Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Mathematical Modelling 28 (2004) 817–833
APPLIED MATHEMATICAL MODELLING
www.elsevier.com/locate/apm

A language measure for performance evaluation of discrete-event supervisory control systems ☆

Xi Wang, Asok Ray *

Department of Mechanical Engineering, The Pennsylvania State University, 137 Reber Building, University Park, PA 16802-1412, USA

Received 25 October 2002; received in revised form 8 December 2003; accepted 16 December 2003
Available online 27 February 2004

Abstract

This paper formulates a signed real measure of sublanguages of a regular language based on the principles of automata theory and real analysis. The measure allows total ordering of any set of partially ordered sublanguages of a regular language for quantitative evaluation of the controlled behavior of a deterministic finite-state automaton (DFSA) plant under different supervisors. The computational complexity of the language measure algorithm is polynomial in the number of DFSA states.
© 2004 Elsevier Inc. All rights reserved.

Keywords: Discrete-event control; Formal languages; Performance measure

1. Introduction

The concept of discrete-event supervisory (DES) control, pioneered by Ramadge and Wonham [1] and subsequently enhanced by other researchers (for example, citations in [2]), provides a framework for achieving prescribed qualitative performance of physical plants. In this setting, the legal behavior of a physical plant is modelled by a deterministic finite-state automaton, abbreviated as DFSA, that is equivalent to a regular language [3–5]. A parallel composition of the unsupervised plant automaton and a supervisor automaton yields a sublanguage of the unsupervised plant language, which enables restricted legal behavior of the supervised plant [1].

☆ This work has been supported in part by Army Research Office (ARO) under Grant No. DAAD19-01-1-0646; and NASA Glenn Research Center under Grant Nos. NAG3-2448 and NNCO4GA49G.
* Corresponding author. Tel.: +1-814-865-6377; fax: +1-814-863-4848.
E-mail addresses: xxw117@psu.edu (X. Wang), axr2@psu.edu (A. Ray).

0307-904X/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2003.12.003

Traditionally, the concept of permissiveness has been used in DES control literature [2,6] to facilitate qualitative comparison of DES controllers under the language controllability condition. It has been shown that there exists a unique supremal controllable sublanguage that is maximally permissive and controllable with respect to the set of uncontrollable events associated with a DFSA plant model [7]. Design of maximally permissive DES controllers has been proposed by several researchers based on different conditions. However, maximal permissiveness does not necessarily imply best performance of the supervised plant from the perspectives of 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 motivation of this paper is to present 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 the Myhill–Nerode Theorem [4], where 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 the 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 few good strings as possible. Different supervisors may achieve this goal in different ways and generate a partially ordered set of controlled languages. The language measure then creates a total ordering on the performance of the controlled languages, which provides a precise quantitative comparison of the controlled plant behavior under different supervisors. This feature is formally stated as follows:

> Given that the relation $\subseteq$ induces a partial ordering on a set of controlled sublanguages $\{L(S_j/G), j = 1, \ldots, m\}$ of the plant language $L(G)$, the language measure μ induces a total ordering $\leqslant$ 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 originally formulated and solved by Wang and Ray [8] who introduced a quantitative measure of regular languages. The present paper enhances this concept of language measure by augmentation with additional key concepts and an engineering example. The major contribution of the present paper is mathematically rigorous formulation and systematic construction of a real signed measure of regular languages based on the fundamental principles of real analysis and automata theory. This language measure can quantify sublanguages of a regular language and is readily applicable to analysis and synthesis of discrete-event supervisory control algorithms. Specifically, performance indices of DES supervisors can be defined in terms of the language measure, regardless of how the supervisor is designed (e.g., maximally permissive or not; blocking or non-blocking; and completely or partially observed).

The paper is organized in seven sections. Section 2 formally introduces the concept of quantitative measure of regular languages representing DFSA models of unsupervised plants. Section 3

extends the language measure to those of controlled plant sublanguages under different supervisors. Section 4 addresses the issue of language measure computation. Section 5 discusses the usage of the language measure for quantitative analysis and synthesis of DES control systems. Section 6 presents an engineering example to demonstrate how the language measure has been used in the analysis and design of a DES controller for an aircraft gas turbine engine. The paper is summarized and concluded in Section 7 along with recommendations for future research.

2. The language measure

Let $G_i \equiv (Q, \Sigma, \delta, q_i, Q_m)$ be a DFSA model that represents the discrete-event dynamic behavior of a physical plant, where Q is the set of states with cardinality n, i.e., $|Q| = n$; the (finite) alphabet of events is denoted as Σ and its Kleene closure Σ^* is the set of all finite-length strings of events including the empty string ϵ; the (possibly partial) function $\delta : Q \times \Sigma \to Q$ represents the state transitions of the DFSA and δ^*: $Q \times \Sigma^* \to Q$ is an extension of δ; the state $q_i \in Q$ is the initial state; and $Q_m \subseteq Q$ is the set of marked (i.e., accepted) states.

Definition 2.1. The language $L(G_i)$ generated by a DFSA G_i initialized at the state $q_i \in Q$ is defined as

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

Definition 2.2. [9] Given a DFSA plant model G_i and two supervisors S_1, S_2, the (event disabling) mapping $L(G_i) \to 2^{\Sigma_c}$, where $\Sigma_c \subseteq \Sigma$ is the subset of controllable events. S_1 is said to be less permissive than S_2, denoted as $S_1 \preceq S_2$, if

$$\forall s \in L(G_i), \quad S_2(s) \subseteq S_1(s). \tag{2}$$

In other words, S_1 may disable more events than S_2 following the execution of any event string s. That is, $L(S_1/G_i) \subseteq L(S_2/G_i)$.

Definition 2.3. The language $L_m(G_i)$ marked by a DFSA G_i initialized at the state $q_i \in Q$ is defined as

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

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 $Q_m^+ \cap Q_m^- = \emptyset$, where Q_m^+ contains all good marked states at which we desire to terminate, and Q_m^- contains all bad marked states that we want to avoid terminating on, although it may not always be possible to completely avoid the bad states while attempting to reach the good states. In general, the marked language $L_m(G_i)$ consists of both good and bad event strings that, starting from the initial state q_i, respectively lead to Q_m^+ and Q_m^-. Any event string belonging to the language $L^0 = L(G_i) - L_m(G_i)$ leads to one of the non-marked states belonging to $Q - Q_m$ and L^0 does not contain any one of the good or bad strings. Ray and Phoha [10] and Surana and Ray [11] have provided a detailed explanation on partitioning of the language into positive, negative, and zero measures following the Hahn Decomposition Theorem [12].

Definition 2.4. For every $q_k \in Q$, let $L(q_i, q_k)$ denote the set of all strings that, starting from the state q_i, terminate at the state q_k, i.e.,

$$L(q_i, q_k) = \{s \in \Sigma^* | \delta^*(q_i, s) = q_k \in Q\}. \tag{4}$$

Based on the equivalence classes defined in the Myhill–Nerode Theorem [4], the regular languages $L(G_i)$ and $L_{\mathrm{m}}(G_i)$ can be expressed as

$$L(G_i) = \bigcup_{q_k \in Q} L(q_i, q_k) = \bigcup_{k=1}^{n} L(q_i, q_k), \tag{5}$$

$$L_{\mathrm{m}}(G_i) = \bigcup_{q_k \in Q_{\mathrm{m}}} L(q_i, q_k) = L_{\mathrm{m}}^{+} \cup L_{\mathrm{m}}^{-}, \tag{6}$$

where the sublanguage $L(q_i, q_k) \subseteq G_i$ having the initial state q_i is uniquely labeled by the terminal state q_k, $k \in \mathscr{I} \equiv \{1, \ldots, n\}$ that is the index of Q. Then, $L(q_i, q_j) \cap L(q_i, q_k) = \emptyset \ \forall j \neq k$; and $L_{\mathrm{m}}^{+} \equiv \bigcup_{q \in Q_{\mathrm{m}}^{+}} L(q_i, q)$ and $L_{\mathrm{m}}^{-} \equiv \bigcup_{q \in Q_{\mathrm{m}}^{-}} L(q_i, q)$ are good and bad sublanguages of $L_{\mathrm{m}}(G_i)$, respectively. Consequently, $L(G_i) = L^0 \cup L_{\mathrm{m}}^{+} \cup L_{\mathrm{m}}^{-}$, where $L^0 \equiv \bigcup_{q \notin Q_{\mathrm{m}}} L(q_i, q)$.

Definition 2.5. Let Θ be a σ-algebra of $L(G_i)$. Then, the set function $\mu : \Theta \to \mathbf{R} \equiv (-\infty, +\infty)$, is called a signed real measure if the following two conditions are satisfied [12]:

(1) $\mu(\emptyset) = 0$;
(2) $\mu(\cup_{j=1}^{\infty} L_j) = \sum_{j=1}^{\infty} \mu(L_j)$ $\forall L_j \in \Theta$ and $L_j \cap L_k = \emptyset$ if $j \neq k$.

We select the power set $2^{L(G_i)}$ as the σ-algebra of $L(G_i)$. Consequently, a signed real measure $\mu : 2^{L(G_i)} \to \mathbf{R}$ for the DFSA, whose (regular) language is $L(G_i)$, is constructed as follows:

$$\forall q \in Q, \quad \mu(L(q_i, q)) \begin{cases} = 0, & q \notin Q_{\mathrm{m}}, \\ > 0, & q \in Q_{\mathrm{m}}^{+}, \\ < 0, & q \in Q_{\mathrm{m}}^{-}. \end{cases} \tag{7}$$

To achieve the above goal of signed measure, we characterize the marked states such that each state in Q_{m}^{+} is assigned a positive weight and each state in Q_{m}^{-} a negative weight; and each unmarked state is assigned the zero weight. The weights are chosen by the designer based on the perception of each marked state's role in the system performance.

Definition 2.6. The characteristic function $\chi : Q \to [-1, 1]$ that assigns a signed real weight to state-based sublanguages $L(q_i, q)$ is defined as

$$\forall q \in Q, \quad \chi(q) \in \begin{cases} [-1, 0), & q \in Q_{\mathrm{m}}^{-}, \\ \{0\}, & q \notin Q_{\mathrm{m}}, \\ (0, 1], & q \in Q_{\mathrm{m}}^{+}. \end{cases} \tag{8}$$

The state weighting vector, denoted by $\mathbf{X} = [\chi_1 \quad \chi_2 \quad \cdots \quad \chi_n]^{\mathrm{T}}$, where $\chi_k \equiv \chi(q_k)$ $\forall k \in \mathscr{I}$, is called the **X**-vector. The kth element χ_k of **X**-vector is the weight assigned to the corresponding terminal state q_k.

To compute the measure of the language $L(q, q_i)$, we assign a cost to each string terminating at the state q starting from the initial state q_i. To this end, the event cost is defined conceptually similar to the conditional transition probability, assuming that the DFSA model is Markov.

Definition 2.7. 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 $\forall q_i \in Q$, $\forall \sigma_j \in \Sigma$, $\forall s \in \Sigma^*$,

(1) $\tilde{\pi}[\sigma_j, q_i] \equiv \tilde{\pi}_{ij} \in [0,1)$; $\sum_j \tilde{\pi}_{ij} < 1$;
(2) $\tilde{\pi}[\sigma_j, q_i] = 0$ if $\delta(q_i, \sigma_j)$ is undefined; $\tilde{\pi}[\epsilon, q_i] = 1$;
(3) $\tilde{\pi}[\sigma_j s, q_i] = \tilde{\pi}[\sigma_j, q_i]\tilde{\pi}[s, \delta(q_i, \sigma_j)]$.

Now we introduce the language measure in terms of the event cost function $\tilde{\pi}$ and the characteristic function χ.

Definition 2.8. The signed real measure of every singleton string set $\{s\} \in 2^{L(G_i)}$, where $s \in L(q_i, q)$, is defined as $\mu(\{s\}) \equiv \tilde{\pi}[s, q_i]\chi(q)$ implying that

$$\forall s \in L(q_i, q), \quad \mu(\{s\}) \begin{cases} = 0, & q \notin Q_{\mathrm{m}}, \\ > 0, & q \in Q_{\mathrm{m}}^+, \\ < 0, & q \in Q_{\mathrm{m}}^-. \end{cases} \tag{9}$$

It follows from Definition 2.8 that the signed measure of the sublanguage $L(q_i, q) \subseteq L(G_i)$, of all events starting at q_i and terminating at q is

$$\mu(L(q_i, q)) = \left(\sum_{s \in L(q_i, q)} \tilde{\pi}[s, q_i] \right) \chi(q). \tag{10}$$

Definition 2.9. The signed real measure of the language of a DFSA G_i initialized at a state $q_i \in Q$, is defined as

$$\mu_i \equiv \mu(L(G_i)) = \sum_{q \in Q} \mu(L(q_i, q)). \tag{11}$$

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

Since $\mu(L(q_i, q)) = 0 \ \forall q \notin Q_{\mathrm{m}}$ by Definition 2.8, it follows from Definition 2.9 that μ_i is the signed measure of the marked language $L_{\mathrm{m}}(G_i)$ of the DFSA G_i, i.e., $\mu_i = \mu(L_{\mathrm{m}}(G_i))$. Therefore $(L(G_i), 2^{L(G_i)}, \mu_i)$ forms a measure space. In principle, any measure can be defined on the measurable space $(L(G_i), 2^{L(G_i)})$. The choice of the signed language measure as given by Definitions 2.9 has been motivated by the physical significance of marked states in terms of DES system operations and control objectives.

The marked language $L_{\mathrm{m}}(G_i)$ of the unsupervised plant consists of good strings terminating on Q_{m}^+ and bad strings terminating on Q_{m}^-. A supervisor S may disable some of the bad strings and retain some of the good string enabled, depending on its ability to access controllable events. Different supervisors $S_j : j \in \{1, 2, \ldots, n_s\}$ for a DFSA G_i may achieve this goal in different ways and generate a partially ordered set of controlled sublanguages $\{L_{\mathrm{m}}(S_j/G_i) : j \in \{1, 2, \ldots, n_s\}\}$. The real signed measure μ provides a precise quantitative comparison of the controlled plant behavior under different supervisors because the set $\{\mu(L_{\mathrm{m}}(S_j/G_i)) : j \in \{1, 2, \ldots, n_s\}\}$ is totally ordered. In essence, the range of the language measure μ is totally ordered while its domain $2^{L(G_i)}$ is partially ordered. Section 4 shows how the language measure is computed.

3. Measure of supervised plant sublanguages

The previous section formulated a quantitative measure for a regular language, equivalently, a deterministic finite-state automaton (DFSA) model of an unsupervised plant. Given the $\mathbf{\Pi}$-matrix and $\mathbf{X}$-vector, the corresponding language measure is evaluated following Definition 2.9. This section addresses an important issue: how to evaluate the measure of different supervisors on a common quantitative basis. It is noted that all supervisors are designed with respect to the same DFSA plant model G. Let $L(S)$ be the generating language of a supervisor S. The language $L(S/G)$ of the supervised plant is formally defined by the parallel composition of G and S as follows.

Definition 3.1. Let $G \equiv (Q_1, \Sigma, \delta_1, q_{11}, Q_{\mathrm{m1}})$ and $S \equiv (Q_2, \Sigma, \delta_2, q_{21}, Q_{\mathrm{m2}})$ be the DFSAs that generate the plant language and the supervisor's specification language, respectively. The supervised plant behavior is represented by a DFSA $S/G = (Q, \Sigma, \delta, q_1, Q_{\mathrm{m}})$, where $Q \equiv Q_1 \times Q_2$; $Q_{\mathrm{m}} \equiv Q_{\mathrm{m1}} \times Q_{\mathrm{m2}}$; $q_j \equiv \langle q_{1j}, q_{2j} \rangle$; and for $q \in Q$ and $\sigma \in \Sigma$, the state transition is defined as

$$\delta(q, \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), \tag{12}$$

implying that $L(S/G) = L(G) \cap L(S)$ and $L_{\mathrm{m}}(S/G) = L_{\mathrm{m}}(G) \cap L_{\mathrm{m}}(S)$. Specifically, $S/G \equiv S$ if $L(S) \subseteq L(G)$.

It follows from the construction of the language measure in Section 2 that $L(G) = \cup_{k=1}^{n_1} L(q_{11}, q_{1k})$, where $n_1 = |Q_1|$ and $q_{1k} \in Q_1$, $1 \leqslant k \leqslant n_1$. Following Definition 3.1, each sublanguage $L(q_{11}, q_{1k}) \subseteq L(G)$ can be further partitioned as $L(q_{11}, q_{1k}) \cap L(q_{21}, q_{2j})$, $1 \leqslant k \leqslant n_1$, $1 \leqslant j \leqslant n_2$ where $n_2 = |Q_2|$. Therefore, the set of strings retained in $L_{\mathrm{m}}(G) \cap L_{\mathrm{m}}(S)$, corresponding to every $q_{1k} \in Q_{\mathrm{m1}}$, can be expressed as

$$L(q_{11}, q_{1k}) \bigcap \left(\bigcup_{q_{2j} \in Q_{\mathrm{m2}}} L(q_{21}, q_{2j}) \right), \tag{13}$$

that, in turn, is used for refining the partitioning of the unsupervised language to obtain the parameters of the supervised sublanguage. The objective is to measure the performance of a supervisor in terms of the language measure of the supervised plant language. In this context, the role of a well-designed supervisor is delineated below.

The supervised plant sublanguage should attempt to retain (as many as possible) strings that terminate on a good state (i.e., $q \in Q_{\mathrm{m1}}^{+}$) and discard (as many as possible) strings that terminate on a bad state (i.e., $q \in Q_{\mathrm{m1}}^{-}$). Therefore, a more positive measure of the supervised sublanguage is deemed to yield a better performance.

The above interpretation on refinement of equivalence classes in the supervised sublanguage (i.e., the closed-loop system) shows how the parameters (i.e., event cost matrix $\widetilde{\mathbf{\Pi}}$ and the characteristic vector $\mathbf{X}$) of the unsupervised plant can be used to generate the respective parameters for different (regular) sublanguages of supervised plants.

Definition 3.2. Let G, S and S/G be as in Definition 3.1. Let $\widetilde{\mathbf{\Pi}}_P$-matrix be the event cost matrix and $\mathbf{X}_P$-vector be the characteristic vector for the unsupervised plant G. The event cost matrix $\widetilde{\mathbf{\Pi}}_{S/G}$ and characteristic vector $\mathbf{X}_{S/G}$ of the supervised DFSA S/G are defined as

$$\tilde{\pi}_{S/G}[\sigma, \langle q_{1i}, q_{2j}\rangle] = \tilde{\pi}_P[\sigma, q_{1i}] \tag{14}$$

$\forall \tilde{\pi}_{S/G} \in \widetilde{\mathbf{\Pi}}_{S/G}\ \forall \sigma \in \Sigma$ and $1 \leqslant i \leqslant n_1,\ 1 \leqslant j \leqslant n_2$.

$$\chi_{S/G}(\langle q_{1i}, q_{2j}\rangle) = \chi_P(q_{1i}) I(q_{2j}), \tag{15}$$

where $I(\cdot)$ is the indicator function defined as

$$I(q) = \begin{cases} 1, & q \in Q_{\mathrm{m}2}, \\ 0, & q \notin Q_{\mathrm{m}2}. \end{cases} \tag{16}$$

Let $s \in L(q_1, q_{ij})$ where $q_{ij} \equiv \langle q_{1i}, q_{2j}\rangle$. By Definition 2.9, the measure of the singleton set $\{s\}$ in the unsupervised plant language $L(G)$ is $\mu_P(\{s\}) = \tilde{\pi}_P[s, q_{11}]\chi_P(q_{1i})$. By Definition 3.2, the measure of $\{s\}$ in the supervised plant S/G is given by

$$\mu_C(\{s\}) = \tilde{\pi}_C[s, q_1]\chi_C(q_{ij}) = \tilde{\pi}_P[s, q_{11}]\chi_P(q_{1i}) I(q_{2j}).$$

Therefore, if no controllable event in the string s is disabled by the supervisor, then $\mu_C(\{s\}) = \mu_P(\{s\})$; otherwise $\mu_{S/G}(s) = 0$. Definition 2.9 guarantees that identical strings in different supervised plant sublanguages $L(S/G)$ are assigned an identical measure to provide a common base for quantitative evaluation of different supervisors' performance. Section 6 illustrates performance comparison of three different supervisors.

4. Language measure computation

Various methods of obtaining regular expressions for DFSAs are reported in Hopcroft et al. [4], Drobot [3], and Wonham [7]. 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 values 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. A procedure for language measure computation in closed form [8] is presented below.

Definition 4.1. Let $L_i \equiv Ł_{\mathrm{m}}(G_i)$, $i \in \mathscr{I} = \{1, \ldots, n\}$, denote the regular expression representing the marked language of an n-state DFSA $G_i \equiv (Q, \Sigma, \delta, q_i, Q_{\mathrm{m}})$ where q_i is the initial state.

Definition 4.2. 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 $j, k \in \mathscr{I}$, i.e., $\delta(q_j, \sigma) = q_k$, $\forall \sigma \in \sigma_j^k \subseteq \Sigma$.

Lemma 4.1. *Let u, v be two known regular expressions and r be an unknown regular expression that satisfies the following algebraic identity*:

$$r = ur + v. \tag{17}$$

Then, the following relations are true:

(1) $r = u^*v$ *is a solution to Eq.* (17).
(2) *If* $\epsilon \notin u$, *then* $r = u^*v$ *is the unique solution to Eq.* (17).

The proof of Lemma 4.1 can be found in [3]. Part (2) of it is also known as Arden's rule [7].

Example 4.1. Let $\Sigma = \{a, b\}$; $Q = \{1, 2, 3\}$; the initial state is 1, and the sole marked state is 2 in Fig. 1. Let the set of linear algebraic equations represent the transitions at each state of the DFSA.

$$\begin{cases} L_1 = a_1^1 L_1 + b_1^2 L_2, \\ L_2 = a_2^1 L_1 + b_2^3 L_3 + \epsilon, \\ L_3 = a_3^1 L_1 + b_3^2 L_2, \end{cases} \tag{18}$$

where the 'forcing' term ϵ is introduced on the right side of the ith equation whenever $q_i \in Q_m$, $i \in \mathscr{I}$. By application of Lemma 4.1, the regular expression for the marked language $L_m(G_1)$ is

$$L_m(G_1) \equiv L_1 = (a_1^1)^* b_1^2 (a_2^1 (a_1^1)^* b_1^2 + b_2^3 a_3^1 (a_1^1)^* b_1^2 + b_2^3 b_3^2)^*.$$

The method of system description in Example 4.1 can be extended to the general case without any difficulty. Given a DFSA $G_i = (Q, \Sigma, \delta, q_i, Q_m)$ with $|Q| = n$, we proceed to obtain the system equation by a set of regular expressions L_i of the marked language $L_m(G_i)$, $i \in \mathscr{I}$, as follows:

$$\forall q_i \in Q, \quad L_i = \sum_j R_{i,j} + \mathscr{E}_i, \quad i \in \mathscr{I}, \tag{19}$$

where $\forall i$, $R_{i,j}$ is defined as

1. If $\exists \sigma \in \Sigma$, such that $\delta(q_i, \sigma) = q_j \in Q$, $j \in \mathscr{I}$, then $R_{i,j} = \sigma_i^j L_j$, otherwise, $R_{i,j} = \emptyset$.
2. If $q_i \in Q_m$, $\mathscr{E}_i = \epsilon$, otherwise, $\mathscr{E}_i = \emptyset$.

The set of symbolic equations may be written as

$$L_i = \sum_j \sigma_i^j L_j + \mathscr{E}_i. \tag{20}$$

We note the following special cases.

(1) If $\mathscr{E}_i = \emptyset$, $\forall L_i$, then $L_m(G) = \emptyset$. This implies that the DFSA has no marked state.
(2) If $\exists q_i \in Q$ such that $L_i = \epsilon$, then q_i is marked. Furthermore, q_i is a deadlock state.

In order to convert the symbolic equations (20) into a set of algebraic equations, we introduce the (one-hop) state transition cost that is defined below.

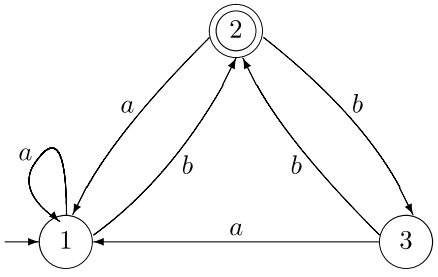

Fig. 1. Example 1.

Definition 4.3. The state transition cost of the DFSA G_i is defined as a function $\pi : Q \times Q \to [0, 1)$ such that $\forall q_i, q_j \in Q$, $\pi[q_j|q_i] = \sum_{\sigma \in \Sigma : \delta(q_i,\sigma)=q_j} \tilde{\pi}[\sigma, q_i] \equiv \pi_{ij}$ and $\pi_{ij} = 0$ if $\{\sigma \in \Sigma : \delta(q_i, \sigma) = q_j\} = \emptyset$. The $n \times n$ state transition cost matrix is defined as

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

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

Now we present an alternative form of language measure (Definition 2.9) in terms of the state transition cost (Definition 4.3) instead of event cost (Definitions 2.7) as delineated below.

Theorem 4.1. *The language measure of the symbolic equations* (20) *is given by*

$$\mu_i = \sum_j \pi_{ij}\mu_j + \chi_i. \tag{21}$$

Proof. Following Eq. (19) and Definition 2.6:

$$\forall i \in \mathscr{I}, \quad \mu(\mathscr{E}_i) = \begin{cases} \chi_i & \text{if } \mathscr{E}_i = \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Therefore, $\mathbf{X} = [\,\chi_1 \quad \chi_2 \quad \cdots \quad \chi_n\,]^{\mathrm{T}}$ is the forcing vector in Eq. (20). Starting from the state q_i, the measure

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

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

$$\forall j \neq k, \quad \sigma_i^j L_j \cap \sigma_i^k L_k = \emptyset \tag{23}$$

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 for some $k \neq j$. This justifies the fourth equality. Since the DFSA model is modelled to be Markov, $\mu(\sigma_i^j L_j) = \mu(\sigma_i^j)\mu(L_j)$. Therefore, by Definitions 2.7 and 4.2, $\mu(\sigma_i^j L_j) = \pi[q_j|q_i]\mu(L_j) = \pi_{ij}\mu(L_j)$. □

In vector notation, Eq. (21) in Theorem 4.1 is expressed as $\boldsymbol{\mu} = \boldsymbol{\Pi}\boldsymbol{\mu} + \mathbf{X}$ whose solution is given by

$$\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Pi})^{-1}\mathbf{X} \tag{24}$$

provided that the matrix $\mathbf{I} - \boldsymbol{\Pi}$ is non-singular. Definitions 2.7 and 4.3 state that each element in the $\boldsymbol{\Pi}$-matrix is non-negative and each row sum is less than 1. These conditions make the $\boldsymbol{\Pi}$-matrix a contraction operator that is sufficient for the matrix $(\mathbf{I} - \boldsymbol{\Pi})^{-1}$ to be a bounded linear operator [13]. Therefore, Definitions 2.7 and 4.3 provide a sufficient condition for the language measure μ of the DFSA G to be finite.

The jth element of the ith row of the $(\mathbf{I} - \boldsymbol{\Pi})^{-1}$ matrix, denoted as v_i^j, is the language measure of a DFSA with the same state transition function δ as G 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 i.e., $\chi_l = 1$ if $l = j$ and $\chi_l = 0$ if $l \neq j$. In that case, $\mu_i = \mu(L_{\mathrm{m}}(G_i))$ becomes $\mu_i = \sum_j v_i^j \chi_j$. Numerical evaluation of the language measure of G_i requires Gaussian elimination of the single variable μ_i involving the real square matrix $(\mathbf{I} - \boldsymbol{\Pi})$. As such the computational complexity of the language measure algorithm is polynomial in the number of plant states.

In general the language measure of G_i does not require computation of μ_j, $j \neq i$. However, on-line supervisory control may require the information on the performance of the automaton starting from an arbitrary state q_i, $i \neq 1$. In that case, $\mu_i = \sum_j v_i^j \chi_j$, $i \neq 1$ should be computed.

Example 4.2 (Example 4.1 revisited). Let us assign the $\boldsymbol{\Pi}$-matrix and $\mathbf{X}$-vector in Example 4.1 as follows:

$$\boldsymbol{\Pi} = \begin{pmatrix} 0.3 & 0.4 & 0 \\ 0.2 & 0 & 0.6 \\ 0.5 & 0.4 & 0 \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

then $\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Pi})^{-1}\mathbf{X} = [\,1.2048 \quad 2.1084 \quad 1.4458\,]^{\mathrm{T}}$, therefore, $\mu_1 = 1.2048$, $\mu_2 = 2.1084$, and $\mu_3 = 1.4458$.

5. Usage of the language measure for supervisor synthesis

The language measure, introduced in the previous sections, serves as a common quantitative tool to compare the performance of different supervisors and is assigned a state characteristic $\mathbf{X}$-vector and an event cost $\tilde{\boldsymbol{\Pi}}$-matrix. The $\mathbf{X}$-vector is chosen based on the designer's perception of the individual state's impact on the system performance. On the other hand, event costs

(i.e., elements of the $\tilde{\mathbf{\Pi}}$-matrix) are dependent on the plant states, where the respective events are generated and are conceptually similar to their transition probabilities that have been modelled in the setting of a stochastic finite-state automaton (SFA) in diverse applications including the speech recognition problem [1,14,15]. While the main objective in the speech recognition problem is to identify the minimal SFA for a stochastic regular language, the event cost $\tilde{\mathbf{\Pi}}$-matrix needs to be identified as language measure parameters for the given DFSA plant model with known structure. The $\tilde{\mathbf{\Pi}}$-matrix and the $\mathbf{X}$-vector are critical for (quantitative) performance comparison of different supervisory controllers.

Several researchers have proposed optimal control of DFSA based on different assumptions. Some of these researchers have attempted to quantify the controller performance using different types of cost assigned to the individual events. Kumar and Garg [16] introduced the concept of payoff and control costs that are incurred only once regardless of the number of times the system visits the state associated with the cost; consequently, the resulting cost is not a function of the dynamic behavior of the plant. Sengupta and Lafortune [17] used control cost in addition to the path cost in optimization of the performance index for trade-off between finding the shortest path and reducing the control cost. Although costs were assigned to the events, no distinction was made for events generated at (or leading to) different states that could be "good" or "bad". These optimal control strategies have addressed performance enhancement of discrete-event control systems without a quantitative measure of languages. The (signed) language measure μ may serve as an index for synthesis of an optimal control policy that maximizes the performance of a controlled sublanguage. The salient concept is succinctly presented below.

Let $\mathscr{S} \equiv \{S^0, S^1, \ldots, S^N\}$ be a set of supervisory control policies for the open loop 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 open loop 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: $\mathbf{\Pi}^k \equiv \mathbf{\Pi}(S^k) \leqslant \mathbf{\Pi}^0$ and $L(S^k/G) \subseteq L(G), \forall S^k \in \mathscr{S}$. The task is to synthesize an optimal cost matrix $\mathbf{\Pi}^* \leqslant \mathbf{\Pi}^0$ that maximizes the performance vector $\mu^* \equiv [\mathbf{I} - \mathbf{\Pi}^*]^{-1}\mathbf{X}$, i.e., $\mu^* \geqslant \mu^k \equiv [\mathbf{I} - \mathbf{\Pi}^k]^{-1}\mathbf{X} \ \forall \ \mathbf{\Pi}^k \leqslant \mathbf{\Pi}^0$ where the inequalities are implied elementwise. The research work in this direction is in progress and the results on robust and optimal DES control have been reported in recent publications [18–20].

6. An example of engineering application

This section presents an engineering example to illustrate the role of the language measure in the design of supervisory control systems for a given plant. A family of supervisors, based on different control specifications, are designed for a twin-engine unmanned aircraft that is used for surveillance and data collection. The measures of the uncontrolled plant language and the controlled sublanguages are compared to quantitatively evaluate the performance of these supervisors.

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); and 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 Fig. 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 Tables 1 and 2, respectively. The state transition function δ and the state-based event cost $\tilde{\pi}_{ij}$ (see Definition 2.7) are entered simultaneously in Table 3. The fraction part in each entry denotes the corresponding state-based event cost $\tilde{\pi}_{ij}$ such that each row sum of the event cost matrix $\tilde{\Pi}$ is strictly less than one. The integer part (within parentheses) in each entry denotes the respective destination state resulting from the occurrence of the event. The values of $\tilde{\pi}_{ij}$ were selected by extensive experiments on engine simulation models and were also based on experience of gas 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 2.6) were chosen as signed real weights based on the perception of each marked state's role on the engine performance.

The characteristic values of the 13 states in Table 1 is given by the characteristic vector $\mathbf{X} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -0.05\ 0.25\ -1.0]^{\mathrm{T}}$. 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 1–10 are not marked, the first 10 elements of the characteristic vector $\mathbf{X}$ are zeros. The implication is that event strings terminating at states 1–10 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.

Three supervisory controllers were designed independently using a graphical interactive package based on the following control 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.

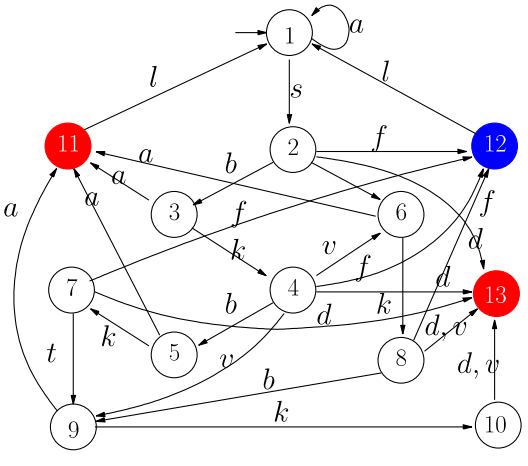

Fig. 2. Finite-state automaton model of the plant.

Table 1
Plant automaton 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	Σ_c
s	Start and take-off	$\surd$
b	A good engine becoming unhealthy	
t	An unhealthy engine becoming inoperable	
v	A good engine becoming inoperable	
k	Keep engine(s) running	$\surd$
a	Mission abortion	$\surd$
f	Mission completion	
d	Destroyed aircraft	
l	Landing	$\surd$

Figs. 3–5 show the finite-state machine diagrams of the supervised plant under control specifications 1–3, respectively. The dashed lines in these figures indicate that the transitions under corresponding supervisory controllers have been deleted from the plant model as a result of disabled (controllable) events. The performance measure μ_1 (i.e., with the initial state 1) of the uncontrolled plant is 0.0823 and for three supervised plants under specifications #1, #2, and #3 were evaluated to be: 0.0807, 0.0822, and 0.0840, respectively. Therefore, the performance of the supervised plant under specifications #1, #2, and #3 is inferior, similar, and superior to that of the unsupervised plant from perspectives of the mission objectives as described by the language measure parameters $\tilde{\pi}$ and χ. The supervisor #3 yields the best performance among the three supervisors based on language measure parameters in Table 3 and the characteristic vector $\mathbf{X}$.

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 (3)	
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									

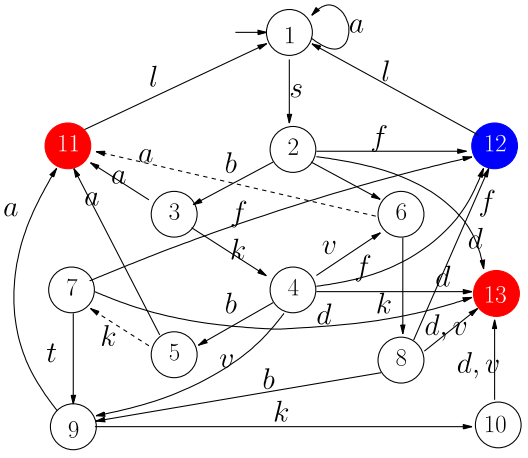

Fig. 3. Controller 1 for Specification #1.

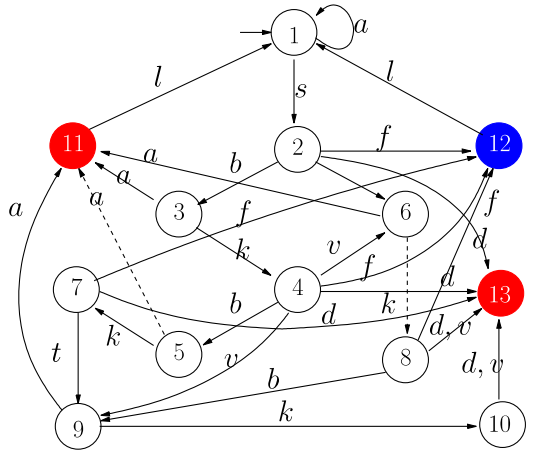

Fig. 4. Controller 2 for Specification #2.

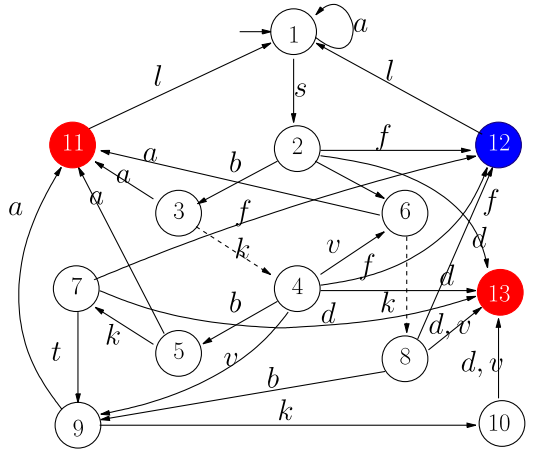

Fig. 5. Controller 3 for Specification #3.

7. Summary and conclusions

This paper presents the concept, formulation and validation of a signed real measure for regular languages and their sublanguages. The objective is to control the legal behavior of physical plants that can often be modelled by finite-state automata. Specifically, the relative performance of supervisors can be quantitatively evaluated in terms of the measure of the controlled 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 leading to bad (good) marked states. As such the measure of a (properly designed) controlled sublanguage should be greater than that of the (unsupervised) plant language.

Cost assignment to each event based on the state, where it is generated, is conceptually similar to the conditional probability of the event. The procedure of controller evaluation in terms of its language measure is validated by an example of a gas turbine engine for three different supervisors. A relatively less permissive supervisor could be more effective than another supervisor that may not adequately delete event strings leading to bad marked states.

The computational complexity of the language measure algorithm is polynomial in the number of states. Potential applications of the language measure are model identification, model order reduction, and analysis and synthesis of robust and optimal control and diagnostic systems in the discrete-event setting. Further research is recommended for development of systematic procedures for assigning/identifying the event cost matrix and the characteristic vector. It would be challenging to extend the concept of (regular) language measure for languages higher up in the Chomsky Hierarchy [5] 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.

Acknowledgements

The authors acknowledge the benefits of discussions with Mr. Amit Surana and Dr. Jinbo Fu.

References

[1] P.J. Ramadge, W.M. Wonham, Supervisory control of a class of discrete event processes, SIAM J. Control Optim. 25 (1) (1987) 206–230.
[2] C.G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Kluwer Academic, Dordrecht, 1999.
[3] V. Drobot, Formal Languages and Automata Theory, Computer Science Press, Rockville, MD, 1989.
[4] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, second ed., Addison-Wesley, Reading, MA, 2001.
[5] J.C. Martin, Introduction to Languages and the Theory of Computation, second ed., McGraw-Hill, New York, 2001.
[6] R. Kumar, V. Garg, Modeling and Control Logical Discrete Event Systems, Kluwer Academic, Dordrecht, 1995.
[7] W.M. Wonham, Discrete Event System Control Theory and Application, University of Toronto, 2001.
[8] X. Wang, A. Ray, Signed real measure of regular languages, in: Proceedings of the American Control Conference 2002, Anchorage, Alaska, vol. 5, 2002, pp. 3937–3942.
[9] R. Kumar, V.K. Garg, Control of stochastic discrete event systems modeled by probabilistic languages, IEEE Trans. Automat. Contr. 46 (4) (2001) 593–606.
[10] A. Ray, S. Phoha, Signed real measure of regular languages for discrete-event automata, Int. J. Control. 76 (2003) 1800–1808.
[11] A. Surana, A. Ray, Signed real measure of regular languages, in: IEEE Conference on Decision and Control, Maui, Hawaii, 2003.
[12] W. Rudin, Real and Complex Analysis, third ed., McGraw-Hill, 1987.
[13] A.W. Naylor, G.R. Sell, Linear Operator Theory in Engineering and Science, Springer-Verlag, New York, 1982.
[14] R. Carrasco, J. Oncina, Learning deterministic regular grammars from stochastic samples in polynomial time, Theor. Informat. Appl. 33 (1) (1999) 1–19.
[15] F. Thollard, P. Dupont, C. de la Higuera, Probabilistic DFA inference using Kullback–Leibler divergence and minimality, in: Proceedings of 17th International Conference on Machine Learning, Stanford University, Stanford, CA, 2000, pp. 975–982.

[16] R. Kumar, V.K. Garg, Optimal supervisory control of discrete event dynamical systems, SIAM J. Control Optim. 33 (2) (1995) 419–439.
[17] R. Sengupta, S. Lafortune, An optimal control theory for discrete event systems, SIAM J. Control Optim. 36 (2) (1998) 488–541.
[18] J. Fu, C.M. Lagoa, A. Ray, Robust optimal control of regular languages with event cost uncertainties, in: IEEE Conference on Decision and Control, Maui, Hawaii, 2003, pp. 3209–3214.
[19] J. Fu, A. Ray, C.M. Lagoa, Unconstrained optimal control of regular languages, in: IEEE Conference on Decision and Control, Las Vegas, Nevada, 2002, pp. 799–804.
[20] J. Fu, A. Ray, C.M. Lagoa, Optimal control of regular languages with event disabling cost, in: Preprints American Control Conference, Denver, Colorado, 2003, pp. 1691–1695.