A Complex Measure for Linear Grammars

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

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

Abstract—The signed real measure of regular languages, introduced and validated in recent 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). However, this approach relies on memoryless state-based tools for supervisory control synthesis and may become inadequate if the transitions in the plant dynamics cannot be captured by finitely many states. From this perspective, the measure of regular languages needs to be extended to that of non-regular languages, such as Petri nets or other higher level languages in the Chomsky hierarchy. Measures for non-regular languages has not apparently been reported in open literature and is an open area of research. This paper introduces a complex measure of linear context free grammars (LCFG) that belong to the class of non-regular languages. The proposed complex measure becomes equivalent to the signed real measure, reported in recent literature, if the LCFG 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 [5] because the task of discrete-event supervisory (DES) control synthesis becomes mathematically tractable and computationally efficient due to simplicity of regular languages. According to the paradigm of DES control, a finite-state automaton (e.g., the discrete-event model of a physical plant) is a language generator whose behavior is constrained by the supervisor to meet a given specification. The (controlled) sub-language 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 measures of regular languages has been reported in literature [7] to provide a mathematical framework for quantitative comparison of controlled sublanguages of the unsupervised plant language. This measure provides a total ordering of the sublanguages of the unsupervised plant and formalizes a procedure for synthesis of DES controllers for finite state automaton plants, as an alternative to the procedure of Ramadge and Wonham [5]. Optimal control of finite state automata has been recently have been reported [6] based on the language measure and formalizes quantitative analysis and synthesis of DES control laws. The approach is state-based and the language measure parameters are identified from physical experiments or simulation on a deterministic finite state automaton (DFSA) model of the plant [8]. 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 [4] in the Chomsky hierarchy [3]. There is apparently no quantitative tool for supervisory control synthesis of Petri nets compared to what are available for finite state automata [6]. Hence, there is a need for developing measures of non-regular languages as a quantitative tool 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 [7] may not be applicable.
This paper first shows that the measure of a regular language proposed in [7] is equivalent to that of the regular grammar, without referring to states of the automaton. Then, it extends the signed real measure of regular languages to a complex measure for the class of non-regular languages [2], generated by the (deterministic) linear context free grammar (LCFG) that is a subclass of deterministic pushdown automata (DPDA) [3]. The signed real measure is extended to a complex measure over the real field, where the multiplication operation of complex numbers is extended to a complex measure 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.

The paper is organized in six sections including the present one. Section II briefly introduces the basic concepts, notations and background materials for formal languages. Section III discusses the measure of regular grammars and shows its equivalence with that of regular languages. Section IV extends the measure to linear grammars and the concept is elucidated with an example in Section IV-C. Section V briefly describes two important future research directions. The paper is summarized and concluded in Section VI.

II. CONCEPTS AND NOTATIONS

This section introduces notations and background materials for formal languages [3] 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 $\Gamma$ 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):

$$
\begin{align*}
&v \rightarrow aw \\
&v \rightarrow a
\end{align*}
$$

(1)

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

Remark 2.2: The generated language for a deterministic finite state automaton (DFSA) is a regular language [3].

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

$$
\begin{align*}
&v \rightarrow aw; \quad v \rightarrow wa; \quad v \rightarrow a
\end{align*}
$$

(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 $\tilde{\Gamma} = (\hat{V}, \hat{T}, \hat{P}, S)$ by augmenting the set $V$ of variables as $\hat{V} \equiv V \cup A$ and by updating the set $P$ of production rules by $\hat{P}$. That is, $\varphi = (v \rightarrow \alpha) \in P$ is replaced by $\tilde{\varphi} = (v \rightarrow a\alpha) \in \hat{P}$, where $v \in V, \alpha \in T \cup \{\epsilon\}$ and $\alpha \in A$. This is analogous to the trim operation in regular languages [5].

Remark 2.4: The modified grammar $\tilde{\Gamma} = (\hat{V}, \hat{T}, \hat{P}, S)$ 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 $v \rightarrow aw$. The production $A \rightarrow \epsilon$ is added to $P$ in each step of the modification; and $v \rightarrow \epsilon$ 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.

A geometric approach is adopted in this paper to deal with the possible presence of both right and left derivations in a linear context free grammar. A production rule of the type $V \rightarrow \sigma V_1$ is fundamentally different form a production $V \rightarrow V_1 \sigma$. If $V_1$ is subsequently replaced by a symbol $\sigma_1$, the order of generation of $\sigma$ and $\sigma_1$ is maintained in the derived string in the first case and it is reversed in the second. Note that if $V \rightarrow V_1 \sigma$ is the first right linear production rule used in a particular derivation, then $\sigma$ is necessarily the last terminal in the derived string. This possible non-causality of derivations is handled by introducing
the following notions: imaginary transitions, generated path, path mapping function, and the event plane.

In this paper, generation of a symbol through a right linear production is denoted by an imaginary transition as opposed to a symbol generated by a left linear production which is denoted by a real transition. An imaginary transition is labelled by the prefixing $i$ with the generated symbol. For example, $V \rightarrow V_1\sigma$ implies $i\sigma$ has occurred while $V \rightarrow \sigma V_1$ implies simply $\sigma$ has occurred. The concept of real and imaginary transitions facilitates the notion of a generated path.

**Definition 2.4:** A generated path $\lambda \in \{(\epsilon, i) \times \Sigma\}^*$ is the sequential order of transitions (real or imaginary) in any particular derivation.

For example, if a derivation proceeds sequentially through the production rules $V_1 \rightarrow V_2\sigma_1$, $V_2 \rightarrow \sigma_2V_3$, $V_3 \rightarrow V_4\sigma_3$, $V_4 \rightarrow \sigma_4V_4$, then the generated path $\lambda \equiv i\sigma_1\sigma_2i\sigma_3\sigma_4$.

**Definition 2.5:** The set of all paths, generated by an LCFG $\Gamma$, is denoted as $P_{\Gamma} \in 2^{(\{(\epsilon, i) \times \Sigma\})^*}$.

A given generated path $\lambda$ corresponds to a particular derived string. However, in non-regular grammars, a single string may be derived through more than one paths. Mathematically, there exists a surjective mapping from the set of all generated paths by a LCFG to the set of all derived strings $i.e.$ the language generated by the grammar. Note, if the LCFG is regular, this map is injective and hence bijective.

**Definition 2.6:** The path mapping function $\rho : P_{\Gamma} \rightarrow L(\Gamma)$ is defined as follows: For any path $\lambda$, the corresponding derived string is obtained by concatenating all the symbols generated by real transitions followed by a concatenation of the ones generated by imaginary transitions in a reversed order.

For example, $\rho(\lambda \equiv i\sigma_1\sigma_2i\sigma_3\sigma_4) \rightarrow \sigma_4\sigma_3\sigma_2\sigma_1$. In regular grammars, the absence of imaginary transitions implies that $\rho$ is the identity map.

An example of the right invariant relation is the well-known Nerode equivalence relation $(\mathcal{N})$ on a language $L$ which is defined as follows [3]:

$$\forall x, y \in L, x\mathcal{N}y, \text{ if } \forall u \in \Sigma^*, xu \in L \Leftrightarrow yu \in L \tag{3}$$

A language $L$ is regular if and only if there exists a Nerode equivalence relation of finite index. Applying the notion of Nerode equivalence on $P_{\Gamma}$, it follows that if two paths $\lambda_1, \lambda_2$ are generated through production rule sequences $\{P_{11}, P_{12}, \ldots, P_{1n}\}$ and $\{P_{21}, P_{22}, \ldots, P_{2n}\}$ such that the final variable on the righthand side of the derivation is identical (say $V_i$) in the two cases, then $\lambda_1, \mathcal{N}\lambda_2$. This follows immediately from noting that if $\lambda$ is a path initiating from $V_i$, then both $\lambda_1, \lambda$ and $\lambda_2, \lambda$ are elements of $P_{\Gamma}$, and if $\lambda \in 2^{\{(\epsilon, i) \times \Sigma\}}$ cannot be generated from the variable $V_i$, then $\lambda_1, \lambda, \lambda_2, \lambda \notin P_{\Gamma}$. This observation suggests that at least in an LCFG, variables convey the same meaning as states in the context of 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. Note, the context-free nature of LCFG implies each variable can be rewritten by the specified production rules, irrespective of where the variable occurred. This is precisely the kind of Markov property that associates variables with states and terminals with events.

**Definition 2.7:** The event mapping $\eta : \Sigma \rightarrow \mathbb{Z}$ is a function that maps the event alphabet into the set of integers. Let $\Sigma = \{\sigma_1, \ldots, \sigma_k, \ldots, \sigma_n\}$, then

$$\eta(\sigma_i) = k \tag{4}$$

**Definition 2.8:** 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 $S \rightarrow \sigma_2V_i$ and $S \rightarrow V_i\sigma_4$ transfer the state located at the origin $(0, 0)$, to $(\eta(\sigma_i), 0)$ and $(0, \eta(\sigma_4))$, respectively. Thus, there exists two possible directions in which the same event $\sigma_k$ may cause transition from the same state $v_i$ depending on whether the event is causing an imaginary transition or a real transition. Figures 1 and 2 illustrate the idea. Note, for the regular grammar of Figure 2, the derivations always occur along the real axis of the event plane.

**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 [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 [3], sketches of the proofs that are necessary for developing the underlying theory are presented here.
construct a (possibly) nondeterministic finite state automaton language.

If \( \Gamma \) be a regular grammar. Let us construct the grammar having the linear grammar \( S \rightarrow \alpha T, T \rightarrow bS \).

Proof: Let \( G = (Q, \Sigma, \delta, q_0, A) \) be an FSA. Let us construct the grammar \( \Gamma \) with \( V = Q \) and \( T = \Sigma \). The set of productions is constructed as follows:

\[
\forall q_i, q_j \in Q, \quad s_i \in \Sigma, \quad \begin{cases} 
\text{Add } q_i \rightarrow s_i q_j \text{ if } \delta(q_i, s_i) = q_j \\
\text{Add } q_i \rightarrow s_i \text{ if } \delta(q_i, s_i) \in A
\end{cases}
\]

Theorem 3.1: If \( L \) is a regular language, then there is a regular grammar \( \Gamma \) such that either \( L = L(\Gamma) \) or \( L = \emptyset \).

Proof: Let \( G = (Q, \Sigma, \delta, q_0, A) \) be an FSA. Let us construct the grammar \( \Gamma \) with \( V = Q \) and \( T = \Sigma \). The set of productions is constructed as follows:

\[
\forall q_i, q_j \in Q, \quad s_i \in \Sigma, \quad \begin{cases} 
\text{Add } q_i \rightarrow s_i q_j \text{ if } \delta(q_i, s_i) = q_j \\
\text{Add } q_i \rightarrow s_i \text{ if } \delta(q_i, s_i) \in A
\end{cases}
\]

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

Proof: Let \( \Gamma = (V, T, P, S) \) be a regular grammar. Let us construct a (possibly) 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 \( \delta \) is defined as follows:

\[
\delta(v_i, s_j) = \begin{cases} 
\delta_1(v_i, s_j) \cup \delta_2(v_i, s_j) \cup \delta_3(v_i, s_j) \\
\delta_1(v_i, s_j) = v_j, \quad \text{if } v_i \rightarrow s_v v_j \in P \\
\delta_2(v_i, s_j) = [W], \quad \text{if } v_i \rightarrow s_v \in P \\
\delta_3(v_i, s_j) = \emptyset, \quad \text{otherwise}
\end{cases}
\]

Remark 3.1: It follows from Theorem 3.2 and Theorem 3.1 that a language \( L \) is regular iff there is a regular grammar \( \Gamma \) such that either \( L = L(\Gamma) \) or \( L = \emptyset \). Therefore, there is 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 [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 regular grammar. The same holds true for the alphabet set \( \Sigma \) 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 \( V - V_m \) and the set \( V_m \) is further partitioned into good and bad marked variables as \( V_m^+ \) and \( V_m^- \) [7].

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

\[
L(\Gamma_i) = \{ s \in \Sigma^* \mid \text{there is a derivation of } s \text{ from } \Gamma_i \}
\]

Definition 3.2: The language \( L_m(\Gamma_i) \) generated by a CFG \( \Gamma_i \) initialized at state \( v_i \in V \) is defined as:

\[
L_m(\Gamma_i) = \{ s \in \Sigma^* \mid \text{there is a derivation of } s \text{ from } \Gamma_i \text{ which terminates on a marked variable} \}
\]

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, \( L(v_i, v_k) = \{ s \in \Sigma^* \mid \text{there is a derivation of } s \text{ from } v_i \text{ that terminates on } v_k \} \).

Definition 3.4: The characteristic function \( \chi : V \rightarrow [-1, 1] \) is defined in exact analogy with the state based approach [7]:

\[
\forall v_i \in V, \quad \chi(v_i) \in \begin{cases} [-1, 0), & v \in V_m^- \\
[0], & v \notin V_m \\
(0, 1], & v \in V_m^+
\end{cases}
\]
and thus $\chi$ assigns a signed real weight to each of the sublanguages $L(v_i, v)$.

Similar to the measure of regular languages [7], the characteristic vector is denoted as: $X = [x_1, x_2, \cdots, x_l]^T$, where $x_j = \chi(v_j)$, is called the $X$-vector. The $j$-th element $x_j$ of $X$-vector is the weight assigned to the corresponding terminal state $v_j$. Hence, the $X$-vector is also called the state weighting vector in the sequel.

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 $V_m^n$ and $V_m^n$ 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. The regular languages $L(\Gamma_i)$ and $L_m(\Gamma_i)$ can be expressed as:

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

$$L_m(\Gamma_i) = \bigcup_{v_i \in V_m} L(v_i, v_k) = L_m^0(\Gamma_i) \cup L_m^n(\Gamma_i)$$

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

Now we construct a signed real measure $\mu : 2^{\Sigma I} \rightarrow \mathbb{R} \equiv (-\infty, +\infty)$ on the $\sigma$-algebra $K = 2^{\Sigma I}$. The construction is exactly equivalent to that for the state-based automata [7]. With the choice of this $\sigma$-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 $L^0$, $L^+$, and $L^-$, respectively called null, positive, and negative sublanguages. The event costs are defined below.

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

1. $\tilde{\pi}(\sigma_j, v_i) \equiv \pi_j \in (0, 1)$; $\sum_j \tilde{\pi}_j < 1$;
2. $\tilde{\pi}(\sigma_j, v_i) = 0$ if $\tilde{\pi}(\sigma_j, v_i) \in P$, where $P$ is the set of production rules; and $\tilde{\pi}(\epsilon, v_i) = 1$;
3. $\tilde{\pi}(\omega, v_i) = \prod_{j \in \Sigma} \tilde{\pi}(\sigma_j, v_j)$, where $v_i \rightarrow \sigma_j v_j \in P$.

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

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

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

$$\mu(L(v_i, v)) = \left(\sum_{\sigma \in L(v_i, v)} \tilde{\pi}(\sigma, v_i)\right) \chi(v)$$

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

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

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

Based on the reasoning of the state based approach [7], it follows that:

$$\mu_i = \sum_{v_i \in V} \pi_{ij} \chi_i$$

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

$$\mu = (I - \Pi)^{-1} X$$

**Remark 3.2:** The matrix $\Pi$ is a contraction operator [6] and hence $(I - \Pi)$ is invertible. So, the $\mu$-vector in Equation (10) is uniquely defined.

**IV. LANGUAGE MEASURE FOR LINEAR GRAMMARS**

This section extends the concept of language measure to linear grammars $(V, T, P, S)$ that are a generalization of regular grammars [3].

**A. Linear Grammar Measure Construction**

It follows from Definition 2.6 that, given a path $\omega$ in the language, the generated string is obtained by the path mapping $\phi(\omega)$. The approach of this paper is to construct a measure of the set of all such paths rather than the measure of strings. This is necessary since any attempt to apply the Myhill-Nerode theorem on a linear non-regular language results in an infinite number of equivalence classes (the Nerode equivalence relation is not of
finite index). However, as we will argue shortly, the language of all paths \( P_{\Gamma} \) is regular and hence a right invariant relation of finite index exists on \( P_{\Gamma} \). Then it follows from the reasoning in [7], that a well-defined signed real measure exists on \( P_{\Gamma} \). In this section we will construct a complex measure on \( P_{\Gamma} \) allowing one to differentiate between the real and imaginary transitions and is thus more intuitive in the case of linear non-regular grammars. Moreover, it is argued that the defined complex measure coincides for the \( P_{\Gamma} \) and \( L(\Gamma) \).

**Lemma 4.1:** If there exists a complex measure \( \theta \) on \( P_{\Gamma} \) with the \( \sigma \)-algebra \( 2^{P_{\Gamma}} \), i.e. if \( (P_{\Gamma}, 2^{P_{\Gamma}}, \theta) \) is a well-defined measure space then there exists a measure \( \mu \) on \( L(\Gamma) \) with the \( \sigma \)-algebra \( 2^{L(\Gamma)} \), such that

\[
\theta(P_{\Gamma}) = \mu(L(\Gamma))
\]

**Proof:**

Note the statement of the lemma does not state that \( (P_{\Gamma}, 2^{P_{\Gamma}}, \theta) \) is a well-defined measure space, but makes a claim in case it is. Assume this is in fact the case.

Now define the map \( \overline{\mu} : L(\Gamma) \rightarrow \mathbb{C} \) as follows:

\[
\forall \omega \in L(\Gamma), \quad \overline{\mu}(\omega) = \sum_{\lambda \in \varphi^* (\omega)} \theta(\lambda)
\]

That \( \overline{\mu} \) is well-defined follows immediately from the facts that \( (P_{\Gamma}, 2^{P_{\Gamma}}, \theta) \) is a measure space and \( \varphi \) is surjective. Now, \( \overline{\mu} \) induces a map \( \mu : 2^{L(\Gamma)} \rightarrow \mathbb{C} \) as follows:

\[
\forall \Omega \subseteq L(\Gamma), \quad \mu(\Omega) = \sum_{\omega \in \Omega} \overline{\mu}(\omega)
\]

It is trivial to check that \( \mu \) is in fact a measure on the space \( L(\Gamma) \).

It then follows immediately, that

\[
\mu(L(\Gamma)) = \sum_{\omega \in L(\Gamma)} \overline{\mu}(\omega) = \sum_{\lambda \in \varphi^*(\omega)} \theta(\lambda) = \theta(P_{\Gamma})
\]

**Remark 4.1:** The construction of Lemma 4.1 is actually the natural way of inducing a measure on the quotient space of a measure space.

**Lemma 4.2:** For a linear grammar \( \Gamma \), the path language \( P_{\Gamma} \) is regular.

**Proof:** The statement immediately follows from the fact that \( P_{\Gamma} \) is a language on the alphabet \( \Sigma \cup \Sigma \) and can be generated by a left-linear grammar with production rules of the type \( S \rightarrow \sigma V \) or \( S \rightarrow \sigma V \).

We continue with our explicit construction as follows:

Let \( \Gamma = \{ x : x = a + ib \text{ and } a \in [0,1], b \in [0,1] \} \), i.e., \( \Gamma \) is the closed unit square on the complex plane \( C \). Let a binary operator \( \star : C \times C \rightarrow C \) be defined as: \( (a + ib) \star (c + id) = ac + ibd \), \( \forall a, b, c, d \in \mathbb{R} \). The identity for the operator \( \star \) 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 there exist \( a \) a unique \( z^{-1} \in C \) such that \( z^{-1} = \frac{1}{a} + i\frac{1}{b} \). That is, \( z \star z^{-1} = (1 + i) \). The operator \( \star \) can be extended to multi-dimensional cases by \( \star : C^{\alpha \times \beta} \rightarrow C^{\alpha \times \beta} \) as follows:

If \( A \in C^{\alpha \times \beta} \), then \( A \star B = C \in C^{\alpha \times \beta} \) in the sense that \( c_{ij} = \sum_{k=1}^{\alpha} a_{ik} \star b_{kj} \). Further, if \( A = A_r + iA_{im} \) and \( B = B_r + iB_{im} \) where the pairs \( (A_r, B_r) \) and \( (A_{im}, B_{im}) \) denote the real and imaginary parts of the matrices \( A \) and \( B \), respectively, it follows that \( A \star B = A_r B_r + iA_{im} B_{im} \).

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:

\[
I = (1 + i)I
\]

where \( A \star I = I \star A = A \quad \forall A \in C^{\alpha \times \beta} \).

**Remark 4.2:** The inverse of a matrix \( A \in C^{\alpha \times \beta} \) under the \( \star \) operation, if it exists, is given as:

\[
A^{-1} = (A_r + iA_{im})^{-1} = A_r^{-1} + iA_{im}^{-1}
\]

that is different from the standard inverse \( A^{-1} \). Notice that both real \( (A_r) \) and imaginary \( (A_{im}) \) parts of the matrix \( A \) must be individually invertible in the usual sense for existence of \( A^{-1} \).

In view of the \( \star \) operator, definitions of the language measure parameters, \( \chi, \tilde{\chi}, \pi \), are generalized as follows:

**Definition 4.1:** The characteristic function \( \chi : V \rightarrow \Gamma \) is defined as:

\[
\forall v \in V, \quad \chi(v) = \begin{cases} 
  k(1 + i) & \text{if } v \in V_- \\
  0 & \text{if } v \notin V_-
\end{cases}
\]

where

\[
\begin{align*}
&k \in [-1,0) \quad \text{if } v \in V_-^c \\
&k = 0 \quad \text{if } v \notin V_-
\end{align*}
\]

The characteristic value \( \chi \) 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 $1 + i$. 

**Definition 4.2:** The event cost of the LCFG $\Gamma$ is defined as a function $\tilde{\sigma} : (\Sigma \cup \Sigma^*) \times V \to \Upsilon$ such that $\forall v_k, v_l \in V, \forall \sigma_j \in \Sigma, \forall \omega \in (\Sigma \cup \Sigma)^*$,

1. $\tilde{\sigma}(\sigma_j, v_k) = \Re(\tilde{\sigma}(\omega), v_l) \in [0, 1]; \sum_{k \in [0, 1]} \Re(\tilde{\sigma}(\omega), v_l) < 1$;
2. $\tilde{\sigma}(\sigma_j, v_k) = \Im(\tilde{\sigma}(\omega), v_l) \in (0, 1); \sum_{k \in (0, 1)} \Im(\tilde{\sigma}(\omega), v_l) < 1$;
3. $\tilde{\sigma}(\sigma_j, v_k) = 1$ if $\tilde{\sigma}(\omega, v_l) \in P$;
4. $\tilde{\sigma}(\epsilon, v_l) = 1 + i$;
5. $\tilde{\sigma}(\tau \omega, v_l) = \tilde{\sigma}(\tau, v_l) \star \tilde{\sigma}(\omega, v_l)$

where $\tau \in (\Sigma \cup \Sigma^*) \times \omega \in (\Sigma \cup \Sigma)^*$; and $v_k, v_l \in \tau v_l$ if $\tau \equiv \sigma$ and $v_k \rightarrow v_l \tau$ if $\tau \equiv \sigma$.

**Definition 4.3:** The state transition cost of the LCFG $\Gamma$ is defined as $\pi : V \times V \to \Upsilon$ such that $\forall v_k, v_l \in V$,

\[
\pi[v_k, v_l] = \sum_{\sigma_j \in \Sigma, \sigma_j \notin P} \Re(\tilde{\sigma}(\sigma_j, v_k)) + \sum_{\sigma_j \in \Sigma, \sigma_j \notin P} \Im(\tilde{\sigma}(\sigma_j, v_k))
\]

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

\[
\Pi = \begin{bmatrix}
\pi_{11} & \ldots & \pi_{1n} \\
\vdots & \ddots & \vdots \\
\pi_{n1} & \ldots & \pi_{nn}
\end{bmatrix}
\]

**Definition 4.4:** Let $\Gamma$ be a linear grammar, initialized at a state $v_k \in V$. The complex measure $\mu$ of every singleton path set $\Lambda = \{\lambda\} \in 2^{P \Gamma}$ is defined as $\mu(\Lambda) = \tilde{\sigma}(\lambda, v_k) \star \chi(v)$. The complex measure $\mu$ of every singleton string set $\Omega = \{\omega\} \in 2^{(P \Gamma)}$ is defined as $\mu(\Omega) = \sum_{\epsilon \in \Omega} \tilde{\sigma}(\lambda, v_k) \star \chi(v)$. Then, the complex measure of the sublanguage $L(v_k, v) \subseteq L(\Gamma)$ of all strings terminated at the state $v \in V$ is defined as:

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

Complex measure of the language of the linear grammar $\Gamma_k$ is defined as:

\[
\mu_k = \mu(L(\Gamma_k)) = \sum_{i \in \Sigma} \mu(L(v_k, v))
\]

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

**B. Computation of the $\mu$-Vector**

This subsection presents a procedure to formulate the complex measure of $P_{\Gamma}$, and by Lemma 4.1 it coincides with the complex measure for $L(\Gamma_k)$. Now,

\[
P_{\Gamma_k} = (\cup_{\sigma_j \neq P} P_{\Gamma_j}) \cup_k \epsilon_k
\]

where the null event $\epsilon_k$ is defined as:

\[
\epsilon_k = \begin{cases} 
\epsilon, & \text{ if self loop at } v_i \\
\emptyset, & \text{ otherwise}
\end{cases}
\]

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 $\sigma_j$ and the current state changes to $v_j$, then the set of all paths thereafter is exactly equal to $P_{\Gamma_j}$. The expression is structurally identical to that given for $DFSA$ in [7] with the understanding that the event $\sigma_j$ can be either real or imaginary.

Hence,

\[
\mu(P_{\Gamma_k}) = \mu \left( (\cup_{\sigma_j \neq P} P_{\Gamma_j}) \cup_k \epsilon_k \right)
\]

\[
= \mu(\cup_{\sigma_j \neq P} P_{\Gamma_j}) + \mu(\epsilon_k)
\]

\[
= \sum_{j} \mu(\sigma_j \neq P_{\Gamma_j}) + \chi(v_k)
\]

\[
= \sum_{j} \pi_{ij} \star \mu(P_{\Gamma_j}) + \chi(v_k)
\]

(25)

The first three steps in Eq. (25) follow from the fact that if the first symbol for two paths is different, then the paths cannot be identical. However, the generated strings may still be the same.

The fourth step follows from property (6) of the $\tilde{\sigma}$ function in Definition 4.2. The final step trivially follows from Definition 4.4 of the measure. In vector form, the complex measure $\mu$ is given by:

\[
\mu(L(\Gamma_k)) = \Pi \star X + (I - \Pi)^{-1} \star X
\]

\[
= \left( (I - \Re(\Pi)) + i(I - \Im(\Pi)) \right)^{-1} \star X
\]

\[
= (I - \Re(\Pi))^{-1} \Re(\Pi) X + i(I - \Im(\Pi))^{-1} \Im(\Pi) X
\]

where $\Re$ and $\Im$ refer to the real and imaginary parts of the matrices, respectively; and existence of the matrix inverses is guaranteed by the following conditions:

\[
\sum_j \Re(\tilde{\sigma}(\omega)) < 1; \quad \sum_j \Im(\tilde{\sigma}(\omega)) < 1 \forall k.
\]
A simple example is presented below to illustrate how the complex $\mu$ is computed for an LCFG.

C. Example

Let a language $L$ generate all strings of the type $\{a^nb^n : n \geq 0\}$ over the alphabet $\Sigma = \{a,b\}$. The non-regular language $L$ can be generated by the grammar $\{v \rightarrow avb|e\}$ that can be rewritten as: $v_1 \rightarrow av_2$; and $v_2 \rightarrow v_1b$. The resulting $\tilde{\Pi}$ matrix is expressed in the matrix form as:

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

where the parameters $p$ and $q$ can be identified from the experimental time series data of the system dynamics [8]. The $\Pi$-matrix is then obtained as:

$$\text{Re}\Pi = \begin{bmatrix} 0 & p \\ 0 & 0 \end{bmatrix} \text{ and } \text{Im}\Pi = \begin{bmatrix} 0 & 0 \\ q & 0 \end{bmatrix}.$$ 

Assigning characteristic values (i.e., weights) of the two states $v_1$ and $v_2$ to be $\chi_1$ and $\chi_2$ respectively, the complex measure vector is evaluated as:

$$\mu(L) = \{ (\chi_1 + px_2) + iq\chi_1 \\ \chi_2 + iq(\chi_2 + q\chi_1) \}$$

V. Future Work

Future research will focus two main directions. Further development of the notion of the event plane will help investigating important geometric properties of formal languages on which little work is reported. Relating formal language theory to the notions of algebraic topology and geometry will have far reaching implications. A deformation retract of the event plane followed by an identification of the edges described by the derivations lead to a very interesting geometric interpretation. Figures 3 and 4 illustrate the idea. The first step in Figure 3 is to continuously deform the event plane to collapse it along the dotted lines (a deformation retract [1]). The succeeding steps are identification of edges marked by identical paths on the plane. The result is a compact topological space, namely the double wedge of spheres [1]. In Figure 4, the same construction is carried out for the regular grammar of Figure 2 which results in a double wedge of circles. It immediately follows that even for a general regular grammar, such a quotient space construction always yields a $n$-wedge of circles (Figure 4), whereas for a non-regular linear grammar, the geometric object generated is more complicated (e.g. in Figure 3, it is a 3-dimensional object). Future work will investigate the relationship between the computational properties of linear grammars and the geometric properties of the associated quotient spaces.

The other direction is further extension of the notion of measure to a general context free language. One possible approach is to generalize the concept of the event plane to that of the event sheaf. To handle a general CFG, one must deal with production rules of the type $S \rightarrow aAB$ is to consider a sheaf of event planes and lift the second variable $B$ to the next higher plane. The idea is illustrated in Figure 5 where a grammar $\{S \rightarrow aTV \mid aTVW; V \rightarrow TbS; \cdots\}$ is considered. Future work will focus on such event sheaf representations of general context free grammars and the possibility of generalizing the results of this paper to the entire class of context free languages.

VI. Summary and Conclusions

This paper introduces the notion of a quantitative measure of non-regular languages [2], generated by linear context free grammars (LCFG) that belong to the low end of Chomsky hierarchy [3]. It shows that the measure of regular languages, reported in [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.

References

Fig. 3. Quotient space construction for the grammar of Figure 1. The quotient space is homotopic in the natural sense to wedge of spheres.

Fig. 4. Corresponding construction for a strictly regular grammar. The quotient space is a double wedge of circles.

Fig. 5. Suggested event sheaf for a non-linear grammar (only a few derivations shown).