A Complex Measure of Non-Regular Languages for Discrete-Event Supervisory Control

Ishanu Chattopadhyay ixc128@psu.edu
Asok Ray axr2@psu.edu
Xi Wang xxw117@psu.edu
The Pennsylvania State University
University Park, PA 16802

Keywords: *Language Measure; Discrete Event Supervisory Control; Linear Grammars*

Abstract— **The measure of regular languages, recently introduced in technical literature, has been the driving force for quantitative analysis and synthesis of discrete-event supervisory (DES) control systems dealing with finite state automata (equivalently, regular languages). This paper extends the signed real measure of regular languages, to a complex measure of non-regular languages, generated by *linear context free grammars*; the concept is illustrated by an example. The complex measure becomes equivalent to the signed real measure if the linear context free grammar is degenerated to a regular grammar.**

I. Introduction

Finite state automata (FSA) (equivalently, regular languages) have been widely used to model and synthesize supervisory control laws for discrete-event plants [6] [1]. The mathematical simplicity of regular languages makes the control synthesis computationally efficient. According to the paradigm of supervisory control, a discrete event system (e.g., the model of a physical plant) is a language generator whose behavior is constrained by a supervisor to meet a given specification. The (controlled) sublanguage of the plant behavior could be different under different supervisors that satisfy their own respective specifications. Such a partially ordered set of sublanguages requires a quantitative measure for total ordering of their respective performance. To address this issue, a signed real measure of regular languages has been reported in literature [8] [7] to provide a mathematical framework for quantitative comparison of regular languages. This measure formalizes a procedure for design of discrete event supervisory (DES) controllers for finite state automaton plants, as an alternative to the procedure of Ramadge and Wonham [6]. Fu et al. [3] [2] have reported optimal and robust control of finite state automata based on the language measure to formalize quantitative analysis and synthesis of DES control laws. The approach is state-based and the language measure parameters are identified from experiments on the physical process or from simulation experiments on a deterministic finite state automaton (DFSA) model of the plant [9]. However, using memoryless state-based tools for supervisory control synthesis may suffer serious shortcomings if the details of transitions cannot be captured by finitely many states. This problem has been partially circumvented by Petri nets that can accommodate certain classes of non-regular languages [5] in the Chomsky hierarchy. There is apparently no quantitative tools for supervisory control synthesis of Petri nets compared to what are available for finite state automata [3] [2]. Hence, there is a need for developing quantitative tools of supervisory control synthesis for discrete-event systems that cannot be represented by regular languages. Toward achieving this goal, the first step is to construct measure(s) of non-regular languages where the state-based approach [8] [7] may not be applicable.

This paper first shows that the measure of a regular language proposed in [8] [7] is equivalent to that of the regular grammar, without referring to states of the automaton. Then, the paper extends the signed real measure of regular languages to a complex measure for the class of non-regular languages, generated by the (deterministic) *linear context free grammar* (*LCFG*) that is a subclass of deterministic pushdown automata (DPDA) [4]. The main idea is to extend the signed real measure [8] [7] to a complex measure over the real field, where the multiplication operation of complex numbers is different from the conventional one. In this case, the complex space over the real field degenerates to the union of a pair of one-dimensional real spaces instead of being isomorphic to the two-dimensional real space. The extended complex-valued language measure, formulated in this paper, is potentially applicable to analysis and synthesis of DES control laws where the plant model could be represented by an *LCFG*.

This paper is organized in six sections including the present section. Section II briefly introduces the notations and background materials for formal languages. Section III presents the measure of regular grammars and shows its equivalence with that of regular languages. Section IV extends the measure of regular grammars to that of linear grammars. Section V presents an example to elucidate the concept of measure for linear grammars. The paper is summarized and concluded in Section VII.

II. Concepts and Notations

This section introduces notations and background materials for formal languages along with definitions of key concepts.

Definition 2.1: A context free grammar (CFG) is a 4-tuple $\Gamma = (V, T, P, S)$, where V and T are mutually disjoint (i.e., $V \cap T = \emptyset$) finite sets of variables and terminals, respectively; and P is a finite set of productions by which strings are derived from the start symbol S. Each production in P is of the form $v \rightarrow \alpha$, where $v \in V$ and $\alpha \in (V \cup T)^*$.

Remark 2.1: The language generated by a grammar Γ consists of all strings obtained from legal (i.e., permissible) productions beginning with the start symbol.

Definition 2.2: A regular grammar is a CFG (V, T, P, S) where every production in P takes exactly one of the following two alternative pairs of forms (i.e., there are either right derivations or left derivations but not both):

$$\begin{cases} v \to \alpha w \\ v \to \alpha \end{cases} \quad \text{or} \quad \begin{cases} v \to w\alpha \\ v \to \alpha \end{cases} \tag{1}$$

where $v, w \in V$ and $\alpha \in T \cup \{\epsilon\}$; and ϵ is the empty string.

Remark 2.2: The generated language for a deterministic finite state automaton ($DFSA$) is a regular language [4].

Definition 2.3: A linear grammar is a CFG (V, T, P, S) where every production in P takes one of the following forms:

$$v \to \alpha w; \quad v \to w\alpha; \quad v \to \alpha \tag{2}$$

where $v, w \in V$ and $\alpha \in T \cup \{\epsilon\}$.

Remark 2.3: In view of Remark 2.1 and Definition 2.3, the set of production rules P in a linear grammar $\Gamma = (V, T, P, S)$ can be modified as $\tilde{\Gamma} = (\tilde{V}, T, \tilde{P}, S)$ by augmenting the set V of variables as $\tilde{V} \equiv V \cup A$ and by updating the set P of production rules by $\tilde{P}$ that is given as:
if $\varphi = (v \to \alpha) \in P$ is replaced by $\tilde{\varphi} = (v \to a\alpha)$ where $v \in V$, $\alpha \in T \cup \{\epsilon\}$, and $a \in A$. This is analogous to the trim operation in regular languages [1] [6].

Remark 2.4: The modified grammar $\tilde{\Gamma} = (\tilde{V}, T, \tilde{P}, S)$ is a superset of the original grammar $\Gamma = (V, T, P, S)$ in the sense that it contains the generated language of $\Gamma = (V, T, P, S)$ and, in addition, has productions of the type $v \to aw$. The production $A \to \epsilon$ is added to P in each step of the modification; and $v \to \epsilon$ must exist $\forall v \in V$.

Remark 2.5: Regular grammars have only right (or left) derivations with a single variable. In contrast, linear grammars include both right and left derivations with a single variable. This is precisely what allows the linear grammars to model a certain class of non-regular languages.

In the sequel, the terms ***state*** and ***variable*** are used interchangeably as they convey the similar meaning in the present context; the same applies to the terms ***terminals*** and ***events***.

III. Measure of Regular Grammars

This section first introduces the concept of regular-grammar-based measures and then shows its equivalence to that of recently reported state-based measure [8] [7]. In essence, the concept of the state-based language measure is reformulated in terms of regular grammars, followed by construction of the measure. While detailed proofs of the supporting theorems are given in [4], sketches of the proofs that are necessary for developing the underlying theory are presented here.

Theorem 3.1: If L is a regular language, then $\exists$ a regular grammar Γ such that either $L = L(\Gamma)$ or $L = L(\Gamma) \cup \{\epsilon\}$. *Proof:* Let $G = (Q, \Sigma, \delta, q_i, A)$ be an FSA. We construct the grammar Γ with $V = Q$ and $T = \Sigma$. The set of productions is constructed as follows:

1) Add $q_i \to s_r q_j$ if $\delta(q_i, s_r) = q_j$;
2) Add $q_i \to s_r$ if $\delta(q_i, s_r) \in A$

where $q_i, q_j \in Q$ and $s_r \in \Sigma$. ■

Theorem 3.2: If Γ is a regular grammar, then $L(\Gamma)$ is a regular language.

Proof: Let $\Gamma = (V, T, P, S)$ be a regular grammar. We construct a nondeterministic finite state automaton G that exactly accepts the language $L(\Gamma)$. Specifically, let $G = (V \cup \{W\}, T, \delta, S, \{W\})$ where W is the only marked state and δ is defined as follows:

$$\delta(v_i, s_r) \equiv \delta_1(v_i, s_r) \cup \delta_2(v_i, s_r) \cup \delta_3(v_i, s_r) \tag{3}$$

where
$$\begin{aligned} \delta_1(v_i, s_r) &= v_j, && \text{if } v_i \to s_r v_j \in P \\ \delta_2(v_i, s_r) &= \{W\}, && \text{if } v_i \to s_r \in P \\ \delta_3(v_i, s_r) &= \phi, && \text{otherwise} \end{aligned}$$
■

Remark 3.1: It follows from Theorem 3.2 and Theorem 3.1 that a language L is regular iff $\exists$ a regular grammar Γ such that either $L = L(\Gamma)$ or $L = L(\Gamma) \cup \{\epsilon\}$. Therefore, $\exists$ a regular grammar for every finite state automaton that exactly generates the language of the regular grammar and vice versa.

A. Formulation of Regular Grammar Measures

This section follows the same construction procedure as in [8] [7] because there exists a one-to-one-correspondence between the state set Q of an automaton and the variable set V of the corresponding grammar. The same holds true for the alphabet set Σ and the terminal T of the regular grammar. The notion of marked states as well as that of good and bad marked states translates naturally to this framework. The variable set V can be partitioned into sets of marked variables V_m and non-marked variables $V - V_m$. The set V_m is further partitioned into good and bad marked variables as V_m^+ and V_m^-.

Definition 3.1: The language $L(\Gamma_i)$ generated by a context free grammar (CFG) Γ_i initialized at state $v_i \in V$ is defined as:

$$L(\Gamma_i) = \{s \in \Sigma^* | \ \exists \ a \ derivation \ of \ s \ from \ \Gamma_i\} \tag{4}$$

Definition 3.2: The language $L_m(\Gamma_i)$ generated by a CFG Γ_i initialized at state $v_i \in V$ is defined as:
$L_m(\Gamma_i) = \{s \in \Sigma^* | \exists$ a derivation of s from Γ_i which terminates on a marked variable $\}$

Definition 3.3: For every $v_i, v_k \in V$, the set of all strings that, starting from v_i, terminate on v_k is defined as the language $L(v_i, v_k)$. That is,
$L(v_i, v_k) = \{s \in \Sigma^* | \ \exists$ a derivation of s from v_i that terminates on $v_k\}$

Definition 3.4: The characteristic function $\chi : V \to [-1, 1]$ is defined in exact analogy with the state based approach:

$$\forall v_i \in V, \qquad \chi(v_i) \in \begin{cases} [-1, 0), & v \in {V_m}^- \\ \{0\}, & v \notin V_m \\ (0, 1], & v \in {V_m}^+ \end{cases}$$

and it assigns a signed real weight to the sublanguages $L(v_i, v)$.

Similar to the measure of regular languages [8] [7], the characteristic vector is denoted as: $\mathbf{X} = [\chi_1\ \chi_2\ \cdots\ \chi_n]^T$, where $\chi_j \equiv \chi(v_j)\ \forall k$, is called the $\mathbf{X}$-vector. The j-th element χ_j of $\mathbf{X}$-vector is the weight assigned to the corresponding terminal state v_j. Hence, the $\mathbf{X}$-vector is also called the state weighting vector.

As mentioned before, the marked language $L_m(\Gamma_i)$ consists of both good and bad event strings that, starting from the initial state v_i, lead to V_m^+ and V_m^- respectively. Any event string belonging to the language $L^0(\Gamma_i) = L(\Gamma_i) - L_m(\Gamma_i)$ terminates on one of the non-marked states belonging to $V - V_m$; and L^0 does not contain any one of the good or bad strings. Based on the equivalence classes defined in the Myhill-Nerode Theorem [4] that holds good since we are dealing with regular languages in the guise of regular grammars. The regular languages $L(\Gamma_i)$ and $L_m(\Gamma_i)$ can be expressed as:

$$L(\Gamma_i) = \bigcup_{v_k \in V} L(v_i, v_k) = \bigcup_{k=1}^{n} L(v_i, v_k) \qquad (5)$$

$$L_m(\Gamma_i) = \bigcup_{v_k \in Q_m} L(v_i, v_k) = L_m^+(\Gamma_i) \cup L_m^-(\Gamma_i) \qquad (6)$$

where the sublanguage $L(v_i, v_k) \subseteq \Gamma_i$, having the initial state v_i, is uniquely labeled by the terminal state $v_k, k \in \mathcal{I}$ and $L(v_i, v_j) \cap L(v_i, v_k) = \emptyset\ \forall j \neq k$; and $L_m^+ \equiv \bigcup_{v \in V_m^+} L(v_i, v)$ and $L_m^- \equiv \bigcup_{v \in V_m^-} L(v_i, v)$ are good and bad sublanguages of $L_m(\Gamma_i)$, respectively. Then, $L^0(\Gamma_i) = \bigcup_{v \notin V_m} L(v_i, v)$ and $L(\Gamma_i) = L^0(\Gamma_i) \cup L_m^+(\Gamma_i) \cup L_m^-(\Gamma_i)$.

Now we construct a signed real measure $\mu : 2^{L(\Gamma_i)} \rightarrow \mathbf{R} \equiv (-\infty, +\infty)$ on the σ-algebra $K = 2^{L(\Gamma_i)}$. The construction is exactly equivalent to that for the state-based automata. With the choice of this σ-algebra, every singleton set made of an event string $\omega \in L(\Gamma_i)$ is a measurable set, which qualifies itself to have a numerical quantity based on the above decomposition of $L(\Gamma_i)$ into L^0(null), L^+(positive), and L^-(negative), respectively called null, positive, and negative sublanguages. The event costs are defined below.

Definition 3.5: The event cost of the regular grammar Γ_i is defined as: $\tilde{\pi} : \Sigma^* \times V \rightarrow [0, 1]$ such that $\forall v_i \in V$, $\forall \sigma_j \in \Sigma$, $\forall \mathcal{S} \in \Sigma^*$,

(1) $\tilde{\pi}[\sigma_j, v_i] \equiv \tilde{\pi}_{ij} \in [0, 1)$; $\sum_j \tilde{\pi}_{ij} < 1$;
(2) $\tilde{\pi}[\sigma_j, v_i] = 0$ if $\nexists v_i \rightarrow \sigma_j v_k \in P$, where P is the set of production rules; and $\tilde{\pi}[\epsilon, v_i] = 1$;
(3) $\tilde{\pi}[\sigma_j \omega, v_i] = \tilde{\pi}[\sigma_j, v_i]\ \tilde{\pi}[\omega, v_k]$, $v_i \rightarrow \sigma_j v_k \in P$.

Definition 3.6: The state transition cost of the regular grammar Γ_i is defined as: $\pi : V \times V \rightarrow [0, 1)$ such that $\forall v_i, v_j \in V$, $\pi[v_i, v_j] = \sum_{\sigma \in \Sigma : \exists v_i \rightarrow \sigma v_j \in P} \tilde{\pi}[\sigma, v_i] \equiv \pi_{ij}$ and $\pi_{ij} = 0$ if $\{\sigma \in \Sigma : v_i \rightarrow \sigma v_j\} \cap P = \emptyset$. The $n \times n$ state transition cost $\mathbf{\Pi}$-matrix is defined as:

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

Remark 3.2: The variable-based definition (3.5)and state-based definition [8] [7] of the event cost function are exactly equivalent and so are definitions of the state transition cost. Therefore, the same Π-matrix will be obtained if the variables are interpreted as states.

Definition 3.7: The signed real measure μ of every singleton string set $S = \{s\} \in 2^{L(\Gamma_i)}$ where $s \in L(v_i, v)$ is defined as $\mu(S) \equiv \tilde{\pi}(s, v_i)\chi(v)$. The signed real measure of the sublanguage $L(v_i, v) \subseteq L(\Gamma_i)$ is defined as

$$\mu(L(v_i, v)) = \left(\sum_{s \in L(v_i, v)} \tilde{\pi}[s, v_i] \right) \chi(v) \qquad (7)$$

The signed real measure of the language of a regular grammar Γ_i initialized at a state $v_i \in V$, is defined as:

$$\mu_i \equiv \mu(L(\Gamma_i)) = \sum_{v \in \Gamma} \mu(L(v, v_i)) \qquad (8)$$

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

Based on the reasoning of the state based approach, it follows that:

$$\mu_i = \sum_j \pi_{ij} \mu_j + \chi_i \qquad (9)$$

In vector form, $\boldsymbol{\mu} = \mathbf{\Pi}\boldsymbol{\mu} + \mathbf{X}$ whose solution is given by:

$$\boldsymbol{\mu} = (\mathbf{I} - \mathbf{\Pi})^{-1}\mathbf{X} \qquad (10)$$

Remark 3.3: The matrix Π is a contraction operator and hence $(I - \Pi)$ is invertible. So, the $\boldsymbol{\mu}$-vector in Equation (9) is uniquely defined.

IV. Language Measure for Linear Grammars

This section extends the concept of language measure to linear grammars that are a generalization of regular grammars [4]. This is accomplished by introducing the notion of *event plane* from the perspective that there exists a specific direction in which an event may occur in a linear grammar.

Definition 4.1: The event mapping $\eta : \Sigma \rightarrow \mathbb{Z}$ is a function that maps the event alphabet into the set of integers. Let $\Sigma = \{\sigma_1, \cdots, \sigma_k, \cdots, \sigma_m\}$, then

$$\eta(\sigma_k) = k \qquad (11)$$

The *event plane* can be viewed as the complex plane itself on which the trajectory of the discrete-event system is reconstructed as the strings are generated. The transitions $S \rightarrow \sigma_k v_i$ and $S \rightarrow v_i \sigma_k$ transfer the state located at the origin $0, 0$, to $(\eta(\sigma_k), 0)$ and $(0, \eta(\sigma_k))$, respectively. Thus, there exists two possible directions in which the same event σ_k may cause transition from the same state v_i. For a regular grammar, events always occur along a direction that may be either the real axis or the imaginary axis, depending on how the representation of a right regular or a left regular. This concept is further clarified by using the notion of the event plane.

Let us denote an event as σ if it occurs along the real axis and as $i\sigma$ if it occurs along the imaginary axis; let the alphabet of real events be named Σ and the alphabet of imaginary events as $i\Sigma$. Note that Σ and $i\Sigma$ are disjoints finite sets of identical cardinality. However, each $\sigma_j \in \Sigma$ is uniquely identified with the specific $i\sigma_j \in i\Sigma$.

A. Linear Grammar Measure Construction

Several concepts need to be clarified before embarking on the construction of a measure. In finite state machines, a state q_j may be reached from a state q_i through different paths. In analogy, there may exist multiple paths between two states of a linear grammar. Whereas a particular string uniquely determines a path of a given automaton, a string may be generated by different paths in linear grammars.

Definition 4.2: The path mapping function $\wp : \Sigma \cup i\Sigma \rightarrow \Sigma$ generates a string of real events by first concatenating all the real events followed by reverse concatenation of the real events corresponding to the remaining imaginary events. For example, a path mapping is: $\wp(\sigma_j \sigma_k i\sigma_\ell \sigma_p i\sigma_q \sigma_r) = \sigma_j \sigma_k \sigma_p \sigma_r \sigma_q \sigma_\ell$.

It follows from Definition (4.2) that, given a path ω in the language, the generated string is obtained by the path mapping $\wp(\omega)$. The objective is to construct a measure of the set of all such paths rather than the measure of strings. This is necessary for a general form of the Myhill-Nerode theorem to hold, which is central to the construction of the linear grammar measure.

Let $\Upsilon \equiv \{x : x = a + ib \text{ and } a \in [0,1], b \in [0,1]\}$, i.e., Υ is the closed unit square on the complex plane C. Let a binary operator $\star : C \times C \rightarrow C$ be defined as: $(a + ib) \star (c + id) = ac + ibd \ \forall a, b, c, d \in R$. The identity for this operator is $1 + i$ since $\forall z \in C, z \star (1 + i) = z$ and if $z = a + ib$ with $a \neq 0$ and $b \neq 0$, then $\exists\ z^{-1} \in C$ such that $z \star z^{-1} = (1 + i)$. Specifically, if $z = a + ib$ with $a \neq 0$ and $b \neq 0$, then $z^{-1} = \frac{1}{a} + i\frac{1}{b}$.

The operator $\star$ can be extended as: $\star : C^{n\times m} \times C^{m\times l} \rightarrow C^{n\times l}$ as follows: if $\mathcal{A} \in C^{n\times m}$, $\mathcal{B} \in C^{m\times l}$, then $\mathcal{A} \star \mathcal{B} = \mathcal{C} \in C^{n\times l}$ in the sense that $c_{ij} = \Sigma_{k=1}^{n} a_{ik} \star b_{kj}$. Further, if $\mathcal{A} = A_r + iA_{im}$ and $\mathcal{B} = B_r + iB_{im}$ where the pairs (A_r, B_r) and (A_{im}, B_{im}) denote the real and imaginary parts of the matrices $\mathcal{A}$ and $\mathcal{B}$, respectively, it follows that $\mathcal{A} \star \mathcal{B} = A_r B_r + iA_{im} B_{im}$.

The identity for the above $\star$ operation is $(1 + i)I$ where I is the standard identity matrix of dimension $n \times n$. Let us denote the identity for the $\star$ operation by:

$$\mathscr{I} = (1 + i)I \tag{12}$$

where $\mathcal{A} \star \mathscr{I} = \mathscr{I} \star \mathcal{A} = \mathcal{A} \quad \forall \mathcal{A} \in C^{n\times n}$.

Remark 4.1: The inverse of a matrix $\mathcal{A} \in C^{n\times n}$ under the $\star$ operation, is given as: $\mathcal{A}^{\S} = (\mathcal{A}_r + i\mathcal{A}_{im})^{\S} = \mathcal{A}_r{}^{-1} + i\mathcal{A}_{im}{}^{-1}$ that is different from the standard inverse $\mathcal{A}^{-1}$. Notice that both real ($\mathcal{A}_r$) and imaginary ($\mathcal{A}_{im}$) parts of the matrix $\mathcal{A}$ must be individually invertible in the usual sense for existence of $\mathcal{A}^{\S}$.

In view of the $\star$ operator, definitions of the language measure parameters, χ, $\tilde{\pi}$ and π, are generalized as follows:

Definition 4.3: The characteristic function $\chi : V \rightarrow \Upsilon$ is defined $\forall v \in V$ as:

$$\chi(v) = \begin{cases} k(1+i) \ with \ k \in [-1, 0) & if \ v \in V_m{}^- \\ k(1+i) \ with \ k = 0 & if \ v \notin V_m \\ k(1+i) \ with \ k \in (0, 1] & if \ v \in V_m{}^+ \end{cases} \tag{13}$$

The characteristic value χ assigns a complex weight to a language $L(v, u)$ that, starting at the variable v, ends at the variable u. A real weight, in the range of -1 to +1, is assigned to each state as it was done in the case of real grammars, and then this weight is made complex by multiplying (in the usual sense) with $1 + i$. Each event can occur either along the real axis or along the imaginary axis on the event plane. The events are defined to be elements of the set $\{1, i\} \times \Sigma$. An event $\sigma \in \Sigma$ is denoted as $i\sigma$ if it occurs along the imaginary axis. Real occurrences are denoted just by σ similar to what was done for real grammars.

Definition 4.4: The event cost of the LCFG Γ is defined as a function $\tilde{\pi} : (\{1, i\} \times \Sigma)^* \times V \rightarrow \Upsilon$ such that $\forall v_k, v_\ell \in V$, $\forall \sigma_j \in \Sigma$, $\forall \omega \in (\{1, i\} \times \Sigma)^*$,

(1) $\tilde{\pi}[\sigma_j, v_k] \equiv \mathbb{R}e(\tilde{\pi}_{kj}) \in [0, 1)$; $\sum_j \mathbb{R}e(\tilde{\pi}_{kj}) < 1$;
(2) $\tilde{\pi}[i\sigma_j, v_k] \equiv \mathbb{I}m(\tilde{\pi}_{kj}) \in [0, 1)$; $\sum_j \mathbb{I}m(\tilde{\pi}_{ij}) < 1$;
(3) $\tilde{\pi}[\sigma_j, v_k] = i$ if $\nexists v_k \rightarrow \sigma_j v_\ell \in P$;
(4) $\tilde{\pi}[i\sigma_j, v_k] = 1$ if $\nexists v_k \rightarrow v_\ell \sigma_j \in P$;
(5) $\tilde{\pi}[\epsilon, v_\ell] = 1 + i$;
(6) $\tilde{\pi}[\tau\omega, v_k] = \tilde{\pi}[\tau, v_k] \star \tilde{\pi}[\omega, v_\ell]$
where $\tau \in \{1, i\} \times \Sigma$; $\omega \in (\{1, i\} \times \Sigma)^*$; and
$v_k \rightarrow \tau v_l$ if $\tau \equiv \sigma$ and $v_k \rightarrow v_\ell \tau$ if $\tau \equiv i\sigma$

Definition 4.5: The state transition cost of the LCFG Γ is defined as a function $\pi : V \times V \rightarrow \Upsilon$ such that $\forall v_k, v_j \in V$,

$$\begin{aligned} \pi[v_k, v_j] &= \sum_{\substack{\sigma_\ell \in \Sigma: \\ \exists v_k \rightarrow \sigma_\ell v_j \in P}} \mathbb{R}e(\tilde{\pi}[\sigma_\ell, v_k]) + i \sum_{\substack{\sigma_\ell \in \Sigma: \\ \exists v_k \rightarrow v_j \sigma_\ell \in P}} \mathbb{I}m(\tilde{\pi}[k, l]) \\ &\equiv \pi_{kj} \end{aligned} \tag{14}$$

and $\pi_{kj} = 0$ if $\{\sigma \in \Sigma : v_k \rightarrow \sigma v_j$ or $\sigma \in \Sigma : v_k \rightarrow v_j \sigma\} \cap P = \emptyset$. The $n \times n$ complex-valued state transition cost $\mathbf{\Pi}$-matrix is defined as:

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

Definition 4.6: Let Γ_k be a linear grammar, initialized at a state $v_k \in V$. The complex measure μ of every singleton string set $\Omega = \{\omega\} \in 2^{L(\Gamma_k)}$ is defined as: $\mu(\Omega) \equiv \tilde{\pi}(\omega, v_k) \star \chi(v)$. Then, the complex measure of the sublanguage $L(v_k, v) \subseteq L(\Gamma_k)$ of all strings terminated at the state $v \in V$ is defined as:

$$\mu(L(v_k, v)) = \left(\sum_{\omega \in L(v_k, v)} \tilde{\pi}[\omega, v_k] \right) \star \chi(v) \tag{15}$$

Complex measure of the language of the linear grammar Γ_k is defined as:

$$\mu_k \equiv \mu(L(\Gamma_k)) = \sum_{v \in \Gamma} \mu(L(v_k, v)) \tag{16}$$

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

B. Computation of the $\boldsymbol{\mu}$-Vector

This subsection presents a procedure to formulate the complex measure of paths in the language $L(\Gamma_k)$.

$$L_k = \left(\cup_j {\sigma_k}^j L_j \right) \cup_k \varepsilon_k$$

where the null event ε_k is defined as

$$\varepsilon_k = \begin{cases} \epsilon & \text{if self loop at } v_i \\ \emptyset & \text{otherwise} \end{cases}$$

The above expression formalizes the fact that the set of paths from a state v_k is exactly equal to the union of the sets of paths obtained by looking at the first event and then considering all possible legal paths thereafter. Hence, if the first event is σ_ℓ and the current state changes to v_j, then the set of all paths thereafter is exactly equal to L_j. The expression is structurally identical to that given for $DFSA$ in [8] with the understanding that the event ${\sigma_\ell}^j$ can be either real or imaginary. Hence,

$$\begin{aligned} \mu(L_k) &= \mu\bigg(\left(\cup_j \sigma_k^j L_j \right) \cup_k \varepsilon_k \bigg) \\ &= \mu\left(\cup_j \sigma_k^j L_j \right) + \mu(\varepsilon_k) \\ &= \sum_j \mu\big({\sigma_k}^j L_j\big) + \chi(v_k) \\ &= \sum_j \mu\big({\sigma_k}^j\big) \ \star \ \mu(L_j) + \chi(v_k) \\ &= \sum_j \pi_{kj} \ \star \ \mu(L_j) + \chi(v_k) \end{aligned}$$

The first three steps in above equation follow from the fact that if the first symbol for two paths is different, then the paths cannot be identical. However, the strings generated may still be the same. The fourth step follows from property (6) of the $\tilde{\pi}$ function in Definition 4.4. The final step trivially follows from the definition of the measure in 4.6. In vector form, the complex measure μ is given by:

$$\begin{aligned} \boldsymbol{\mu}(L) &= \boldsymbol{\Pi} \star \mu + \mathbf{X} = (\mathscr{I} - \boldsymbol{\Pi})^{\S} \star \mathbf{X} \\ &= \bigg((\mathbf{I} - \mathbb{R}e\boldsymbol{\Pi}) + i(\mathbf{I} - \mathbb{I}m\boldsymbol{\Pi}) \bigg)^{\S} \star \mathbf{X} \\ &= (\mathbf{I} - \mathbb{R}e\boldsymbol{\Pi})^{-1}\mathbb{R}e\mathbf{X} + i(\mathbf{I} - \mathbb{I}m\boldsymbol{\Pi})^{-1}\mathbb{I}m\mathbf{X} \end{aligned}$$

where $\mathbb{R}e$ and $\mathbb{I}m$ refer to the real and imaginary parts of the matrices, respectively; and existence of the matrix inverses is guaranteed by the following conditions:

$$\sum_j \mathbb{R}e(\tilde{\pi}_{kj}) < 1; \quad \sum_j \mathbb{I}m(\tilde{\pi}_{kj}) < 1 \ \ \forall k.$$

A simple example is presented in the next section to illustrate how the complex μ is computed for an *LCFG*.

V. Example

Let a language L generate all strings of the type $a^n b^n$: $n \in \mathcal{N}$ over the alphabet $\Sigma = \{a, b\}$. The non-regular language L can be generated by the grammar: $v_1 \to a v_1 b | \epsilon$ that can be rewritten as: $v_1 \to a v_2$; and $v_2 \to v_1 b$. The resulting $\tilde{\pi}$ matrix is of the form:

$$\tilde{\boldsymbol{\Pi}} = \begin{bmatrix} p & 0 \\ 0 & iq \end{bmatrix}$$

The parameters p and q can be identified from the experimental time series data of the system dynamics [9]. The $\boldsymbol{\Pi}$ matrix is then obtained as:

$$\mathbb{R}e\boldsymbol{\Pi} = \begin{bmatrix} 0 & p \\ 0 & 0 \end{bmatrix} \text{ and } \mathbb{I}m\boldsymbol{\Pi} = \begin{bmatrix} 0 & 0 \\ q & 0 \end{bmatrix}.$$

Assigning characteristic values (i.e., weights) of the two states v_1 and v_2 to be χ_1 and χ_2 respectively, the complex measure vector is evaluated as:

$$\boldsymbol{\mu}(L) = \left\{ \begin{array}{c} (\chi_1 + p\chi_2) + \mathbf{i}\chi_1 \\ \chi_2 + \mathbf{i}(\chi_2 + q\chi_1) \end{array} \right\}$$

VI. Control under Event Unobservability

This section presents future work on extension of state-based optimal and robust control [3] [2] that is achieved by selectively disabling controllable events to maximize the measure of the controlled plant language under the restriction of all events being observable. It might be possible to eliminate this restriction by making the state transitions due to unobservable events as imaginary. However, with this modification, the generated language may become nonregular with a linear grammar. A detailed exposition follows.

Let the plant be modelled as a DFSA $G = (Q, \Sigma, \delta, q_1, A)$. The corresponding regular grammar is obtained as $\Gamma(q_1, V, T, P)$ with $V = Q$ and $T = \Sigma$ by Theorem 3.1. Further let $\Omega \subset \Sigma$ be the set of unobservable events, i.e., $\Omega \equiv \{\sigma \in \Sigma : \alpha \ is \ unobservable\}$. The set of production rules is then modified as follows:

$$\forall q_i, q_j \in Q \ \ \forall (q_i \to s_r q_j) \in P, \textit{ if } s_r \in \Omega$$
$$\textit{then replace } (q_i \to s_r q_j) \textit{ by } (q_i \to q_j s_r)$$

In accordance with Section IV, this modification implies that the unobservable events lead to imaginary transitions. The modified set of production rules has both left and right derivations and hence the modified grammar is linear. Let us denote this operation as *coloring* the regular grammar, which defines the mapping as $Col : REG \times \Omega \to LIN$, where REG is the set of regular grammars and LIN is the set of deterministic linear grammars. The properties of the mapping *Col* are as follows:

1) *Col* is well defined which follows from the algorithmic definition of the mapping;
2) $\Omega = \varnothing \Rightarrow Col(\Gamma_r, \Omega) = \Gamma_r \ i.e.$ *Col* is identity if there are no unobservable events;
3) *Col* is one to one. Injectivity follows trivially from the algorithmic definition. However *Col* is not surjective.

Next the notion of ordered set operations on strings is introduced.

Definition 6.1: Ordered difference of two strings s_1 and s_2 is the first string with the symbols occurring in the second one deleted. Thus it is a mapping $\ominus : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$

Definition 6.2: Ordered intersection of a string s_1 and a subset K of the alphabet Σ is the string s_1 with the symbols occurring in K deleted. Formally it is a mapping $\ominus : \Sigma^\star \times K \subset \Sigma \to \Sigma^\star$. Note that the same symbol is used to denote both ordered difference and ordered intersection since the case will be definitely clear from context.

Definition 6.3: Reversal of a string s_1 is a string s_2 which is just s_1 read right to left. Formally it is a mapping $(\centerdot)^r : \Sigma^\star \to \Sigma^\star$

Definition 6.4: Observed component of a string σ denoted by $\eth(\sigma)$ is defined as: $\eth(\sigma) = \sigma \ominus \Omega$.

Physically, the observed component of a string generated by a regular grammar with a nonempty set of unobservable events is simply the string that is observed, which is different from the actual generated string. $\eth$ is therefore a formal mapping $\eth : \Sigma^\star \to \Sigma^\star$ for a given Ω and note for an empty Ω, $\eth$ is the identity map.

In both regular and linear grammars, there is exactly one variable and one terminal on the right hand side of each production rule. Trivial exceptions that may occur is taken care of by the $trim$ operation described in Remark 2.3. This implies that, at any given time step, the derivation consists of a string of terminals and exactly one variable. If we denote the derivation at the k^{th} step by $\overline{s}_k$, then for a regular grammar $\overline{s}_k = \sigma_1 \ldots \sigma_k q_i$ by successively application of production rules to the derivations, where $\sigma_j \in \Sigma$ and $q_i \in Q$. In this case, an object consists of string fragments and a single variable. The variable is replaced in the successive step in accordance with the production rules P; such derivations are called $livestrings$. For linear grammars, the expression for $\overline{s}_k$ is more involved in the sense that the variable is no longer restricted to be located at the end. This induces a map $col_\Omega : \overline{\Sigma}_{\Gamma_r} \to \overline{\Sigma}_{Col(\Gamma_r)}$ where $\overline{\Sigma^\star}_{\Gamma_r}$ and $\overline{\Sigma^\star}_{Col(\Gamma_r)}$ are the sets of all possible live strings for Γ_r and $Col(\Gamma_r, \Omega)$ respectively. Specifically, if $col_\Omega(\overline{s}_k) = \overline{\sigma}_k$ then $\overline{s}_k$ is a live string for Γ_r and $\overline{\sigma}_k$ is a live string for $Col(\Gamma_r, \Omega)$. To calculate $\overline{\sigma}_k$ explicitly one first determines the unique sequence of events that leads to the live string $\overline{s}_k$. The sequence of events remains the same in $Col(\Gamma, \Omega)$, but the unobservable transitions are now imaginary. This sequence and Col determine the corresponding live string $\overline{\sigma}_k$. Consequently, $\overline{s}_k$ and $\overline{\sigma}_k$ in the last expression is related to the observed component of the live string. The variable represents the *generation point* in a string ($i.e.$, the point at which new symbols are written), it physically denotes the point of current observation. The substring to the left of the variable is the observed component of the generated string and the one to the right is the sequential set of the unobservable events occurring in the system up to the current step.

Pertinent results are summarized below as two theorems, whose proofs are not presented here due to space limitations.

Theorem 6.1: $col_\Omega(\overline{s}_k) = [\eth(\overline{s}_k)][\overline{s}_k \ominus \Sigma][(\overline{s}_k \ominus \eth(\overline{s}_k))^r]$ where $[\centerdot][\centerdot]$ denotes concatenation.

Theorem 6.2: A system is observable if and only if $col_\Omega \equiv identity \ \forall \ \overline{s}_k \in \overline{\Sigma^\star}_{Col(\Gamma_r, \Omega)}$.

VII. Summary and Conclusions

The major motivation for the work reported in this paper is discrete event supervisory (DES) control of complex dynamical systems that cannot be represented by regular languages (equivalently, finite state automata). Toward achieving this goal, the paper presents a quantitative measure of non-regular languages, generated by *linear context free grammars (LCFG)* which belong to the low end of Chomsky hierarchy [4]. It shows that the measure of regular languages proposed in [8] [7] can be obtained by its generating regular grammar, without referring to states of the automaton. Then, the paper extends the signed real measure to a complex measure for the class of non-regular languages, generated by *LCFG*. The extended language measure is potentially applicable to quantitative analysis and synthesis of DES control systems where the plant model of a complex dynamical system is not restricted to be a finite state machine. The paper has also discussed the issues of quantitative synthesis of optimal and robust control policies under partial observation.

Future research in this area includes generalization of the measure to a wider class of context free grammars and beyond, with specific interest in Petri nets. In addition, the issue of measuring paths rather than strings needs to be explored in more details.

References

[1] C.G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, Kluwer Academic Publishers, 1999.

[2] J. Fu, C.M. Lagoa, and A. Ray, *Robust optimal control of regular languages with event cost uncertainties*, Proceedings of IEEE Conference on Decision and Control, Maui, Hawaii (2003), 3209–3214.

[3] J. Fu, A. Ray, and C.M. Lagoa, *Optimal control of regular languages with event disabling cost*, Proceedings of American Control Conference, Denver, Colorado (2003), 1691–1695.

[4] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation, 2nd ed.*, Addison-Wesley, 2001.

[5] J.O. Moody and P.J. Antsaklis, *Supervisory control ofdiscrete event systems using petri nets*, Kluwer Academic, 1998.

[6] P. J. Ramadge and W. M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM J. Control and Optimization **25** (1987), no. 1, 206–230.

[7] A. Ray and S. Phoha, *Signed real measure of regular languages for discrete-event automata*, International Journal of Control **76** (2003), no. 18, 1800–1808.

[8] X. Wang and A. Ray, *A language measure for performance evaluation of discrete event supervisory control systems*, Applied Mathematical Modelling (2004), in press.

[9] X. Wang, A. Ray and A. Khatkhate, *On-line identification of language measure parameters for discrete event supervisory control*, IEEE Conference on Decision and Control, Maui, Hawaii (2003), 6307–6312.