, a non-empty string p of events (i.e., $p \neq \epsilon$) starting from q_i and terminating at q_k is called a path. A path p from q_i to q_k is said to pass through q_j if there exists $s \neq \epsilon$ and $t \neq \epsilon$ such that $p = st$; $\hat{\delta}(q_i, s) = q_j$ and $\hat{\delta}(q_j, t) = q_k$.

Definition 15: A path language p_{ik}^j is defined to be the set of all paths from q_i to q_k, which do not pass through any state q_r for $r > j$, and $\epsilon \notin p_{ik}^j$. The path language p_{ik} is defined to be the set of all paths from q_i to q_k. Thus, the language $L_{i,k}$ is obtained in terms of the path language p_{ik} as

$$L_{i,k} = \begin{cases} p_{ii} \bigcup \{\epsilon\}, & \text{if } k = i \\ p_{ik}, & \text{if } k \neq i \end{cases}$$
$$\Rightarrow \nu(L_{i,k}) = \begin{cases} \nu(p_{ii}) + 1, & \text{if } k = i \\ \nu(p_{ik}), & \text{if } k \neq i. \end{cases}$$

Every path language p_{ik}^j is a regular language and is a subset of $L(G_i)$. As shown in Ray and Phoha (2003), following recursive relation holds for $0 \leq j \leq n-1$, where $\|\mathrm{Q}\| = n$.

Theorem 3: Given a DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$, the following recursive relation holds for $1 \leq j \leq n-1$

$$\left.\begin{aligned} p_{lk}^0 &= \{\sigma \in \Sigma : \delta(q_l \sigma) = q_k\} \\ p_{lk}^{j+1} &= p_{lk}^j \bigcup p_{l,j+1}^j \left(p_{j+1,j+1}^j\right)^* p_{j+1,k}^j. \end{aligned}\right\} \tag{30}$$

Proof: Since the states are numbered form 1 to n in increasing order, $p_{lk}^0 = \{\sigma \in \Sigma : \delta(q_l, \sigma) = q_k\}$ follows directly form the state transition map $\delta : Q \times \Sigma \to Q$ and Definition 15.

Given $p_{lk}^j \subseteq p_{lk}^{j+1}$, let us consider the set $p_{lk}^{j+1} - p_{lk}^j$ in which each string passes through q_{j+1} in the path from q_l to q_k and no string must pass through q_m for $m > (j+1)$. Then, it follows that

$$p_{lk}^{j+1} - p_{lk}^j = p_{l,j+1}^j p_{j+1,k}^{j+1}$$

where $p_{j+1,k}^{j+1}$ can be expanded as

$$p_{j+1,k}^{j+1} = \left(p_{j+1,j+1}^j p_{j+1,k}^{j+1}\right) \bigcup p_{j+1,k}^j$$

that has a unique solution following Theorem 1 because $\epsilon \notin p_{j+1,j+1}^j$ based on Definition 15. Therefore,

$$p_{lk}^{j+1} = p_{lk}^j \bigcup p_{l,j+1}^j \left(p_{j+1,j+1}^j\right)^* p_{j+1,k}^j. \qquad \square$$

Based on the three lemmas proved below, the above relations can be transformed into an algebraic equation conceptually similar to Theorem 1 in Method I. Along with the procedure to compute the language measure it is established that, $\forall i \in \mathcal{I}_Q$, $\sum_{j=1}^n \pi_{ij} < 1$ is a sufficient condition for finiteness of μ.

Lemma 2: $\nu((p_{kk}^0)^*(\cup_{j\neq k}p_{kj}^0)) \in [0,1)$.

Proof: Following Definitions 6 and 8, $\nu(p_{kk}^0) \in [0,1)$. Therefore, by convergence of geometric series,

$$\nu\left((p_{kk}^0)^*\left(\bigcup_{j\neq k}p_{kj}^0\right)\right) = \frac{\sum_{j\neq k}\nu\left(p_{kj}^0\right)}{1-\nu\left(p_{kk}^0\right)} \in [0,1)$$

because $\sum_j \nu(p_{kj}^0) \Rightarrow \sum_{j\neq k}\nu(p_{kj}^0) - \nu(p_{kk}^0)$. □

Lemma 3: $\nu(p_{j+1,j+1}^j) \in [0,1)$.

Proof: The path $p_{j+1,j+1}^j$ may contain at most j loops, one around each of the states $q_1, q_2, \ldots, q_j$. If the path $p_{j+1,j+1}^j$ does not contain any loop, then $\nu(p_{j+1,j+1}^j) \in [0,1)$ because $\forall s \in p_{j+1,j+1}^j$, $\nu(s) < 1$ and each of s originates at state $j+1$.

Next let us suppose that there is a loop around q_ℓ and that does not contain any other loop; this loop must be followed by one or more events σ_k generated at q_ℓ and leading to some other states q_m where $m \in \{1,\ldots,j+1\}$ and $m \neq \ell$. By Lemma 2, $\nu(p_{j+1,\,j+1}^j) \in [0,1)$. Proof follows by starting from the innermost loop and ending with all loops at q_j □

Lemma 4:

$$\nu\left(\left(p_{j+1,j+1}^j\right)^*\right) = \frac{1}{1-\nu\left(p_{j+1,j+1}^j\right)} \in [1,\infty). \tag{31}$$

Proof: Since $\nu(p_{j+1,j+1}^j) \in [0,1)$ from Lemma 3.3, it follows that

$$\nu\left(\left(p_{j+1,j+1}^j\right)^*\right) = \frac{1}{1-\nu\left(p_{j+1,j+1}^j\right)} \in [1,\infty). \quad \square$$

Finally, the main result of this section is stated as the following theorem.

Theorem 4: *Given a DFSA* $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m\rangle$ *the following recursive result holds for* $0 \le j \le n-1$, where $\|Q\| = n$.

$$\nu\left(p_{lk}^{j+1}\right) = \nu\left(p_{lk}^j\right) + \frac{\nu\left(p_{l,j+1}^j\right)\nu\left(p_{j+1,k}^j\right)}{1-\nu\left(p_{j+1,j+1}^j\right)}. \tag{32}$$

Proof:

$$\begin{aligned}
\nu\left(p_{lk}^{j+1}\right) &= \nu\left(p_{lk}^j \bigcup p_{l,j+1}^j\left(p_{j+1,j+1}^j\right)^* p_{j+1,k}^j\right)\\
&= \nu\left(p_{lk}^j\right) + \nu\left(p_{l,j+1}^j\left(p_{j+1,j+1}^j\right)^* p_{j+1,k}^j\right)\\
&= \nu\left(p_{lk}^j\right) + \nu\left(p_{l,j+1}^j\right)\nu\left(\left(p_{j+1,j+1}^j\right)^*\right)\nu\left(p_{j+1,k}^j\right)\\
&= \nu\left(p_{lk}^j\right) + \frac{\nu\left(p_{l,j+1}^j\right)\nu\left(p_{j+1,k}^j\right)}{1-\nu\left(p_{j+1,j+1}^j\right)}.
\end{aligned}$$

The second step in the above derivation follows from fact that $p_{lk}^j \cap p_{l,j+1}^j(p_{j+1,j+1}^j)^* p_{j+1,k}^j = \emptyset$. The third step follows from Definition 8 and the last step is a consequence of Lemma 4. □

Based on the above result, a recursive algorithm to compute a language measure is presented below.

Algorithm 2: Recursive computation of the language measure

(1) For a given $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m\rangle$, specify the characteristic vector $\mathbf{X}$ (see Definition 5) and determine the event cost matrix $\widetilde{\mathbf{\Pi}}$ (see Definition 6) via experimentation or simulation, as described in § 5).
(2) Compute the $\mathbf{\Pi}$-matrix (Definition 7).
(3) $\nu(p_{lk}^0) \longleftarrow \pi_{lk}$ for $1 \le l,k \le n$
(4) for $j = 0$ to $n-1$
  for $l = 1$ to n
    for $k = 1$ to n

$$\nu(p_{lk}^{j+1}) = \nu(p_{lk}^j) + \frac{\nu(p_{l,j+1}^j)\nu(p_{j+1,k}^j)}{1-\nu(p_{j+1,j+1}^j)}$$

    end
  end
end
(5) Calculate $\nu(L_{i,k})$ from $\nu(p_{ik})$ using Definition 15.
(6) $\mu_i \longleftarrow \sum_{q_j\in Q_m}\nu(L_{i,J})\chi_j$ is a measure of the language L_i of the DFSA G_i.

Since there are only three *for* loops, the computational complexity of the above algorithm is polynomial in the number of DFSA states, same as that of Algorithm 1 in Method I.

4. Event cost: a probabilistic interpretation

The signed real measure (see Definition 10) of a regular language is based on the assignment of the characteristic vector $\mathbf{X}$ (see Definition 5) and the event cost matrix $\tilde{\mathbf{\Pi}}$ (Definition 6). The characteristic vector is chosen by the designer based on his/her perception of the individual state's impact on the system performance. On the other hand, the event cost is an intrinsic property of the plant. The event cost $\tilde{\pi}_{jk}$ is conceptually similar to the state-based conditional probability of Markov Chains, except for the fact that it is not allowed to satisfy the equality condition $\sum_k \tilde{\pi}_{jk} = 1$. (Note that $\sum_k \tilde{\pi}_{jk} < 1$ is a requirement for convergence of the language measure.) The rationale for this strict inequality is explained below.

Since the plant model is an inexact representation of the physical plant, there exist unmodelled dynamics to account for. This can manifest itself either as unmodelled events that may occur at each state or as unaccounted states in the model. Let Σ_j^u denote the set

of all unmodelled events at state q_j of the DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$. Creating a new unmarked absorbing state q_{n+1}, called the dump state (Ramadge and Wonham 1987), and extending the transition function δ to $\delta_{\text{ext}}: (Q \cup \{q_{n+1}\}) \times (\Sigma \cup_j \Sigma_j^u) \to (Q \cup \{q_{n+1}\})$, it follows that

$$\delta_{\text{ext}}(q_j, \sigma) = \begin{cases} \delta(q_j, \sigma), & \text{if } q_j \in Q \text{ and } \sigma \in \Sigma \\ q_{n+1}, & \text{if } q_j \in Q \text{ and } \sigma \in \Sigma_j^u \\ q_{n+1}, & \text{if } j = n+1 \text{ and } \sigma \in \Sigma \bigcup \Sigma_j^u. \end{cases} \tag{33}$$

Therefore the residue $\theta_j = 1 - \sum_k \tilde{\pi}_{jk}$ denotes the probability of the set of unmodelled events Σ_j^u conditioned on the state j. The $\mathbf{\Pi}$ matrix can be similarly augmented to obtain a stochastic matrix $\mathbf{\Pi}_{\text{aug}}$ as follows:

$$\mathbf{\Pi}_{\text{aug}} = \begin{bmatrix} \pi_{11} & \pi_{12} & \dots & \pi_{1n} & \theta_1 \\ \pi_{21} & \pi_{22} & \dots & \pi_{2n} & \theta_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \pi_{n1} & \pi_{n2} & \dots & \pi_{nn} & \theta_n \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}. \tag{34}$$

Since the dump state q_{n+1} is not marked, its characteristic value $\chi_{n+1} \equiv \chi(q_{n+1}) = 0$. The characteristic vector then augments to

$$\mathbf{X}_{\text{aug}} = [\mathbf{X}^T \quad 0]^T$$

and, with these extensions, the language measure vector $\boldsymbol{\mu}_{\text{aug}} \equiv [\mu_1 \ \ \mu_2 \ \ \cdots \ \ \mu_n \ \ \mu_{n+1}]^T = [\boldsymbol{\mu}^T \ \ \mu_{n+1}]^T$ of the augmented DFSA $G_{\text{aug}} \equiv \langle Q \cup \{q_{n+1}\}, \Sigma \cup_j \Sigma_j^u, \delta_{\text{ext}}, q_i, Q_m \rangle$ can be expressed as

$$\boldsymbol{\mu}_{\text{aug}} \equiv \begin{pmatrix} \boldsymbol{\mu} \\ \mu_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{\Pi}\boldsymbol{\mu} + \mu_{n+1}[\theta_1 \quad \cdots \quad \theta_n]^T \\ \mu_{n+1} \end{pmatrix} + \begin{pmatrix} \mathbf{X} \\ 0 \end{pmatrix}. \tag{35}$$

Since $\chi(q_{n+1}) = 0$ and all transitions from the absorbing state q_{n+1} lead to itself, i.e., $\mu_{n+1} = \mu(L_m(G_{n+1})) = 0$, equation (35) reduces to that for the original plant G_i. Thus, the event cost can now be interpreted as conditional probability, where the residue $\theta_j = 1 - \sum_k \tilde{\pi}_{jk} > 0$ accounts for the probability of all unmodelled events emanating from the state q_j. With this interpretation of event cost, $\tilde{\pi}[s, q_i]$ (see Definition 6) denotes the probability of occurrence of the event string s in the plant model G_i starting at state q_i and terminating at state $\hat{\delta}(s, q_i)$. Hence, $\nu^i(L_{i,j})$ (see Definition 8), which is a non-negative real number, is directly related to the sum of probabilities that state q_i would be reached via alternative paths as the plant operates. (Note that $\nu^i(L_{i,j}) > 1$ is possible if $L_{i,j}$ contains multiple strings.) The language measure $\mu_i \equiv \mu^i(L(G_i)) = \sum_{j \in \mathcal{I}_Q} \mu^i(L_{i,j}) = \sum_{j \in \mathcal{I}_Q} \nu^i(L_{i,j}) \chi_j$ is then directly related (but not necessarily equal) to the expected value of the characteristic function.

The choice of the characteristic function (see Definition 5) is based on the importance assigned to the individual marked states of the DFSA. Therefore, in the setting of the language measure, a supervisor's performance is superior if the supervised plant is more likely to terminate at a good marked state and/or less likely to terminate at a bad marked state.

5. Estimation of language measure parameters

This section presents a recursive algorithm for identification of the language measure parameters (Wang *et al.* 2005) (i.e., elements of the event cost matrix $\widetilde{\mathbf{\Pi}}$) (see Definition 6) which, in turn, allows computation of the state transition cost matrix $\mathbf{\Pi}$ (see Definition 7) and the language measure $\boldsymbol{\mu}$-vector (see Definition 10). It is assumed that the underlying physical process evolves at two different time scales. In the fast-time scale, i.e., over a short time period, the system is assumed to be an ergodic, discrete Markov process. In the slowly-varying time scale, i.e., over a long period, the system (possibly) behaves as a non-stationary stochastic process. For such a slowly-varying non-stationary process, it might be necessary to redesign the supervisory control policy in real time. In that case, the $\widetilde{\mathbf{\Pi}}$-matrix parameters should be updated at selected slow-time epochs.

5.1. *A recursive parameter estimation scheme*

Let p_{ij} be the transition probability of the event σ_j at the state q_i, i.e.,

$$p_{ij} = \begin{cases} P[\sigma_j | q_i], & \text{if } \exists q \in Q, \text{ s.t. } q = \delta(q_i, \sigma_j) \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

and its estimate be denoted by the parameter $\hat{p}_{ij}$ that is to be identified from the ensemble of simulation and/or experimental data.

Let a strictly increasing sequence of time epochs of consecutive event occurrence be denoted as

$$\mathcal{T} \equiv \{t_k : k \in N_0\}, \tag{37}$$

where $\mathbf{N}_0$ is the set of non-negative integers. Let the indicator $\psi: \mathbf{N}_0 \times \mathcal{I}_Q \times \mathcal{I}_\Sigma \to \{0, 1\}$ represent the incident of occurrence of an event. For example, if the DFSA was in state q_i at time epoch t_{k-1}, then

$$\psi_{ij}(k) = \begin{cases} 1, & \text{if } \sigma_j \text{ occurs at the time epoch } t_k \in \mathcal{T} \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

Consequently, the number of occurrences of any event in the alphabet Σ is represented by $\Psi: \mathbf{N}_0 \times \mathcal{I}_Q \to \{0,1\}$. For example, if the DFSA was in state q_i at the time epoch t_{k-1}, then

$$\mathbf{\Psi}_i(k) = \sum_{j \in \mathcal{I}_\Sigma} \psi_{ij}(k). \tag{39}$$

Let n: $\mathbf{N}_0 \times \mathcal{I}_Q \times \mathcal{I}_\Sigma \to \mathbf{N}_0$ represent the cumulative number of occurrences of an event at a state up to a given time epoch. That is, $n_{ij}(k)$ denotes the number of occurrences of the event σ_j at the state q_i up to the time epoch $t_k \in \mathcal{T}$. Similarly, let N: $\mathbf{N}_0 \times \mathcal{I}_Q \to \mathbf{N}_0$ represent the cumulative number of occurrences of any event in the alphabet Σ at a state up to a given time epoch. Consequently,

$$N_i(k) = \sum_{j \in \mathcal{I}_\Sigma} n_{ij}(k). \tag{40}$$

A frequency estimator, $\hat{p}_{ij}(k)$, for probability $p_{ij}(k)$ of the event σ_j occurring at the state q_i at the time epoch t_k, is obtained as

$$\left.\begin{aligned} \hat{p}_{ij}(k) &= \frac{n_{ij}(k)}{N_i(k)} \\ \lim_{k \to \infty} \hat{p}_{ij}(k) &= p_{ij}. \end{aligned}\right\} \tag{41}$$

Convergence of the above limit is justified because the occurrence of an event at a given state of a stationary Markov chain can be treated as an independent and identically distributed random variable.

A recursive algorithm of learning p_{ij} is formulated as a stochastic approximation scheme, starting at the time epoch t_0 with the initial conditions: $\hat{p}_{ij}(0) = 0$ and $n_{ij}(0) = 0$ for all $i \in \mathcal{I}_Q, j \in \mathcal{I}_\Sigma$; and $\mathbf{\Psi}_i(0) = 0$ for all $i \in \mathcal{I}_Q$. Starting at $k = 0$, the recursive algorithm runs for $\{t_k : k \geq 1\}$. For example, upon occurrence of an event σ_j at a state q_i, the algorithm is recursively incremented as

$$\left.\begin{aligned} n_{ij}(k) &= n_{ij}(k-1) + \psi_{ij}(k) \\ N_i(k) &= N_i(k-1) + \Psi_i(k). \end{aligned}\right\} \tag{42}$$

Next it is demonstrated how the estimates of the language parameters (i.e., the elements of event cost matrix $\widetilde{\Pi}$) are determined from the probability estimates. As stated earlier in § 4 the set of unmodelled events at state q_i, denoted by $\Sigma_i^u \ \forall i \in \mathcal{I}_Q$, accounts for the row-sum inequality: $\sum_j \tilde{\pi}_{ij} <$ (see Definition 6). Then, $\mathrm{P}[\Sigma_i^u] = \theta_i \in (0,1]$ and $\sum_i \tilde{\pi}_{ij} = 1 - \theta_i$. An estimate of the (i,j)th element of the event cost matrix $\widetilde{\mathbf{\Pi}}$-matrix, denoted by $\hat{\tilde{\pi}}_{ij}$, is approximated as

$$\hat{\tilde{\pi}}_{ij}(k) = \hat{p}_{ij}(k)(1 - \theta_i) \quad \forall j \in \mathcal{I}_\Sigma. \tag{43}$$

Additional experiments on a more detailed automaton model would be necessary to identify the parameters $\theta_i \ \forall i \in \mathcal{I}_Q$. If $\theta_i \ll 1$, the problem of conducting additional experimentation can be circumvented by the following approximation.

A single parameter $\theta \approx \theta_i \ \forall i \in \mathcal{I}_Q$, $i \in \mathcal{I}_Q$, such that $0 < \theta \ll 1$, could be selected for convenience of implementation. From the numerical perspective, this option is meaningful because it sets an upper bound on the language measure based on the fact that the max norm $\|\mu\|_\infty \leq \theta^{-1}$. Note that each row sum in the $\widetilde{\mathbf{\Pi}}$-matrix being strictly less than 1, i.e., $\sum_j \tilde{\pi}_{ij} < 1$, is a sufficient condition for finiteness of the language measure (see Appendix II).

Theoretically, $\tilde{\pi}_{ij}$ is the asymptotic value of the estimated probabilities $\hat{\tilde{\pi}}_{ij}(k)$ as if the event σ_j occurs infinitely many times at the state q_i. However, while dealing with finite amount of data, the objective is to obtain a good estimate $\hat{p}_{ij}$ of p_{ij} from independent Bernoulli trials of generating events. Critical issues in this situation are: (i) how much data are needed; and (ii) when to stop if adequate data are available. The next section 5-B addresses these issues.

5.2. *Stopping rules for recursive learning*

A stopping rule is necessary to find a lower bound on the number of experiments to be conducted for identification of the $\widetilde{\mathbf{\Pi}}$-matrix parameters. This section presents two stopping rules that are discussed below.

The first stopping rule is based on an inference approximation having a specified absolute error bound ε with a probability λ. The objective is to achieve a trade-off between the number of experimental observations and the estimation accuracy.

A bound on the required number of samples is estimated using the Gaussian structure of the binomial distribution that is an approximation of the sum of a large number of independent and identically distributed (i.i.d.) Bernoulli trials of $\hat{\tilde{\pi}}_{ij}(t)$. The central limit theorem yields $\hat{\tilde{\pi}}_{ij} \sim \mathcal{N}(\tilde{\pi}_{ij}, \tilde{\pi}_{ij}(1 - \tilde{\pi}_{ij})/N)$, where $\mathcal{N}$ indicates normal (or Gaussian) distribution with $E[\hat{\tilde{\pi}}_{ij}] \approx \tilde{\pi}_{ij}$ and $\mathrm{Var}[\hat{\tilde{\pi}}_{ij}] \equiv \sigma^2 \approx \tilde{\pi}_{ij}(1 - \tilde{\pi}_{ij})/N$, provided that the number of samples N is sufficiently large. Let $\Delta = \hat{\tilde{\pi}}_{ij} - \tilde{\pi}_{ij}$, then $\Delta/\sigma \sim \mathcal{N}(0,1)$. Given $0 < \varepsilon \ll 1$ and $0 < \lambda \ll 1$, the problem is to find a bound N_b on the number N of experiments such that $P\{|\Delta| \geq \varepsilon\} \leq \lambda$. Equivalently,

$$P\left\{\frac{|\Delta|}{\sigma} \geq \frac{\varepsilon}{\sigma}\right\} \leq \lambda \tag{44}$$

that yields a bound N_b on N as

$$N_b \geq \left(\frac{\xi^{-1}(\lambda)}{\varepsilon}\right)^2 \tilde{\pi}_{ij}(1 - \tilde{\pi}_{ij}), \tag{45}$$

where $\xi(x) \equiv 1 - \sqrt{2/\pi}\int_0^x e^{-(t^2/2)}\,\mathrm{d}t$. Since the parameter $\tilde{\pi}_{ij}$ is unknown, one may use the fact that $\tilde{\pi}_{ij}(1-\tilde{\pi}_{ij}) \leq 0.25$ for every $\tilde{\pi}_{ij} \in [0,1]$ to (conservatively) obtain a bound on N only in terms of the specified parameters ε and λ as

$$N_b \geq \left(\frac{\xi^{-1}(\lambda)}{2\varepsilon}\right)^2. \tag{46}$$

The above estimate of the bound on the required number of samples is less conservative than that obtained from the Chernoff bound and is significantly less conservative than that obtained from Chebyshev bound which does not require the assumption of any specific distribution of Δ except for finiteness of the rth ($r = 2$) moment.

The second stopping rule, which is an alternative to the first stopping rule, is based on the properties of irreducible stochastic matrices. Following equation (41) and the state transition function δ of the DFSA, the state transition matrix is constructed at the kth iteration as $\mathbf{P}(k)$ that is an $n \times n$ irreducible stochastic matrix under stationary conditions. Similarly, the state probability vector $\mathbf{p}(k) \equiv [p_1(k)\ p_2(k)\ \cdots\ p_n(k)]$ is obtained by following equation (41)

$$p_i(k) = \frac{N_i(k)}{\sum_{j\in\mathcal{I}_Q} N_j(k)}. \tag{47}$$

The stopping rule makes use of the Perron–Frobenius Theorem to establish a relation between the vector $\mathbf{p}(k)$ and the irreducible stochastic matrix $\mathbf{P}(k)$.

Theorem 5: *Perron–Frobenius Theorem* (*Senata 1973*, *Plemmons and Berman 1979*) *Let* $\mathbf{P}(k)$ *be an* $n \times n$ *irreducible matrix*, *then there exits an eigenvalue r such that*

(1) $r \in \mathbb{R}$ *and* $r > 0$.
(2) *r can be associated strictly positive left and right eigenvectors.*
(3) $r \geq \lambda\ \forall$ *eigenvalue* $\lambda \neq r$.
(4) *The eigenvectors associated with r are unique to constant multiples.*
(5) *If* $0 \leq B \leq P(k)$ *and* β *is an eigenvalue of B, then* $|\beta| \leq |r|$. *Moreover,* $|\beta| = r$ *implies* $B = P(k)$.
(6) *r is a simple root of the characteristic equation of* $P(k)$.

Corollary 1: *Corollary to Perron–Frobenius Theorem*

$$\min_i \sum_{j=1}^n P_{ij}(k) \leq r \leq \max_i \sum_{j=1}^n P_{ij}(k)$$

with equality on either side implying equality throughout.

Since $\mathbf{P}(k)$ is a stochastic matrix, i.e., $\sum_{j=1}^n P_{ij}(k) = 1$, and $\mathbf{P}(k)$ is irreducible, there is a unique eigenvalue $r = 1$ and the corresponding left eigenvector $\mathbf{p}(k)$ (normalized to unity in the sense of absolute sum) representing the state probability vector, provided that the matrix parameters have converged after sufficiently large number of iterations. That is,

$$\|\mathbf{p}(k)(\mathbf{I} - \mathbf{P}(k))\|_\infty \leq \frac{1}{k} \to 0 \quad \text{as } k \to \infty.$$

Equivalently,

$$\|(\mathbf{p}(k) - \mathbf{p}(k+1))\|_\infty \leq \frac{1}{k} \to 0 \quad \text{as } k \to \infty. \tag{48}$$

Taking the expected value of $\|\mathbf{p}(k)\|_\infty$ to be $1/n$, a threshold of η/n is specified, where n is the number of states and $0 < \eta \ll 1$ is a constant. A lower bound on the required number of samples is determined from equation (48) as

$$N_{\text{stop}} \equiv \text{Integer}\left(\frac{n}{\eta}\right) \tag{49}$$

based on the number of states, n, and the specified tolerance η.

6. Usage of the language measure

The two methods of language measure computation, presented in § 3, have the same computational complexity, $\mathcal{O}(n^3)$, where n is the number of states of the DFSA. However, each of these two methods offer distinct relative advantages in specific contexts. For example, while the closed form solution in § 3.1 is more amenable for analysis and synthesis of decision and control algorithms, the recursive solution in § 3.2 might prove very useful for construction of executable codes in real time applications. The following two subsections present usage of the language measure for construction of metric spaces of formal languages and synthesis of optimal discrete-event supervisors.

6.1. *Vector space of formal languages*

The language measure can be used to construct a vector space of sublanguages for a given DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m\rangle$. The total variation measure $|\mu|$ (Rudin 1987) (see Appendix I) induces a metric on this space, which quantifies the distance function between any two sublanguages of $L(G_i)$.

Proposition 1: *Let* $L(G_i)$ *be the language of a DFSA* $G_i = \langle Q, \Sigma, \delta, q_i, Q_m\rangle$. *Let the binary operation of*

exclusive-OR $\oplus$: $2^{L(G_i)} \times 2^{L(G_i)} \to 2^{L(G_i)}$ *be defined as*

$$(K_1 \oplus K_2) = \left(K_1 \bigcup K_2\right) - \left(K_1 \bigcap K_2\right) \qquad (50)$$

$\forall K_1, K_2 \subseteq L(G_i)$. *Then* $(2^{L(G_i)}, \oplus)$ *is a vector space over Galois field* $GF(2)$.

Proof: It follows from the properties of exclusive-OR that the algebra $(2^{L(G_i)}, \oplus)$ is an Abelian group where $\emptyset$ is the zero element of the group and the unique inverse of every element $K \subseteq 2^{L(G_i)}$ is K itself because $K_1 \oplus K_2 = \emptyset$ if and only if $K_1 = K_2$. The associative and distributive properties of the vector space follows by defining the scalar multiplication of vectors as: $0 \otimes K = \emptyset$ and $1 \otimes K = K$. □

The collection of singleton languages made from each element of $L(G_i)$ forms a basis set of vector space $(2^{L(G_i)}, \oplus)$ over $GF(2)$. It is shown below, how total variation (Rudin 1987) of the signed measure μ can be used to define a metric on above vector space.

Proposition 2: *Total variation measure* $|\mu|$ *on* $2^{L(G_i)}$ *is given by* $|\mu|(L) = \sum_{s \in L} |\mu(\{s\})|$ $\forall L \subseteq L(G_i)$.

Proof: The proof follows from the fact that $\Sigma_k |\mu(L_k)|$ attains its supremum for the finest partition of L which consists of the individual strings in L as elements of the partition. □

Corollary 2 (to Proposition 2)**:** *Let* $L(G_i)$ *be the language of a DFSA* $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$. *For any* $K \in 2^{L(G_i)}$, $|\mu|(K) \leq \theta^{-1}$ *where* $\theta = 1 - \|\mathbf{\Pi}\|_\infty$ *and* $\mathbf{\Pi}$ *is the state transition cost matrix of the DFSA.*

Proof: The proof follows from Proposition 2 and Corollary 1. □

Definition 16: Let $L(G_i)$ be the language of a DFSA $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$. The distance function $d: 2^{L(G_i)} \times 2^{L(G_i)} \to [0, \infty)$ is defined in terms of the total variation measure as $\forall K_1, K_2 \subseteq L(G_i)$

$$d(K_1, K_2) = |\mu|\left(\left(K_1 \bigcup K_2\right) - \left(K_1 \bigcap K_2\right)\right) \qquad (51)$$

The above distance function $d(\cdot, \cdot)$ quantifies the difference between two supervisors relative to the supervised performance of the DFSA plant.

Proposition 3: *The distance function defined above is a pseudo-metric on the space* $2^{L(G_i)}$

Proof: Since the total variation of a signed real measure is bounded (Rudin 1987), $\forall K_1, K_2 \subseteq L(G_i)$, $d(K_1, K_2) = |\mu|(K_1 \oplus K_2) \in [0, \infty)$; also by Definition 16, $d(K_1, K_2) = d(K_2, K_1)$. The remaining property of the triangular inequality follows from the inequality $|\mu|(K_1 \oplus K_2) \leq |\mu|(K_1) + |\mu|(K_2)$ which is based on the fact that $(K_1 \oplus K_2) \subseteq (K_1 \cup K_2)$ and $|\mu|(K_1) \leq |\mu|(K_2)$ $\forall K_1 \subset K_2$. □

The pseudo-metric $|\mu|: 2^{L(G_i)} \to [0, \infty)$ can be converted to a metric of the space $(2^{L(G_i)}, \oplus)$ by clustering all languages that have zero total variation measure as the null equivalence class $\mathcal{N} \equiv \{K \in 2^{L(G_i)}: |\mu|(K) = 0\}$. This procedure is conceptually similar to what is done for defining norms in the L_p spaces. In that case, $\mathcal{N}$ contains all sublanguages of $L(G_i)$, which terminate on non-marked states starting from the initial state, i.e., $\mathcal{N} = \{\emptyset \cup (\cup_{q_j \notin Q_m} L_{i,j})$. In the sequel, $|\mu|(\cdot)$ is referred to as a metric of the space $2^{L(G_i)}$. Thus, the metric $|\mu|(\cdot)$ can be generated from $d(\cdot, \cdot)$ as: $|\mu|(K) = d(K, J)$ $\forall K \in 2^{L(G_i)}$ $\forall J \in \mathcal{N}$. Unlike the norms on vector spaces defined over infinite fields, the metric $|\mu|(\cdot)$ for the vector space $(2^{L(G_i)}, \oplus)$ over $GF(2)$ is not a functional. This interpretation of language as a vector and associating a metric to quantify distance between languages, may be useful for analysis and synthesis of discrete-event supervisory (DES) control systems under different settings.

6.2. *Optimal control of regular languages*

The (signed) language measure μ could serve as the performance index for synthesis of an optimal control policy that maximizes the performance of a supervised sublanguage. The salient concept is briefly presented below.

Let $\mathcal{S} \equiv \{S^0, S^1, \ldots, S^N\}$ be a set of supervisory control policies for the unsupervised plant automaton G where S^0 is the null controller (i.e., no event is disabled) implying that $L(S^0/G) = L(G)$. Therefore, the controller cost matrix $\mathbf{\Pi}(S^0) = \mathbf{\Pi}^0$ that is the $\mathbf{\Pi}$-matrix of the unsupervised plant automaton G. For a supervisor $S^k, k \in \{1, 2, \ldots, N\}$, the control policy is required to selectively disable certain controllable events so that the following (elementwise) inequality holds.

$$\mathbf{\Pi}^k \equiv \mathbf{\Pi}(S^k) \leq \mathbf{\Pi}^0 \text{ and } L(S^k/G) \subseteq L(G), \quad \forall S^k \in \mathcal{S}.$$

The task is to synthesize an optimal cost matrix $\mathbf{\Pi}^* \leq \mathbf{\Pi}^0$ that maximizes the performance vector $\boldsymbol{\mu}^* \equiv [\mathbf{I} - \mathbf{\Pi}^*]^{-1}\mathbf{X}$, i.e., $\boldsymbol{\mu}^* \geq \boldsymbol{\mu}^k \equiv [\mathbf{I} - \mathbf{\Pi}^k]^{-1}\mathbf{X}$ $\forall$ $\mathbf{\Pi}^k \leq \mathbf{\Pi}^0$ where the inequalities are implied elementwise. While the details of the underlying theory are available in recent literature (Ray *et al.* 2004), a synthesis procedure for optimal control of regular languages is succinctly presented below.

Let the DFSA model G of the unsupervised plant have the state transition cost matrix: $\mathbf{\Pi}^0 \equiv \mathbf{\Pi}$ (see Definition 7) and the characteristic vector $\mathbf{X}$ (see Definition 5). Then, the performance vector at the

iteration $k = 0$ is given as

$$\boldsymbol{\mu}^0 = [\mu_1^0 \quad \mu_2^0 \quad \ldots \quad \mu_n^0]^T = (I - \boldsymbol{\Pi}^0)^{-1}\mathbf{X}$$

where the jth element μ_j^0 of the vector μ^0 is the performance of the unsupervised plant language, with state q_j as the initial state. Then, $\mu_j^0 < 0$ implies that, if the state q_j is reached, then the plant will yield bad performance thereafter. Intuitively, the control system should attempt to prevent the automaton from reaching q_j by disabling all controllable events that lead to this state. Therefore, the optimal control algorithm starts with disabling all controllable events that lead to every state q_j for which $\mu_j^0 < 0$. This is equivalent to reducing all elements of the corresponding columns of the $\boldsymbol{\Pi}^0$-matrix by disabling those controllable events. In the next iteration, i.e., $k = 1$, the updated cost matrix $\boldsymbol{\Pi}^1$ is obtained as: $\boldsymbol{\Pi}^1 = \boldsymbol{\Pi}^0 - \boldsymbol{\Delta}^0$ where $\boldsymbol{\Delta}^0 \geq 0$ (the inequality being implied elementwise) is composed of event costs corresponding to all controllable events that have been disabled.

It has been shown in Ray *et al.* (2004) that $\boldsymbol{\mu}^{k-1} \leq \boldsymbol{\mu}^k \equiv [I - \boldsymbol{\Pi}^k]^{-1}\mathbf{X}$ elementwise for all $k \geq 1$. Although all controllable events leading to every state corresponding to a negative element of $\boldsymbol{\mu}^1$ are disabled, some of the controllable events that were disabled at $k = 0$ may now lead to states corresponding to positive elements of $\boldsymbol{\mu}^1$. Performance could be further enhanced by re-enabling these controllable events. For $k \geq 1$, $\boldsymbol{\Pi}^{k+1} = \boldsymbol{\Pi}^k + \boldsymbol{\Delta}^k$ where $\boldsymbol{\Delta}^k \geq 0$ is composed of the state transition costs of all re-enabled controllable events at k. It is also shown in Ray *et al.* (2004) that the number of iterations to reach optimality does not exceed the number, n, of *DFSA* states. Therefore, the computational complexity of the optimal control algorithm is polynomial n.

In the optimal control algorithm in Ray *et al.* (2004), if $\boldsymbol{\mu}^0 \geq 0$, i.e., there is no state q_j such that $\boldsymbol{\mu}_j^0 < 0$, then the plant performance cannot be improved by event disabling and the null controller S^0 (i.e., no disabled event) is the optimal controller for the given plant. Therefore, the cases are considered where μ_j^0 for some state q_j.

Starting with $k = 0$ and $\boldsymbol{\Pi}^0 \equiv \boldsymbol{\Pi}^{\text{plant}}$, the control policy is constructed by the following two-step procedure:

Step 1: For every state q_j for which $\mu_j^0 < 0$, disable controllable events leading to q_j. Now, $\boldsymbol{\Pi}^1 = \boldsymbol{\Pi}^0 - \boldsymbol{\Delta}^0$, where $\boldsymbol{\Delta}^0 \geq 0$ is composed of event costs corresponding to all controllable events, leading to q_j for which $\boldsymbol{\mu}_j^0$, which have been disabled at $k = 0$.

Step 2: For $k \geq 1$, if $\mu_j^k \geq 0$, re-enable all controllable events leading to q_j, which were disabled in Step 1. The cost matrix is updated as: $\boldsymbol{\Pi}^{k+1} = \boldsymbol{\Pi}^k + \boldsymbol{\Delta}^k$ for $k \geq 1$, where $\boldsymbol{\Delta}^k \geq 0$ is composed of event costs corresponding to all currently re-enabled controllable events. The iteration is terminated if no controllable event leading to q_j remains disabled for which $\mu_j^k > 0$. At this stage, the optimal performance $\boldsymbol{\mu}^* \equiv [I - \boldsymbol{\Pi}^*]^{-1}\mathbf{X}$.

7. An application example

Ray and Phoha (2003) and Ray *et al.* (2004) have adopted the closed form method of language measure (see § 3.1) as a performance index for optimal supervisory control of a twin-engine unmanned aircraft that is used for surveillance and data collection. The language measure was computed and verified based on both closed form and recursive techniques given in § 3; the results were identical as expected. Engine health and operating conditions, which are monitored in real time based on observed data, are classified into three mutually exclusive and exhaustive categories:

- good;
- unhealthy (but operable);
- inoperable.

In the event of any observed abnormality, the supervisor may decide to continue or abort the mission. The finite state automaton model of the plant in figure 2 has 13 states (excluding the dump state), of which three are marked states, and nine events, of which four are controllable and the remaining five are uncontrollable. All events are assumed to be observable. The states and events of the plant model are listed in table 1 and table 2, respectively. The state transition function δ and the state-based event cost $\tilde{\pi}_{ij}$ (see Definition 6) are entered simultaneously in table 3. The values of $\tilde{\pi}_{ij}$ were selected by extensive experiments on engine simulation models and were also based on experience of gas

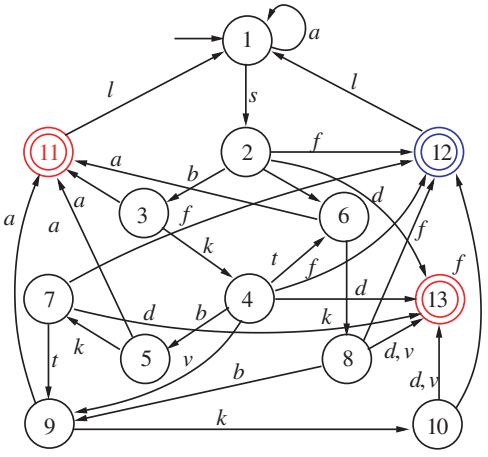

Figure 2. Unsupervised plant, i.e., no disabling of controllable events.

turbine engine operation and maintenance. The dump state and any transitions to the dumped state are not shown in table 3. The elements of the characteristic vector (see Definition 5) were chosen as signed real weights based on the perception of each marked state's role on the engine performance.

The characteristic (χ) values of the 13 states are assigned as: $\mathbf{X} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -0.05\ +0.25\ -1.0]^T$. These parameters are selected by the designer based on his/her perception of each marked state's role in the system performance. As the states 1 to 10 in in table 1 are not marked, the first 10 elements of the characteristic vector **X** are zeros. The implication is that event strings terminating at states 1 to 10 have no bearing on the system performance and hence have zero measure. The state 12 is a good marked state having a positive χ value and the bad marked states 11 and 13 have negative χ values. Therefore, event strings terminating at state 12 have positive measure and those terminating at states 11 and 13 have negative measure.

Four supervisory controllers were designed independently using a graphical interactive package (Wang *et al.* 2003) based on the following specifications.

(1) Specification #1: At least one of the two engines must be in good condition for mission continuation.
(2) Specification #2: None of the two engines must be in inoperable condition for mission continuation.
(3) Specification #3: Both engines must be in good condition for mission continuation.
(4) Optimal Control: The control policy is optimized following the two-step procedure in § 6.2.

The supervised plant automata under specifications #1, #2, #3 and optimal control are displayed in figures 3, 4, 5 and 6, respectively, where dashed lines indicate disabled controllable events. Notice that none of the controllable events are disabled in the unsupervised plant (see figure 2) and the four supervisors disable different sets of controllable events, as seen in figures 3–6.

Table 1. Plant automation states.

State	Description
1	Safe in base
2	Mission executing — two good engines
3	One engine unhealthy during mission execution
4	Mission executing — one good and one unhealthy engine
5	Both engines unhealthy during mission execution
6	One engine good and one engine inoperable
7	Mission execution with two unhealthy engines
8	Mission execution with only one good engine
9	One engine unhealthy and one engine inoperable
10	Mission execution with only one unhealthy engine
11	Mission aborted/not completed (Bad Marked State)
12	Mission successful (Good Marked State)
13	Aircraft destroyed (Bad Marked State)

Table 2. Plant event alphabet.

Event	Event description	Controllable event
s	Start and take-off	√
b	A good engine becoming unhealthy	
t	An unhealthy engine becoming inoperable	
v	A good engine becoming inoperable	
k	Keep engine(s) running	√
a	Mission abortion	√
f	Mission completion	
d	Destroyed aircraft	
l	Landing	√

Table 3. State transition and event cost matrix.

	s	*b*	*t*	*v*	*k*	*a*	*f*	*d*	*l*
1	0.50(2)					0.02(1)			
2		0.05(3)		0.01(6)			0.80(12)	0.10(13)	
3					0.45(4)	0.45(11)			
4		0.12(5)	0.16(6)	0.10(9)			0.50(12)	0.12(13)	
5					0.45(7)	0.45(11)			
6					0.45(8)	0.45(11)			
7			0.25(9)				0.50(12)	0.20(13)	
8		0.20(9)		0.01(13)			0.3(12)	0.4(13)	
9					0.45(10)	0.45(11)			
10			0.35(13)				0.20(12)	0.40(13)	
11									0.95(1)
12									0.95(1)
13									

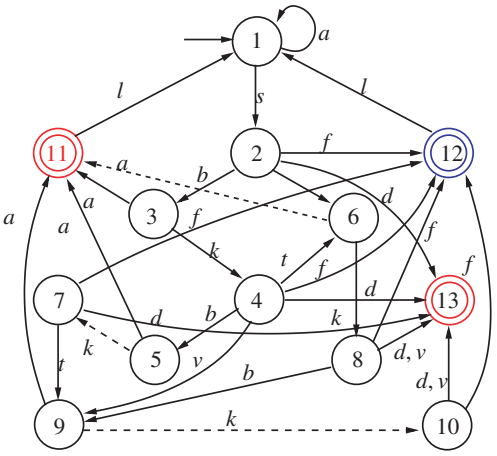

Figure 3. Supervised plant under specification #1.

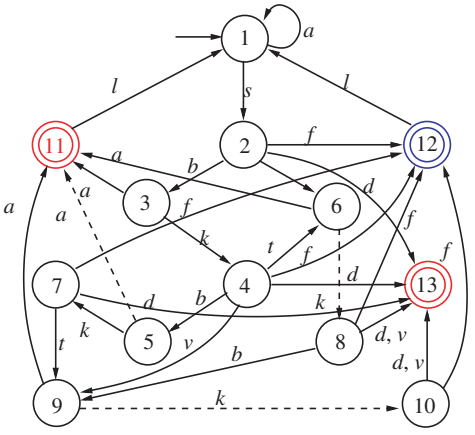

Figure 4. Supervised plant under specification #2.

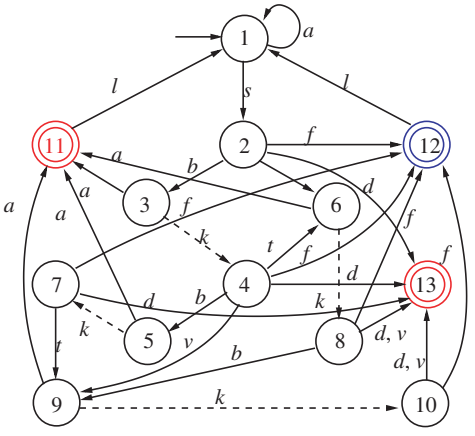

Figure 5. Supervised plant under specification #3.

The performance measure μ_1 (i.e., with the initial state 1) of the unsupervised (i.e., no disabling of control events) plant is 0.0823 and for three supervised plants under specifications #1, #2, #3 and the optimally

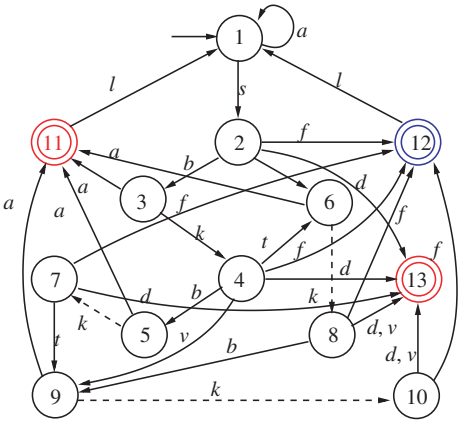

Figure 6. Optimally supervised plant, i.e., with best performance.

supervised plant are evaluated to be: 0.0807, 0.0822, 0.0840 and 0.0850, respectively. Therefore, the performance of the supervised plant under specifications #1, #2 and #3 is inferior, similar, and superior, respectively, to that of the unsupervised plant. As expected, the optimal supervisor has better performance than that of Supervisor #3. Notice that Supervisor #3 does disable the controllable event k from the state 3 to state 4 and the optimal supervisor does not. That is, the optimal supervisor allows continuing operation of an unhealthy engine while the remaining engine is in good condition.

8. Summary, conclusions, and future research

This paper reviews the concept, formulation and validation of a signed real measure for regular languages and their sublanguages based on the principles of automata theory and real analysis. While the domain of the measure μ, i.e., $2^{L(G_i)}$, is partially ordered, its range, which is a subset of $\mathbf{R} \equiv (-\infty, \infty)$, becomes totally ordered. As a result, the relative performance of different supervisors can be quantitatively evaluated in terms of the real signed measure of the supervised sublanguages. Positive weights are assigned to good marked states and negative weights to bad marked states so that a controllable supervisor is rewarded (penalized) for deleting strings terminating at bad (good) marked states. In order to evaluate and compare the performance of different supervisors a common quantitative tool is required. To this effect, the proposed procedure computes the measure of the supervised sublanguage generated by a supervisor using the event cost and characteristic function assigned for the unsupervised plant. Cost assignment to each event based on the state, where it is generated, has been shown similar to the conditional probability of the event. On the other hand, the characteristic function is chosen based on

the designer's perception of the individual state's impact on the system performance. Two techniques are presented to compute the language measure for a DFSA. One of them yields a closed form solution that is obtained as the unique solution of a set of linearly independent algebraic equations. The other is based on a recursive procedure. The computational complexity of both language measure algorithms is identical and is of polynomial order in the number of states of the DFSA. As such it is relatively straight-forward to develop software analysis tools in standard languages such as Matlab, C, and Java (Wang *et al*. 2003).

8.1. *Recommendations for future research*

Optimal discrete-event supervisory control can be enhanced through appropriate usage of the language measure. For example, in the current configuration of the optimal control algorithm (Ray *et al*. 2004), if there is no state q_j such that $\mu_j^0 < 0$, then the plant performance cannot be improved by event disabling and the null controller S^0 (i.e., no disabled event) is the optimal controller for the given plant. This restriction can be lifted through an appropriate performance index that would be a function of the language measure $\boldsymbol{\mu}$ but not identically equal to $\boldsymbol{\mu}$ as reported in Ray *et al*. (2004). Work in this direction is in progress and is expected to be reported in a forthcoming publication.

Synthesis of supervisory control systems may become a significant challenge if some of the events are delayed, intermittent, or not observable at all, possibly due to sensor faults or malfunctions in network communication links. In that case, the control algorithms may turn out to be computationally very complex because of delayed or lost information on the plant dynamics. Future work in this direction should involve research on construction of language measures under partial observation (Chattopadhyay and Ray 2004) and synthesis of optimal control policies under partial observation to mitigate the detrimental effects of loss of observability. The latter research could be an extension of the earlier work on optimal control under full observation (Ray *et al*. 2004).

It would be a challenging task to extend the concept of (regular) language measure for languages higher up in the Chomsky Hierarchy (Martin 1997) such as context free and context sensitive languages. This extension would lead to controller synthesis when the plant dynamics is modelled by non-regular languages such as the Petri-Net. The research thrust should focus on retaining the polynomial order of computational complexity.

Another critical issue is how to extend the language measure for timed automaton, especially if the events are observed with varying delays at different states. Another research topic that may also be worth investigating is: how to extend the field $GF(2)$, over which the vector space of languages has been defined, to richer fields like the set of real numbers.

Other areas of future research include applications of the language measure in anomaly detection, model identification, model order reduction, and analysis and synthesis of interfaces between the continuously-varying and discrete-event spaces in the language-measure setting.

Appendices A: Measure theory

This appendix introduces the concepts of standard measure-theoretic quantities that are used to establish the language measure in the main body of this paper.

Definition A.1: A σ-algebra M of a nonempty language $L(G_i) \subseteq \Sigma^*$ is a collection of subsets of $L(G_i)$ which satisfies the following three conditions.

(i) $L(G_i) \in M$;
(ii) If $K \in M$, then $(L(G_i) - K) \in M$;
(iii) $\cup_{j=1}^{\infty} K_j \in M$ if $K_j \in M\ \forall j$.

Definition A.2: An at most countable collection $\{L_k\}$ of members of a σ-algebra M is a partition of a member $L \in M$ if $L = \cup_k L_k$ and $L_k \cap L_j = \emptyset\ \forall k \neq j$.

Definition A.3: Let M be a σ-algebra of $L(G_i)$. Then, the set function $\mu: M \to \mathbf{R} \equiv (-\infty, +\infty)$, is called a signed real measure if the following two conditions are satisfied (Rudin 1987):

(i) $\mu(\emptyset) = 0$;
(ii) $\mu(\cup_{j=1}^{\infty} L_j) = \sum_{j=1}^{\infty} \mu(L_j)$ for every partition $\{L_j\}$ on any member $L \in M$.

Note that, unlike a positive measure (e.g., the Lebesgue measure), μ is finite such that the series in part (ii) of Definition A.3 converges absolutely in $\mathbf{R}$ and the result is independent of any permutation of the terms under union.

Definition A.4: Relative to the signed real measure μ, a sublanguage $L \in M$ is defined to be

(i) null, denoted as $L = 0$, if $\mu(L \cap J) = 0$, $\forall J \in M$;
(ii) positive, denoted as $L > 0$, if $L \neq 0$ and $\mu(L \cap J) \geq 0$, $\forall J \in M$;
(iii) negative, denoted as L, if $L \neq 0$ and $\mu(L \cap J) \leq 0$, $\forall J \in M$.

Definition A.5: Total variation $|\mu|$ on a σ-algebra M is defined as

$$|\mu|(L) = \sup \sum_k |\mu(L_k)| \tag{52}$$

$\forall L \in M$ where the supremum is taken over all partitions $\{L_k\}$ of L.

Proposition A.1: *Total variation measure $|\mu|$ of any regular language L is non-negative and finite i.e., $|\mu|(L) \in [0, \infty)$. The proof follows from standard theorems on complex measures* (*Rudin 1987*).

Total variation can be, in general, defined for complex measures (Rudin 1987) but it is restricted to a signed real measure in this paper. The total variation of a real signed measure μ, can be represented as $|\mu| = \mu^+ + \mu^-$ where μ^+ and μ^- are called positive and negative variations of μ and are defined as

$$\mu^+ = \tfrac{1}{2}(|\mu| + \mu) \qquad \text{and} \qquad \mu^- = \tfrac{1}{2}(|\mu| - \mu). \tag{53}$$

Both μ^+ and μ^- are positive measures on M. It also follows from the above equation that $\mu = \mu^+ - \mu^-$. This representation of μ as the difference of positive measure μ^+ and μ^- is known as the Jordan Decomposition of μ (Rudin 1987).

Proposition A.2: *Every sublanguage $L \in M$ can be partitioned as $L = L^0 \cup L^+ \cup L^-$ where the mutually exclusive sublanguages L^0, L^+ and L^- are called null, positive, and negative, respectively, relative to a signed real measure μ.*

Proof: The proof is based on the Hahn Decomposition Theorem (Rudin 1987). □

As a consequence of the above result, the following relations hold $\forall L \in M$ for positive and negative variations:

$$\mu^+(L) = \mu\left(L \bigcap L^+\right) \quad \text{and} \quad \mu^-(L) = -\mu\left(L \bigcap L^-\right). \tag{54}$$

Appendix B: Convergence of the language measure

This appendix establishes necessary and sufficient conditions for finiteness of the measure μ, based on certain properties of non-negative matrices, which are stated without proof. Details of these results are available in (Senata 1973, Plemmons and Berman 1979).

Definition B.1: Let A and B be real square matrices of the same order n. Then, the notations for inequalities are as follows:

$$\begin{aligned} A \geq B &\quad \text{if } a_{ij} \geq b_{ij}, \forall i,j \\ A > B &\quad \text{if } A \geq B, A \neq B \\ A \gg B &\quad \text{if } a_{ij} > b_{ij}, \forall i,j. \end{aligned}$$

If the matrix A satisfies the condition $A > \mathbf{0}$, i.e. the null matrix, then A is called a non-negative matrix and if the condition $A \gg 0$ is satisfied, then it is called a positive matrix.

Definition B.2: A square matrix A of order n is cogradient to a matrix E if $PAP^T = E$ for a permutation matrix P; and A is called reducible if A is cogradient to

$$E = \begin{bmatrix} B & \mathbf{0} \\ C & D \end{bmatrix},$$

where B and C are square matrices, or if $n = 1$ and $A = \mathbf{0}$. Otherwise, A is irreducible.

It follows from the above definition that a positive matrix is always irreducible.

Proposition B.1: *A non-negative matrix A is irreducible if and only if, for every (i,j) there exists a natural number k such that $a_{ij}^{(k)} > 0$, where $a_{ij}^{(k)}$ denotes the (i,j)th element of A^k.*

Proposition B.2: *If $A \geq \mathbf{0}$ is irreducible and $B \geq \mathbf{0}$, then $A + B$ is irreducible.*

Another characterization of irreducibility of a non-negative square matrix has a graph-theoretic interpretation. This relationship can help to determine under what conditions a given finite state automaton G, which represents the supervised or unsupervised plant model is irreducible by looking at connectivity of its states. The following definitions are needed to arrive at this conclusion.

Definition B.3: The associated directed graph, $G(A)$ of a square matrix A of order n, consists of n vertices $P_1, P_2, \ldots, P_n$ where an edge leads from P_i to P_j if and only if $a_{ij} \neq 0$.

Definition B.4: A directed graph G is strongly connected if for any ordered pair (P_i, P_j) of vertices of G, there exists a sequence of edges which leads from P_i to P_j.

Proposition B.3: *Given a matrix A, it is irreducible if and only if $G(A)$ is strongly connected.*

If A is a non-negative square matrix, then the following relationship holds between the spectral radius (i.e., maximum absolute eigenvalue) ρ of non-negative matrices.

Proposition B.4: *If $\mathbf{0} \leq A \leq B$ and $A + B$ is irreducible then $\rho(A) < \rho(B)$.*

Definition B.5: A square matrix S of order n is called (row) stochastic if it satisfies

$$s_{ij} \geq 0, \quad \sum_{j=1}^{n} s_{ij} = 1, \quad 1 \leq i \leq n. \tag{55}$$

Proposition B.5: *The maximum eigenvalue of a stochastic matrix S is one, i.e.* $\rho(S) = 1$. *A non-negative matrix A is stochastic if and only if e is an eigenvector of A corresponding to the eigenvalue one, where e is the vector all of whose entries are equal to one.*

In order to show that $(I - \mathbf{\Pi})^{-1}$ is invertible it suffices to show that $\rho(\mathbf{\Pi}) < 1$.

Theorem B.1: *If* $\rho(\mathbf{\Pi}) < 1$ *then there exists at least one* i, $1 \leq i \leq n$, *such that* $\sum_{j=1}^{n} \pi_{ij} < 1$.

Proof: Proof follows from the fact that if $\sum_{j=1}^{n} \pi_{ij} = 1$ $\forall i$, then $\mathbf{\Pi}$ would be a stochastic matrix by Definition B.5. Hence, by Proposition, B.5 $\rho(\mathbf{\Pi}) = 1 \Rightarrow (I - \mathbf{\Pi})^{-1}$ is not invertible. □

Theorem B.2: *If* $\sum_{j=1}^{n} \pi_{ij} < 1$ $\forall i$, $1 \leq i \leq n$, *then* $\rho(\mathbf{\Pi})$.

Proof: Let $\theta_i = (1 - \sum_{j=1}^{n} \pi_{ij})/n > 0$. Let S be a matrix of order n which is defined in the following manner

$$s_{ij} = \theta_i + \pi_{ij}, \quad \forall 1 \leq i, j \leq n.$$

It is clear that $S \gg \mathbf{0}$ and hence S is irreducible. Also S is a stochastic matrix and by Proposition B.5, the spectral radius $\rho(S) = 1$. Since $\mathbf{0} \leq \mathbf{\Pi} <$ and $\mathbf{\Pi} + S$ is irreducible by Proposition B.2, it follows that $\rho(\mathbf{\Pi}) < \rho(S) = 1$ from Proposition B.4. □

The above sufficiency condition is more strict than the necessary condition required in Theorem B.1. However, the necessary condition is not sufficient as seen from the following example.

$$\mathbf{\Pi} = \begin{pmatrix} 0.2 & 0 & 0.8 & 0 \\ 0 & 0.2 & 0.3 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0.1 & 0.2 & 0.4 & 0 \end{pmatrix}.$$

This matrix $\mathbf{\Pi}$ satisfies conditions as required in Theorem B.1, but $\rho(\mathbf{\Pi}) = 1$. It is possible to relax the strict inequality $\sum_{j=1}^{n} \pi_{ij} < 1$ $\forall i$, $1 \leq i \leq n$ in Theorem B.2, but with additional conditions on structure of $\mathbf{\Pi}$. For example, under such relaxed conditions, if $\mathbf{\Pi} + S$ is irreducible, then still $\rho(\mathbf{\Pi}) < 1$. This follows from the fact that application of Proposition B.4, only requires irreducibility of $\mathbf{\Pi} + S$. In order to determine the irreducibility of a matrix, the graph-theoretic interpretation, described earlier, can be a useful tool.

Acknowledgements

The author wishes to thank his former and current students and his colleagues for their contributions in the development of the language measure concept. The author specifically acknowledges the technical contributions of Mr. Amit Surana, Mr. Ishanu Chattopadhyay, Dr. Xi Wang, Dr. Jinbo Fu, and Professor Constantino Lagoa in the development of the language measure theory and its usage in discrete event supervisory control.

This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office (ARO) under Grant No. DAAD19-01-1-0646; and NASA Glenn Research Center under Grant No. NNC04GA49G.

References

C.G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Boston, MA: Kluwer Academic, 1999.

I. Chattopadhyay and A. Ray, "A language measure for partially observed discrete-event supervisory control systems", in *Proceedings of 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, Atlantis, December 2004, pp. 45–50.

Y.-C. Ho and X.-R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*, Boston, MA: Kluwer Academic, 1991.

J.E. Hopcroft, R. Motwani and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd edn, Boston, MA: Addison-Wesley, 2001.

R.E. Kalman, P.L. Falb and M.A. Arbib, *Topics in Mathematical System Theory*, Boston, MA: McGraw-Hill, 1969.

R. Kumar and V. Garg, *Modeling and Control of Logical Discrete Event Systems*, Boston, MA: Kluwer Academic, 1995.

A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, 2nd edn, New York, NY: McGraw-Hill International, 1991.

J.C. Martin, *Introduction to Languages and the Theory of Computation*, 2nd edn, Boston, MA: McGraw-Hill, 1997.

A.W. Naylor and G.R. Sell, *Linear Operator Theory in Engineering and Science*, New York, NY: Springer-Verlag, 1982.

S. Phoha, E. Peluso and R.L. Culver, "A high fidelity ocean sampling mobile network (samon) simulator", *IEEE Journal of Oceanic Engineering, Special Issue on Autonomous Ocean Sampling Networks*, 26, pp. 646–653, 2002.

R.J. Plemmons and A. Berman, *Nonnegative Matrices in the Mathematical Science*, New York: Academic Press, 1979.

P.J. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM J. Control and Optimization*, 25, pp. 206–230, 1987.

A. Ray, J. Fu and C. Lagoa, "Optimal supervisory control of finite state automata", *International Journal of Control*, 77, pp. 1083–1100, 2004.

A. Ray and S. Phoha, "Signed real measure of regular languages for discrete-event automata, *International Journal of Control*, 76, pp. 1800–1808, 2003.

W. Rudin, *Real and Complex Analysis*, 3rd edn, New York: McGraw-Hill, 1987.

E. Senata, *Non-negative Matrices*, New York: John Wiley, 1973.

X. Wang and A. Ray, "A language measure for performance evaluation of discrete-event supervisory control systems", *Applied Mathematical Modelling*, 28, pp. 817–833, 2004.

X. Wang, A. Ray and A. Khatkhate, "On-line identification of language measure parameters for discrete event supervisory control", *Applied Mathematical Modelling*, 29, pp. 597–613, 2005.

X. Wang, A. Ray, S. Phoha and J. Liu, "J-des: a graphical interactive package for analysis and synthesis of discrete event systems", in *Proceedings of American Control Conference*, Denver, Colorado, June 2003, pp. 3405–3410.

S. Yu, *Regular Languages, Handbook of Language Theory*, Chapter 2, Vol. 1, Berlin: Springer, 1997.