INVITED PAPER

Syntactic Modeling and Signal Processing of Multifunction Radars: A Stochastic Context-Free Grammar Approach

Using improved pattern recognition techniques, the target of a radar system can model and identify that system and estimate how much of a threat it poses.

By Nikita Visnevski, *Member IEEE*, Vikram Krishnamurthy, *Fellow IEEE*, Alex Wang, and Simon Haykin, *Fellow IEEE*

ABSTRACT | Multifunction radars (MFRs) are sophisticated sensors with complex dynamical modes that are widely used in surveillance and tracking. This paper demonstrates that stochastic context-free grammars (SCFGs) are adequate models for capturing the essential features of the MFR dynamics. Specifically, MFRs are modeled as systems that "speak" a language that is characterized by an SCFG. The paper shows that such a grammar is modulated by a Markov chain representing radar's policy of operation. The paper also demonstrates how some well-known statistical signal processing techniques can be applied to MFR signal processing using these stochstic syntactic models. We derive two statistical estimation approaches for MFR signal processing—a maximum likelihood sequence estimator to estimate radar's policies of operation, and a maximum likelihood parameter estimator to infer the radar parameter values. Two layers of signal processing are introduced in this paper. The first layer is concerned with the estimation of MFR's policies of operation. It involves signal processing in the CFG domain. The second layer is concerned with identification of tasks the radar is engaged in. It involves signal processing in the finite-state domain. Both of these signal processing techniques are important elements of a bigger radar signal processing problem that is often encountered in electronic warfare applications—the problem of the estimation of the level of threat that a radar poses to each individual target at any point in time.

KEYWORDS | Electronic warfare; Galton–Watson branching process; inside-outside algorithm; maximum likelihood estimation; multifunction radar (MFR); stochastic context-free grammars (SCFGs); syntactic modeling; syntactic pattern recognition

I. INTRODUCTION

Statistical pattern recognition has been a major tool used in building electronic warfare (EW) systems to analyze radar signals. Conventional radars have been historically characterized by fixed parameters such as radio frequency, pulse-width, and peak amplitude [1], [2]. For these radar characterizations, parametric models are sufficient for solving signal processing problems such as emitter identification and threat evaluation. References [3] and [4] discuss template matching of the intercepted radar signal against an EW library for both the emitter type and emitter mode identification. Histogram techniques are described in [5] to study the temporal periodicities in radar signals such as pulse repetition intervals.

Manuscript received October 1, 2006; revised January 24, 2007.

N. Visnevski is with the General Electric Company, Global Research Center, Niskayuna, NY 12309 USA (e-mail: nikita.visnevski@research.ge.com).

V. Krishnamurthy and **A. Wang** are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T1Z4, Canada (e-mail: vikramk@ece.ubc.ca; alexw@ece.ubc.ca).

S. Haykin is with the Adaptive Systems Laboratory, Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: haykin@mcmaster.ca).

Digital Object Identifier: 10.1109/JPROC.2007.893252

With the advent of modern radars, especially multifunction radars (MFRs), statistical pattern recognition approaches described above became inadequate. MFRs are radio-frequency sensors that are widely used in modern surveillance and tracking systems, and they have the capability to perform a multitude of different tasks simultaneously. The list of these tasks often includes such activities as search, acquisition, multiple target tracking, and weapon guidance [6]. MFRs use electronic beam-steering antennas to perform multiple tasks simultaneously by multiplexing them in time using short time slices [7]. At the same time they have to maintain low probability of being detected and jammed. Indeed, MFRs are an excellent example of highly complex man-made large-scale dynamical systems. MFRs' ability to adaptively and actively switch modes and change system parameters greatly limits the applicability of the parametric statistical pattern recognition approaches. The dimensionality of the operational state space for such radars is too large for the statistical approach to be viable.

This paper proposes a different approach to radar modeling and radar signal processing—one based on syntactic pattern recognition. The origins of syntactic modeling can be traced to the classic works of Noam Chomsky on formal languages and transformational grammars [8]–[11]. The central elements of this work are the concepts of a *formal language* and its *grammar*. Languages are typically infinite sets of strings drawn from a finite alphabet of symbols. Grammars, on the other hand, are viewed as finite-dimensional models of languages that completely characterize them.

Many different kinds of grammars and languages have been identified and investigated for practical applications. Among them, the *finite-state grammars* (FSGs) and the *context-free grammars* (CFGs), as well as their stochastic counterparts, are currently the most widely used classes of grammars. *Stochastic finite-state grammars* (SFSGs), also known as hidden Markov models, achieved a great success in the speech community [12], [13]. They were used in modern tracking systems [14] and in machine vision [15]. On the other hand, *stochastic context-free grammars* (SCFGs) are studied in [16] for gesture recognition and the implementation of an online parking lot monitoring task. In [17] and [18] they were used in modeling the dynamics of a bursty wireless communications channel. References [19] and [20] describe syntactic modeling applied to bioinformatics, and [21] and [22] apply these models to the study of biological sequence analysis and RNA. Finally, application of syntactic modeling to pattern recognition is covered in depth in [23].

In this paper, we construct a Markov-modulated SCFG to model an anti-aircraft defense MFR called Mercury. The more traditional approaches such as hidden Markov and state space models are suitable for target modeling [14], [24] but not radar modeling because MFRs are large-scale dynamical systems and their scheduling involves planning and preempting that makes state space approach difficult. In addition to radar modeling, we also consider statistical radar signal processing. The proposed linguistic model of MFRs is naturally divided into two levels of abstraction: task scheduling level and radar control level. We show that the MFRs' SCFG representation at task scheduling level is self-embedding and cannot be reduced to a finite-state form. Thus, signal processing has to be performed in the CFG domain. The MFRs' SCFG representation at the radar control level, on the other hand, is non-self-embedding (NSE). Thus, a finite-state model for such a grammar is obtainable. Finally, we introduce a systematic approach to convert the radar control level SCFG representation to its finite-state counterpart.

The reason for such a two-level approach to radar modeling is that of computational cost. Although SCFGs provide a compact representation for a complex system such as MFRs, they are associated with computationally intensive signal processing algorithms [21], [23], [25], [26]. By contrast, finite-state representations are not nearly as compact (the number of states in the finite-state automaton representing an MFR can be very large [27]), but the associated signal processing techniques are well-known and much less computationally demanding (see discussion on complexity of syntactic parsing algorithms in [21]). It is therefore advantageous to model MFRs as SCFGs, and perform signal processing on their finite-state equivalents as much as possible.

Traditionally, MFRs' signal modes were represented by volumes of parameterized data records known as Electronic Intelligence (ELINT) [1]. The data records are annotated by lines of text explaining when, why, and how a signal may change from one mode to another. This makes radar mode estimation and threat evaluation fairly difficult. In [28], SCFG is introduced as a framework to model MFRs' signal and it is shown that MFRs' dynamic behavior can be explicitly described using a finite set of rules corresponding to the production rules of the SCFG. SCFG has several potential advantages.

1) SCFG is a compact formal representation that can form a homogeneous basis for modeling complex system dynamics [23], [25], [26], and with which it allows model designers to express different aspects of MFR control rules in a single framework [28], and automates the threat estimation process by encoding human knowledge in the grammar [29], [30].
2) The recursive embedding structure of MFRs' control rules is more naturally modeled in SCFG. As we show later, the Markovian type model has dependency that has variable length, and the growing state space is difficult to handle since the maximum range dependency must be considered.
3) SCFGs are more efficient in modeling hidden branching processes when compared to stochastic regular grammars or hidden Markov models with

the same number of parameters. The predictive power of an SCFG measured in terms of entropy is greater than that of the stochastic regular grammar [31]. SCFG is equivalent to a multitype Galton–Watson branching process with finite number of rewrite rules, and its entropy calculation is discussed in [32].

In summary, the main results of the paper are as follows.

1) A careful detailed model of the dynamics of an MFR using formal language production rules. By modeling the MFR dynamics using a linguistic formalism such as an SCFG, an MFR can be viewed as a discrete event system that "speaks" some known, or partially known, formal language [33]. Observations of radar emissions can be viewed as strings from this language, corrupted by the noise in the observation environment.
2) Formal procedure of synthesis of stochastic automaton models from the compact syntactic rules of CFG. Under the condition that the CFG is NSE, the CFG representation can be converted to its finite-state counterpart, where the signal processing is computationally inexpensive.
3) Novel use of Markov modulated SCFGs to model radar emissions generated by MFR. The complex embedding structure of the radar signal is captured by the linguistic model, SCFG, and the MFR's internal state, its policies of operation, is modeled by a Markov chain. This modeling approach enables the combination of the grammar's syntactic modeling power with the rich theory of Markov decision process.
4) Statistical signal processing of SCFGs. The threat evaluation problem is reduced to a state estimation problem of HMM. The maximum likelihood estimator is derived based on a hybrid of the forward-backward and the inside-outside algorithm. (Inside-outside algorithm is an extension of HMM's forward-backward algorithm [34].)

The rest of the paper is organized as follows. Section II provides a self-contained theoretical background of syntactic modeling methodology. Section III describes the MFR in detail and its role in electronic warfare. Section IV and Section V present the threat estimation algorithms and a detailed description of the synthesis procedure of stochastic automaton models from the syntactic rules of CFG. Finally, Section VI concludes the paper.

II. ELEMENTS OF SYNTACTIC MODELING

This section presents important elements from the theory of syntactic modeling, syntactic pattern recognition, and syntax analysis. The aim is to provide the reader with a brief overview of the concepts that will be used in the rest of the paper, and a more comprehensive discussion can be found in [23]. We use the definitions and notations common to the theory of formal languages and computational linguistics [25], [26].

We will start by introducing the concept of *formal languages*. These languages are most accurately defined in the set-theoretic terms as collections of strings having a certain predefined structure. In practice, a finite-dimensional model of the language is required, and it should help answering the two fundamental questions of the theory of formal languages.

- Given a language, how can we derive any string from this language? (The problem of string *generation*.)
- Given a certain string and a language, how can we tell if this string is part of this language? (The problem of string *parsing* or *syntax analysis*.)

The finite-dimensional models of languages that help answering these fundamental questions are called *grammars*. If we focus on the problem of string generation, such grammars are typically called *generative grammars*. If, on the other hand, we are interested in string parsing, it is customary to refer to the language grammars as *transformational grammars*.

1) Formal Languages: Let $\mathcal{A}$ be an arbitrary set of symbols that we will call an *alphabet*. In general, an alphabet does not have to be finite, but from the practical standpoint we will assume that $\mathcal{A}$ is a finite set of symbols.

Using symbols from $\mathcal{A}$, one can construct an infinite number of strings by *concatenating* them together. We call an ε-string an *empty string*—a string consisting of no symbols. Let us denote by $\mathcal{A}^+$ an infinite set of *all* finite strings formed by concatenation of symbols from $\mathcal{A}$, and let us denote by $\mathcal{A}^* = \mathcal{A}^+ \cup \varepsilon$. For example, if $\mathcal{A} = \{a, b, c\}$, then

$$\mathcal{A}^+ = \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots\} \quad (1)$$

$$\mathcal{A}^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots\}. \quad (2)$$

The $\mathcal{A}^+$ operation is called *positive (transitive) closure* of $\mathcal{A}$, and the $\mathcal{A}^*$ operation is called *Kleene (reflexive and transitive) closure*.

Definition 2.1: The *language* L defined over an alphabet $\mathcal{A}$ is a set of some finite-length strings formed by concatenating symbols from $\mathcal{A}$.

Evidently, $L \subseteq \mathcal{A}^*$, and in particular, $\emptyset$, $\mathcal{A}$, and $\mathcal{A}^*$ are also languages.

2) Grammars: The definition of the formal language (Def. 2.1) is extremely broad and therefore, has very limited practical application. A more useful way of defining formal languages is through the use of *grammars* [8]–[11].

Definition 2.2: A *deterministic grammar* G is a four-tuple

$$G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0) \tag{3}$$

where:

$\mathcal{A}$ is the alphabet (the set of terminal symbols of the grammar);

$\mathcal{E}$ is the set of nonterminal symbols of the grammar;

Γ is the finite set of grammatical production rules (syntactic rules);

S_0 is the starting nonterminal.

In general, Γ is a partially defined function of type

$$\Gamma : (\mathcal{A} \cup \mathcal{E})^* \rightarrow (\mathcal{A} \cup \mathcal{E})^*. \tag{4}$$

However, as we will see later, certain restrictions applied to the production rules Γ allow us to define some very useful types of grammars.

In the rest of this paper, unless specified otherwise, we will write nonterminal symbols as capital letters, and symbols of the alphabet using lower case letters. This follows the default convention of the theory of formal languages.

Def. 2.1 provides a set-theoretic definition of a formal language. Now, using Def. 2.2 we can redefine the language in terms of its grammar $L \triangleq L(G)$.

To illustrate the use of grammars, consider a simple language $L = L(G)$ whose grammar $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is defined as follows:

$$\begin{aligned} \mathcal{A} &= \{a, b\} \quad S_0 \rightarrow aS_1|b \\ \mathcal{E} &= \{S_0, S_1\} \quad S_1 \rightarrow bS_0|a. \end{aligned} \tag{5}$$

These are some of the valid strings in this language, and examples of how they can be derived by repeated application of the production rules of (5):

1) $S_0 \Rightarrow b$;
2) $S_0 \Rightarrow aS_1 \Rightarrow aa$;
3) $S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abb$;
4) $S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow abaa$;
5) $S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow ababS_0 \Rightarrow ababb$;
6) $S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow ababS_0 \Rightarrow \ldots \Rightarrow ababab \ldots abb$;
7) $S_0 \Rightarrow aS_1 \Rightarrow abS_0 \Rightarrow abaS_1 \Rightarrow ababS_0 \Rightarrow \ldots \Rightarrow ababab \ldots abaa$.

This language contains an infinite number of strings that can be of arbitrary length. The strings start with either a or b. If a string starts with b, then it only contains one symbol. Strings terminate with either aa or bb, and consist of a distinct repeating pattern ab.

This simple example illustrates the power of the grammatical representation of languages. Very simple grammars can define rather sophisticated languages.

3) Chomsky Hierarchy of Grammars: In Def. 2.2, the production rules of the grammar are given in a very general form. Reference [10] used the properties of the production rules of grammars to develop a very useful hierarchy that is known in the literature as the Chomsky hierarchy of grammars.

- **Regular Grammars (RG):** Only production rules of the form $S \rightarrow aS$ or $S \rightarrow a$ are allowed. This means that the left-hand side of the production must contain one nonterminal only, and the right-hand side could be either one terminal or one terminal followed by one nonterminal. The grammar of the language in the last example of this section is a regular grammar. Regular grammars are sometimes referred to as *finite-state grammars*.
- **CFGs:** Any production rule of the form $S \rightarrow \beta$ is allowed. This means that the left-hand side of the production rule must contain one nonterminal only, whereas the right-hand side can be any string.
- **Context-Sensitive Grammars (CSG):** Production rules of the form $\alpha_1 S \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$ are allowed. Here $\alpha_1, \alpha_2 \in (\mathcal{A} \cup \mathcal{E})^*$, and $\beta \neq \varepsilon$. In other words, the allowed transformations of nonterminal S are dependent on its context α_1 and α_2.
- **Unrestricted Grammars (UG):** Any production rules of the form $\alpha_1 S \alpha_2 \rightarrow \gamma$ are allowed. Here α_1, α_2, $\gamma \in (\mathcal{A} \cup \mathcal{E})^*$. The unrestricted grammars are also often referred to as *type-0 grammars* due to Chomsky [10].

Chomsky also classified languages in terms of the grammars that can be used to define them. Fig. 1 illustrates this hierarchy of languages. Each inner circle of this diagram is a subset of the outer circle. Thus, context-sensitive language (CSL) is a special (more restricted) form of unrestricted language (UL), context-free language (CFL) is a special case of CSL, and regular language (RL) is a

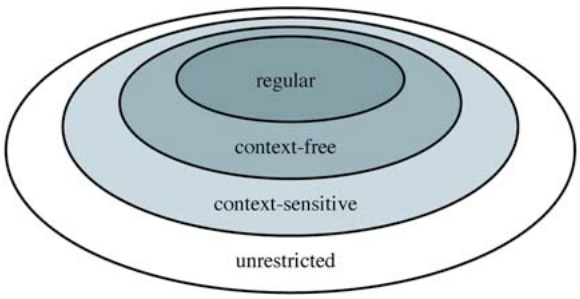


Fig. 1. ***The Chomsky hierarchy of formal languages.***

Table 1 Deterministic Grammars, Production Rules, and Languages

Grammar	Production rule structure	Language
FSG	$S \rightarrow aS$ $S \rightarrow a$	Finite State (Regular) Language (RL)
CFG	$S \rightarrow \beta$	Context-Free Language (CFL)
CSG	$\alpha_1 S \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$	Context-Sensitive Language (CSL)
UG	$\alpha_1 S \alpha_2 \rightarrow \gamma$	Unrestricted (type-0) Language (UL)

special case of CFL. Table 1 provides a condensed summary of the classes of grammars, their production rule structures, and classes of languages that they define. More detailed treatment of the Chomsky hierarchy is given by [21].

Syntactic modeling of DES (discrete event system) developed in this paper will make extensive use of FSG and CFG. CSG and UG will not be used in our modeling approach.

4) Relationship Between Regular Languages and Finite-State Automata:

Definition 2.3: A *finite-state automaton* (FSA) Λ is a five-tuple

$$\Lambda = (Q, \Sigma, \delta, q_0, F) \tag{6}$$

where:

Q is the set of states of the FSA;
Σ is the set of input symbols of the FSA;
δ is the transition function of the FSA;
q_0 is the initial state of the FSA;
F is the set of final (accepting) states of the FSA ($F \subset Q$).

FSAs were shown to be equivalent to RG and RL (see [25], [26], and [35]–[38]). In fact, using Def. 2.2 and Def. 2.3 we can observe that if $Q = \mathcal{E}$, $\Sigma = \mathcal{A}$, and $q_0 = S_0$, we can relate δ and Γ in such a way that $L(\Lambda) = L(G)$. $L(\Lambda)$ is also called the language accepted by the FSA Λ. The set of final (accepting) states F is the set of states such that any input string from $L(\Lambda)$ causes Λ to transition into one of these states. An FSA equivalent of the grammar (5) is shown in Fig. 2.

5) Context-Free Languages and CFGs: The next, less restricted member of the Chomsky hierarchy of grammars is the *CFG*. Languages that can be accepted by FSAs are limited in terms of strings that they can contain. The most famous example of a language that cannot be accepted by FSAs is the language of *palindromes*.[1] It was shown to be a CFL [26]. A simple language of palindromes can, for example, be defined by the following set of production rules:

$$P \rightarrow bPb|aPa|b|a|\varepsilon \tag{7}$$

and an example string from this language is *bababaaababab*. According to Table 1, the grammar in (7) is a CFG.

CFGs are often associated with tree-like graphs instead of FSAs since the dependency between the elements of the strings of the CFL are nested [25], [26], [35], [39], [40]. Due to this fact, the task of processing the strings from a CFL is a more computationally complex procedure than that of an RL. On the other hand, [41] have shown that CFG could be more compact descriptions of the RL than RG. It is often convenient to describe complex finite-state systems in the context-free form, but it is less computationally intensive to perform analysis of these systems using FSA. This fact is at the center of the large scale DES modeling methodology that we are going to develop in the rest of this section.

As Fig. 1 clearly demonstrates, RL are a proper subset of the class of CFL. However, given a general CFG, one cannot tell if this grammar describes an RL or a CFL (this task was shown to be undecidable [40]). We will now look at the property of *self-embedding* of the CFG and see how this property helps in determining the class of the languages described by such CFG.

6) Non-Self-Embedding CFGs:

Definition 2.4: A CFG $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is *self-embedding* if there exists a nonterminal symbol $A \in \mathcal{E}$ such that a string $\alpha A \beta$ can be derived from it in a finite number of derivation steps, with α, $\beta \neq \varepsilon$ being any string of terminal and nonterminal symbols.

For example, the nonterminal symbol P in the palindrome grammar (7) is such a self-embedding nonterminal, and the CFG of palindromes is self-embedding.

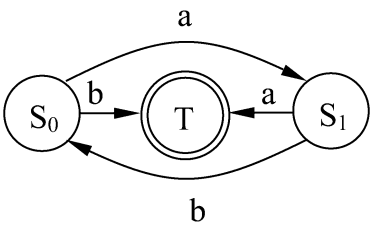


Fig. 2. *FSA equivalent to the grammar example (5). State S_0 is the starting state, and T is an accepting state, as indicated by the double circle.*

[1]A *palindrome* is a string that reads the same way both left-to-right and right-to-left.

Definition 2.5: A CFG $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is *non-self-embedding* if there exists no nonterminal symbols for which the condition of the Def. 2.4 can be satisfied.

[11] has demonstrated that if a CFG is NSE, it generates a finite-state language (FSL). In Section V-A, we will describe an algorithm to verify the NSE property of CFGs, and show how to obtain FSAs for these grammars.

7) Stochastic Languages and Stochastic Grammars: A number of practical applications contain certain amounts of uncertainty that are often represented by probabilistic distributions. These factors require extension of the concepts described above into the domain of stochastic languages.

Definition 2.6: A *weighted grammar* G_w is a five-tuple

$$G_w = (\mathcal{A}, \mathcal{E}, \Gamma, P_w, S_0) \tag{8}$$

where:

- $\mathcal{A}$ is the alphabet (the set of terminal symbols of the grammar);
- $\mathcal{E}$ is the set of nonterminal symbols of the grammar;
- Γ is the finite set of grammatical production rules (syntactic rules);
- P_w is the set of weighting coefficients defined over the production rules Γ;
- S_0 is the staring nonterminal.

Here is a simple example of a weighted grammar:

$$\begin{aligned} S_0 &\xrightarrow{9} aS_1 \\ S_0 &\xrightarrow{1} b \\ S_1 &\xrightarrow{1} bS_0 \\ S_1 &\xrightarrow{9} a. \end{aligned} \tag{9}$$

This grammar has been obtained from grammar (5) by associating with its productions the set of weights $P_w = \{(9, 1), (1, 9)\}$. Note that the set of weights P_w does not have to be normalized.

Definition 2.7: A *stochastic grammar* G_s is a five-tuple

$$G_s = (\mathcal{A}, \mathcal{E}, \Gamma, P_s, S_0) \tag{10}$$

where $\mathcal{A}$, $\mathcal{E}$, Γ, and S_0 are the same as in Def. 2.6, and P_s is the set of probability distributions over the set of production rules Γ.

Clearly, stochastic grammars are simply a more restricted case of the weighted grammars. Here is a simple example of a stochastic grammar:

$$\begin{aligned} S_0 &\xrightarrow{0.9} aS_1 \\ S_0 &\xrightarrow{0.1} b \\ S_1 &\xrightarrow{0.1} bS_0 \\ S_1 &\xrightarrow{0.9} a. \end{aligned} \tag{11}$$

This grammar has been obtained from grammar (5) by applying to its productions the probability distributions $P_s = \{(0.9, 0.1), (0.1, 0.9)\}$.

Stochastic and weighted grammars are classified and analyzed on the basis of their underlying *characteristic grammars* [23], [42]. A characteristic grammar G_c is obtained from the stochastic grammar G_s (weighted grammar G_w) by removing the probability distribution P_s (set of weights P_w) from the grammar definition.

If the resulting characteristic grammar is an FSG, the stochastic grammar is called stochastic finite-state grammar (SFSG). If the characteristic grammar is a CFG, the stochastic grammar is referred to as an SCFG. For example, grammar (11) is an SFSG, and grammar (5) is its characteristic grammar.

Characteristic grammars play important roles in deriving syntactic models of real-life systems. The typical procedure is illustrated in Fig. 3. The characteristic grammar is a bridge between the internal deterministic rules of the system, and the stochastic environment in which this system is operating or observed.

8) Stochastic Finite-State Languages, Markov Chains, and Hidden Markov Models: Just as FSAs constitute one of the

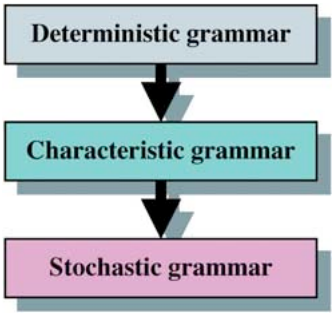


Fig. 3. *Derivation procedure for the stochastic grammars. First, a deterministic grammar for the system is constructed. Then, after considerations of possible sources of uncertainties, the deterministic grammar is modified into a characteristic grammar that accommodates for these uncertainties. Finally, a probability distribution is assigned to the characteristic grammar, yielding a stochastic grammar of the system.*

representation forms of FSLs, discrete-state discrete-time Markov chains are naturally considered the equivalent representations of the SFSLs [33]. This representation has been successfully utilized in bioinformatics and computational genomics [19], [21] as well as in natural language and speech processing [12].

A discrete-state discrete-time Markov chain can be defined as a stochastic timed automaton [33]:

Definition 2.8: A *discrete-time Markov chain* defined over a discrete state space is a tuple

$$\gamma = (\mathbf{A}, \pi) \tag{12}$$

where:

$\mathbf{A}$ is the $N \times N$ state transition probability matrix,
π is the $N \times 1$ vector of initial state probability distribution, and
N is the number of states in the Markov chain.

We will illustrate the relationship between SFSGs and Markov chains by looking at the transition structure of the grammar (11). We can construct a Markov chain that will reflect the transitions within Γ of (11) as

$$\gamma = \left(\mathbf{A} = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0.9 \\ 0 & 0 & 0 \end{bmatrix}, \pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \tag{13}$$

where $\mathbf{A}$ and π are defined with respect to the state ordering $\{S_0, S_1, T\}$ as shown in Fig. 4.

The example above illustrates a strong parallel between FSAs in the case of deterministic grammars, and Markov chains in the case of stochastic ones. However, Markov chains defined by Def. 2.8 do not accommodate for the alphabet $\mathcal{A}$ of the grammar. Therefore, Markov chains can only capture transition dynamics of the grammar, but do not address generation and transformation aspects of the SFSGs discussed earlier. Hidden Markov models (HMMs) address this issue.

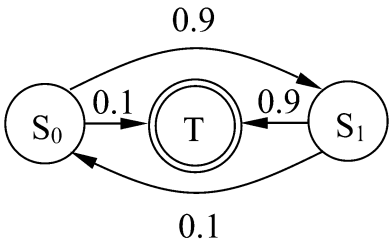


Fig. 4. ***Example of a Markov chain for the SFSG (11). Note that Markov chains only capture the transition dynamics of the grammar since the terminal symbols of the grammar do not feature in the Markov chain structure.***

HMMs [12], [13], [19] are particularly suitable for representing stochastic languages of the finite-state discrete-event systems observed in noisy environments. They separate the uncertainty in the model attributed to the observation process from the uncertainties associated with the system's functionality. Generally speaking, HMMs are Markov chains indirectly observed through a noisy process [13], [33], [43]–[45].

Definition 2.9: A *HMM* λ is a three-tuple

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi) \tag{14}$$

where:

$\mathbf{A}$ is the $N \times N$ state transition probability matrix of the underlying Markov chain,
$\mathbf{B}$ is the $N \times M$ observation probability matrix that establishes probability distributions of observing certain discrete symbols associated with a certain state of the chain,
π is the $N \times 1$ vector of initial state probability distribution of the underlying Markov chain,
N is the number of states of the underlying Markov chain, and
M is the number of possible discrete observation symbols.

To illustrate how HMMs relate to SFSGs, we would like to revisit the grammar (11). The Markov chain for this grammar is defined by (13). Now we can extend this chain bringing in the alphabet $\mathcal{A}$ of the grammar (11) through the structure of the observation probability matrix $\mathbf{B}$.

However, this extension requires a transformation of the structure of the Markov chain in Fig. 4. Def. 2.9 associates the observation probability matrix $\mathbf{B}$ with the states of the chain, whereas SFSGs associate generation of nonterminals with transitions of the state machine. The former case is known in the literature as the *Moore machine*, and the latter is referred to as the *Mealy machine* [26]. Therefore, to accommodate for the structural constraints of the HMM, the Markov chain in Fig. 4 has to be converted to the Moore machine form as described in detail in [26].

The resulting HMM has the following structure:

$$\lambda = \left(\mathbf{A} = \begin{bmatrix} 0 & 0.1 & 0.9 & 0 \\ 0.9 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \pi = \begin{bmatrix} 0.9 \\ 0 \\ 0 \\ 0.1 \end{bmatrix} \right) \tag{15}$$

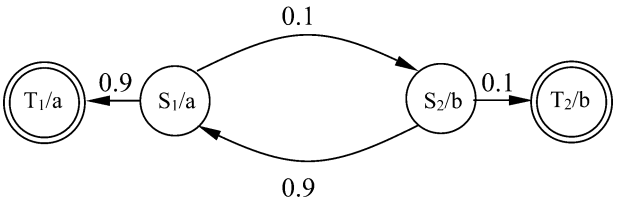


Fig. 5. *Example of an HMM for the SFSG (11). Each state is labeled by two symbols separated by a slash. The first symbol identifies the state of the system, and the second determines the output produced by the system in this state. To accommodate for the terminal symbols of the grammar (11) through the use of the observation probability matrix* B*, the structure of the Markov chain in Fig. 4 had to be transformed to the Moore machine. Consequently, the underlying Markov chain of this HMM has different set of discrete states* $\{S_1, S_2, T_1, T_2\}$.

where $\mathbf{A}$ as well as rows of π and $\mathbf{B}$ are defined with respect to the state ordering $\{S_1, S_2, T_1, T_2\}$ as shown in Fig. 5, and columns of $\mathbf{B}$ are defined with respect to the ordering $\{a, b\}$.

III. ELECTRONIC WARFARE APPLICATION—ELECTRONIC SUPPORT AND MFR

With the above background in syntactic modeling, we are now ready to study MFRs, and devise electronic support algorithms that deal with their ever increasing sophistication of their remote sensing capabilities.

Electronic warfare (EW) can be broadly defined as any military action with the objective of controlling the electromagnetic spectrum [46]. An important aspect of EW is the radar-target interaction. In general, this interaction can be examined from two entirely different points of view—the viewpoint of the radar and the viewpoint of the target. From the radar's point of view, its primary goal is to detect targets and to identify their critical parameters. From the target's point of view, the goal is to protect itself from a radar-equipped threat by collecting radar emissions and evaluating threat in real time (electronic support). In this paper, the target's viewpoint is the focus, and MFRs are the specific threat considered.

The framework of EW considered in this paper consists of three layers: receiver/deinterleaver, pulse train analyzer, and syntactic processor [47]. The layers are depicted in Fig. 6 and a brief description is given here: The receiver processes the radar pulses intercepted by the antenna, and outputs a sequence of pulse descriptor words, which is a data structure containing parameters such as carrier frequency, pulse amplitude, or pulse width. The deinterleaver processes the pulse descriptor words, groups them according to their possible originating radar emitters and stores them in their corresponding track files. The pulse train analyzer processes the track file, and further groups the pulse descriptor words into radar words. (See Section III-A for definitions.) Finally, the syntactic processor analyzes the syntactic structure of the radar words, estimates the state of the radar system and its threat level, and outputs the results on a pilot instrumentation panel. Because the receiver, deinterleaver, and pulse train analyzer have been well studied, the syntactic processor is the focus of this paper.

The syntactic processor captures the knowledge of the "language" that MFRs speak. It is a complex system of rules and constraints that allow radar analysts to distinguish "grammatical" radar signal from "ungrammatical" one. In other words, an analogy is drawn between the structural description of the radar signal and the syntax of a language, and the structural description could, therefore, be specified by the establishment of a grammar [48]. As far as EW is concerned, the optimal approach is to collect a corpus of radar samples, and induce the grammar directly without human intervention. However, because of the degree of complexity and potential lack of data on the MFR signal, grammatical induction approach is impractical. As a result, in this paper, the grammar is constrained to be SCFG, and its context-free backbone is specified by radar analysts from studying MFRs' signal generation mechanism. Section III-A describes MFRs' system architecture and the building blocks making up the radar signal, and Section III-B discusses the shortcomings of HMM and explains why SCFG is preferred.

A. MFR Signal Model and Its System Architecture

In order to describe MFRs' system architecture, we begin with the building blocks making up MFRs' signal generation process, and they are defined as follows.

- *Radar word*: A fixed arrangement of finite number of pulses that is optimized for extracting a particular target information; for example, pulses with a fixed pulse repetition frequency.

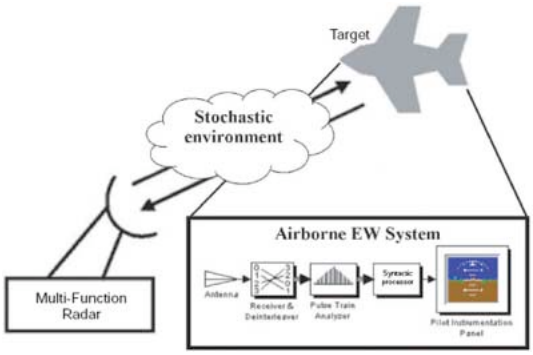


Fig. 6. *The electronic warfare (EW) framework considered in this paper. The radar signal emitted by the MFR is captured at the EW system on board the target after being corrupted by the stochastic environment. The EW system consists of an antenna, a receiver/deinterleaver, a pulse train analyzer, and a syntactic processor.*

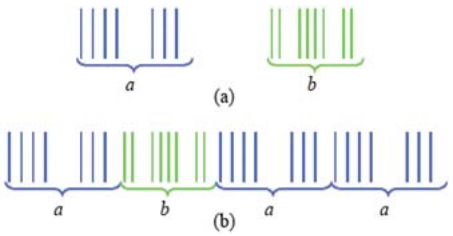


Fig. 7. ***Radar words can be viewed as fundamental building blocks of the MFR signal. (a) Shows two distinct radar words labeled as a and b. (b) Illustrates how a radar phrase as represented by a pulse sequence can be decomposed into a series of radar words as defined in (a).***

- *Radar phrase (radar task)*: Concatenation of finite number of radar words. Each phrase may be implemented by more than one concatenation of radar words. Examples are search and target acquisition.
- *Radar policy*: Preoptimized schemes that allocate resources to radar phrases. An example is rules of engagement or policies of operation.

Fig. 7 illustrates how a radar phrase and radar words are related. Fig. 7(a) shows two radar words that are represented by symbols "*a*" and "*b*," where vertical bars represent radar pulses. Fig. 7(b) illustrates a sequence of radar words for a radar phrase, and which is constructed from concatenation of *a* and *b* into a sequence "*abaa*."

The generation process of radar words is governed according to MFRs' system architecture[2] as illustrated in Fig. 8. An MFR consists of three main components: situation assessment, system manager, and phrase scheduler/radar controller. The situation assessment module provides feedback of the tactic environment, and the system manager, based on the feedback, selects a radar policy. Each radar policy is a resource allocation scheme that represents tradeoffs between different performance measures, and it dictates how the phrase scheduler/radar controller will operate. Examples of the radar policies are long range track acquisition and short range self protect target acquisition policies: the major performance measures in these two policies are false alarm rate and track latency; track latency (false alarm rate) is tolerable in long range track acquisition policy (short range self protect target acquisition policy) and may be sacrificed for lower false alarm rate (track latency).

The scheduling and generation of radar words, on the other hand, is dictated by two controllers, phrase scheduler and radar controller, and their corresponding queues, planning queue and command queue, respectively. The reason for having the queues is driven by the need for MFR to be both adaptive and fast [49]. The planning queue stores scheduled radar phrases that are ordered by time and priority, and it allows the scheduling to be modified by phrase scheduler. Due to the system's finite response time, radar phrases in the planning queue are retrieved sequentially and entered to the command queue where no further planning or adaptation is allowed. Radar controller maps the radar phrases in the command queue to radar words and which are fixed for execution.

More specifically, the phrase scheduler models MFRs' ability to plan ahead its course of action and to pro-actively monitor the feasibility of its scheduled tasks [50]. Such an ability is essential because MFR switches between radar phrases, and conflicts such as execution order and system loading must be resolved ahead of time based on the predicted system performance and the tactic environment. In addition, planning is also necessary if MFR is interfaced with an external device, where the execution of certain phrases needs to meet a fixed time line. Radar controller, on the other hand, models MFR's ability to convert radar phrases to a multitude of different radar words depending on the tactic environment. Such an arrangement follows the macro/micro architecture as described in Blackman and Popoli [14]; the phrase scheduler determines *which* phrase the MFR is to perform that best utilize the system resources to achieve the mission goal, and the radar controller determines *how* the particular phrase is to be performed.

The MFR's operational details that are to be modeled are described here. Phrase scheduler processes the radar phrases in the planning queue sequentially from left to right. (If the queue is empty, an appropriate radar phrase is inserted.) To process different types of radar phrases, phrase scheduler calls their corresponding control rules;

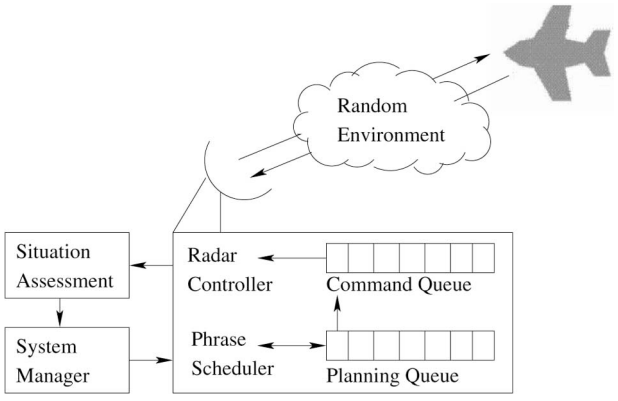


Fig. 8. ***The figure illustrates MFRs' system architecture. The situation assessment module evaluates the tactic environment and provides feedback to the system manager. The system manager, based on the feedback, selects the radar policy in which the phrase scheduler/radar controller will operate. The phrase scheduler initiates and schedules radar phrases in the planning queue and the phrases fixed for execution are moved to the command queue. The phrases in the command queue are mapped to appropriate radar words by the radar controller and are sent to MFR for execution.***

[2]The system architecture does not include multiple target tracking functionalities such as data association. The paper focuses on a single target's self protection and threat estimation, and thus models only the radar signal that a single target can observe.

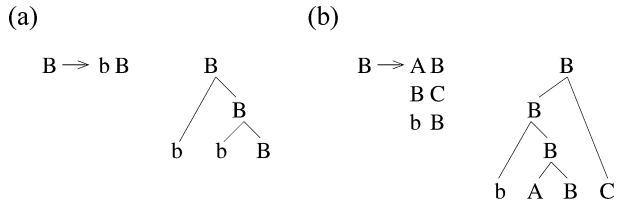


Fig. 9. *The figure illustrates the derivation process with two different types of production rules. (a) The derivation of the rule of regular grammar type. (b) The derivation of the rule of CFG type.*

the rule takes the radar phrase being processed as input, and responds by appending appropriate radar phrases into the command queue and/or the planning queue. The selection of the control rules is a function of radar policies, and which are expressed by how probable each rule would be selected. Similar to phrase scheduler, the radar controller processes the radar phrases in the command queue sequentially and maps the radar phrases to radar words according to a set of control rules.

B. Inadequacy of HMM for Modeling MFR

The distinguishing features of MFRs compared to conventional radars are their ability to switch between radar tasks, and their use of schedulers to plan ahead the course of action [49]. In order to model such features, as will be shown later in the next section, partial production rules of the form

1) $B \rightarrow bB$ and
2) $B \rightarrow AB|BC|bB$

are devised (See Section III-D for details); Fig. 9 illustrates the production rules and their derivations. The significance of the rules is that since HMM is equivalent to stochastic regular grammar [51] [rules of the form 1)], and MFRs follow rules that strictly contain regular grammar [rules of the form 2) cannot be reduced to 1)], HMM is not sufficient to model MFRs' signal. Furthermore, for sources with hidden branching processes (MFRs), SCFG is shown to be more efficient than HMM; the estimated SCFG has lower entropies than that of HMM [31].

Remark: The set of production rules presented above is a self-embedding CFG and thus its language is not regular, and cannot be represented by a Markov chain [10]. For the rules presented, self-embedding property can be shown by a simple derivation

$$B \rightarrow AB \rightarrow ABC.$$

In addition to the self-embedding property derived from the scheduling process, the generation of words by the radar controller poses another problem. For each radar phrase scheduled, a variable number of radar words may be generated. If HMM is applied to study the sequence of radar words, the Markovian dependency may be of variable length. In this case, maximum length dependency needs to be used to define the state space, and the exponential growing state space might be an issue.

C. A Syntactic Approach to MFR

In terms of natural language processing, we model the MFR as a system that "speaks" according to an SCFG. Based on the discussion in Section III-A, the syntactic approach is suitable in modeling the phrase scheduler because the scheduler operates according to a set of formal rules. On the other hand, the radar controller is suitably modeled because of two reasons: 1) each sequence of radar words is semantically ambiguous, i.e., many statistically distinct patterns (radar word sequence) possess the same semantic meanings (radar phrase), and 2) each radar phrase consists of finite number of radar words, and radar words are relatively easy to discriminate.

A Simple Example of MFR: As an illustrative example showing the correspondence between the grammar and the MFR, consider production rules of the form 1) $A \rightarrow aA$ and 2) $A \rightarrow BA$, where A and B are considered as radar phrases in the planning queue and a as a radar phrase in the command queue. The rule $A \rightarrow aA$ is interpreted as directing the phrase scheduler to append a to the queue list in the command queue, and A in the planning queue. Similarly, $A \rightarrow BA$ is interpreted as delaying the execution of A in the scheduling queue and inserting B in front of it. Suppose the planning queue contains the radar phrase A, a possible realization of the radar words' generation process is illustrated in Fig. 10. (The figure

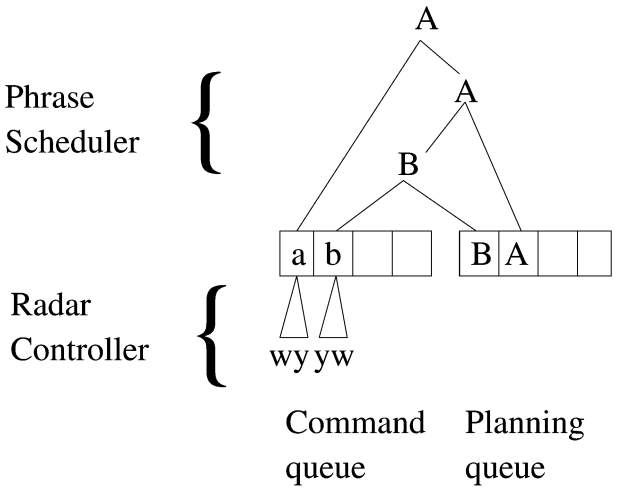


Fig. 10. *A possible realization of the scheduling process represented by a grammatical derivation process. A and B are radar phrases in the planning queue, a and b are radar phrases in the command queue and w and y are radar words.*

Table 2 List of All Mercury Emitter Phrase Combinations According to the Functional State of the Radar

Phrase	Words	Phrase	Words
Four-Word Search	$w_1w_2w_4w_5$	Track Maintenance (TM)	$w_1w_7w_7w_7$
	$w_2w_4w_5w_1$		$w_2w_7w_7w_7$
	$w_4w_5w_1w_2$		$w_3w_7w_7w_7$
	$w_5w_1w_2w_4$		$w_4w_7w_7w_7$
Three-Word Search	$w_1w_3w_5w_1$		$w_5w_7w_7w_7$
	$w_3w_5w_1w_3$		$w_6w_7w_7w_7$
	$w_5w_1w_3w_5$		$w_1w_8w_8w_8$
Acquisition	$w_1w_1w_1w_1$		$w_2w_8w_8w_8$
	$w_2w_2w_2w_2$		$w_3w_8w_8w_8$
	$w_3w_3w_3w_3$		$w_4w_8w_8w_8$
	$w_4w_4w_4w_4$		$w_5w_8w_8w_8$
	$w_5w_5w_5w_5$		$w_6w_8w_8w_8$
NAT or TM	$w_1w_6w_6w_6$		$w_1w_9w_9w_9$
	$w_2w_6w_6w_6$		$w_2w_9w_9w_9$
	$w_3w_6w_6w_6$		$w_3w_9w_9w_9$
	$w_4w_6w_6w_6$		$w_4w_9w_9w_9$
	$w_5w_6w_6w_6$		$w_5w_9w_9w_9$
Range Resolution	$w_7w_6w_6w_6$		$w_6w_9w_9w_9$
	$w_8w_6w_6w_6$		$w_7w_7w_7w_7$
	$w_9w_6w_6w_6$		$w_8w_8w_8w_8$
Acq., NAT, TM	$w_6w_6w_6w_6$		$w_9w_9w_9w_9$

also illustrates the operation of the radar controller; the triangle represents the mapping of the radar phrase to the radar words, which are denoted by y and w.) It can be seen that as long as the command queue phrases appear only to the left of planning queue phrases in the rule, the command queue and the planning queue are well represented.

D. A Syntactic Model for an MFR Called Mercury

The syntactic modeling technique is discussed in this subsection, and the discussion is based on an anti-aircraft defense radar called Mercury. The radar is classified and its intelligence report is sanitized and provided in Appendix A. Table 2 provides an exhaustive list of all possible Mercury phrases, and associates them with the functional states of the radar. This table was obtained from specifications in the sanitized intelligence report and is central to grammatical derivations that follow.

The SCFG modeling the MFR is $G = \{N_p \cup N_c, T_c, P_p \cup P_c, S\}$, where N_p and N_c are nonterminals representing the sets of radar phrases in the planning queue and command queue respectively, T_c is terminals representing the set of radar words, P_p is production rules mapping N_p to $(N_c \cup N_p)^+$, P_c is the set of production rules mapping N_c to T_c^+, and S is the starting symbol. It should be noted that the selection of production rules is probabilistic because MFR's inner workings cannot be known completely. In this subsection, each components of MFR as illustrated in 8 will be discussed in turn.

1) Phrase Scheduler: The phrase scheduler models the MFR's ability to plan and to preempt radar phrases based on the radar command and the dynamic tactic environment. Its operational rules for the scheduling and rescheduling of phrases are modeled by the production rule P_p, and it is found that P_p could be constructed from a small set of basic rules. Suppose $N_p = \{A, B, C\}$ and $N_c = \{a, b, c\}$, The basic control rules that are available to the phrase scheduler are listed below

$$\begin{array}{ll} \text{Markov} & B \rightarrow bB|bC \\ \text{Adaptive} & B \rightarrow AB|BC \\ \text{Terminating} & B \rightarrow b. \end{array}$$

The interpretation of the rules follows the example given at the end of the previous subsection. A rule is *Markov* if it sent a radar phrase to the command queue, and rescheduled either the same or different radar phrase in the planning queue. A rule is *adaptive* if it either preempted a radar phrase for another radar phrase or if it scheduled a radar phrase ahead of time in the radar's time line after the current phrase. A rule is *terminating* if it sent a radar phrase to the command queue without scheduling any new phrases.

The significance of the Markov rule is obvious, as it represents the dynamics of FSA. A simple example of the Markov rule is illustrated in Fig. 11 based on Mercury's functional states. According to the specifications in Appendix A, relative to each individual target, the Mercury emitter can be in one of the seven *functional states*—search, acquisition, nonadaptive track (NAT), three stages of range resolution, and track maintenance (TM). The transitions between these functional states can be captured by the state machine illustrated in Fig. 11.

The Mercury's state machine is generalized by including the adaptive and terminating rules. The inclusion

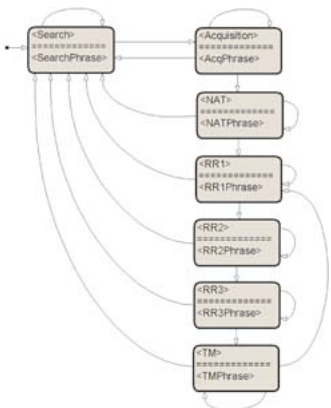


Fig. 11. ***Mercury emitter functionality at the high level. There are seven functional states of the emitter (search, acquisition, NAT, three stages of range resolution, and TM). The transitions between the states are defined according to the specification of Appendix A. The state machine is shown as the Moor automaton with outputs defined by the states. A corresponding phrase from Table 2 is generated in every state of the automaton.***

of the adaptive rule models MFRs' ability to reschedule radar phrases when the system loading or the tactic environment changes. The two adaptive rules model after the MFRs' ability to 1) preempt and 2) Plan the radar phrases. The preempt ability is demonstrated by the rule $B \rightarrow AB$ where the radar phrase B is preempted when a higher priority task A enters the queue. On the other hand, the ability to plan is captured in the rule $B \rightarrow BC$ where the phrase C is scheduled ahead of time if its predicted performance exceeds a threshold. On the other hand, the terminating rule reflects the fact that the queues have finite length, and the grammatical derivation process must terminate and yield a terminal string of finite length.

All the control rules, except the adaptive rule, could be applied to any radar phrases available. The adaptive rule schedules phrases ahead of time and thus requires a reasonable prediction on the target's kinematics; it would not be applicable to phrases where the prediction is lacking. Applying the rules to Mercury's radar phrases, the production rule P_p could be constructed and it is listed in Fig. 12.

2) Radar Controller and the Stochastic Channel: In this section, we develop deterministic, characteristic, and stochastic phrase structure grammars of the Mercury's radar controller as described in Appendix A. The grammar is derived as a word-level syntactic model of the emitter.

$$
\begin{aligned}
< S > \quad &\rightarrow \quad < FourWSearch > | < ThreeWSearch > | \\
&\quad < ACQ > | < NAT > | < RR1 > | \\
&\quad < RR2 > | < RR3 > | < TM > \\
< ThreeWSearch > \quad &\rightarrow \quad < ThreeWSearchPhrase >< ThreeWSearch > | \\
&\quad < ThreeWSearchPhrase >< ACQ > | \\
&\quad < ThreeWSearchPhrase > \\
< FourWSearch > \quad &\rightarrow \quad < FourWSearchPhrase >< FourWSearch > | \\
&\quad < FourWSearchPhrase >< ACQ > | \\
&\quad < FourWSearchPhrase > \\
< ACQ > \quad &\rightarrow \quad < AcqPhrase >< ACQ > | \\
&\quad < AcqPhrase >< NAT > | \\
&\quad < AcqPhrase >< ThreeWSearch > | \\
&\quad < AcqPhrase >< FourWSearch > | \\
&\quad < AcqPhrase > \\
< NAT > \quad &\rightarrow \quad < NATPhrase >< NAT > | < NATPhrase > \\
&\quad < NATPhrase >< RR_1 > | \\
&\quad < NATPhrase >< ThreeWSearch > | \\
&\quad < NATPhrase >< FourWSearch > \\
< RR_1 > \quad &\rightarrow \quad < RR_1Phrase >< RR_1 > | < RR_1Phrase > \\
&\quad < RR_1Phrase >< RR_2 > | \\
&\quad < RR_1Phrase >< ThreeWSearch > | \\
&\quad < RR_1Phrase >< FourWSearch > \\
< RR_2 > \quad &\rightarrow \quad < RR_2Phrase >< RR_2 > | < RR_2Phrase > \\
&\quad < RR_2Phrase >< RR_3 > | \\
&\quad < RR_2Phrase >< ThreeWSearch > | \\
&\quad < RR_2Phrase >< FourWSearch > \\
< RR_3 > \quad &\rightarrow \quad < RR_3Phrase >< RR_3 > | < RR_3Phrase > \\
&\quad < RR_3Phrase >< TM > | \\
&\quad < RR_3Phrase >< ThreeWSearch > | \\
&\quad < RR_3Phrase >< FourWSearch > \\
< TM > \quad &\rightarrow \quad < TMPhrase >< TM > | < TMPhrase > \\
&\quad < RR_3 >< TM > | \\
&\quad < TMPhrase >< ThreeWSearch > | \\
&\quad < TMPhrase >< FourWSearch >
\end{aligned}
$$

Fig. 12. ***Production rules of Mercury's phrase scheduler.***

We consider how the dynamics of radar words that make up one of the individual vertical slots in Fig. 23 captures internal processes occurring within the radar emitter.

Deterministic Grammar: According to Def. 2.2, a deterministic grammar is defined through its alphabet, the set of nonterminals, and the set of grammatical production rules. Using the Mercury specification of Appendix A, we can define the alphabet as

$$\mathcal{A} = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9\} \tag{16}$$

where $w_1, \ldots, w_9$ are the words of the Mercury emitter.

At every functional state, the radar emits a phrase consisting of four words drawn from the Table 2. Some phrases are unique and directly identify the functional state of the radar (i.e. $[w_1w_2w_4w_5]$ can only be encountered during search operations). Other phrases are characteristic to several radar states (i.e. $[w_6w_6w_6w_6]$ can be utilized in acquisition, NAT, and TM). These phrases form strings in the radar language that we are interested in modeling syntactically.

To complete the derivation of the grammar, we must define the rules for the $\langle XPhrase \rangle$ nonterminals where X stands for the corresponding name of the emitter state in which this phrase is emitted.

Using data from Table 2, we define the triplets

$$T_6 \rightarrow w_6w_6w_6 \quad T_8 \rightarrow w_8w_8w_8$$
$$T_7 \rightarrow w_7w_7w_7 \quad T_9 \rightarrow w_9w_9w_9$$

and the blocks of four words

$$
\begin{aligned}
&Q_1 \rightarrow w_1w_1w_1w_1 \quad Q_4 \rightarrow w_4w_4w_4w_4 \quad Q_7 \rightarrow w_7w_7w_7w_7 \\
&Q_2 \rightarrow w_2w_2w_2w_2 \quad Q_5 \rightarrow w_5w_5w_5w_5 \quad Q_8 \rightarrow w_8w_8w_8w_8 \\
&Q_3 \rightarrow w_3w_3w_3w_3 \quad Q_6 \rightarrow w_6w_6w_6w_6 \quad Q_9 \rightarrow w_9w_9w_9w_9.
\end{aligned}
$$

The $\langle ThreeWSearchPhrase \rangle$ and $\langle FourWSearchPhrase \rangle$ rules are:

$$
\begin{aligned}
&\langle FourWSearchPhrase \rangle \\
&\quad \rightarrow w_1w_2w_4w_5|w_2w_4w_5w_1|w_4w_5w_1w_2|w_5w_1w_2w_4 \\
&\langle ThreeWSearchPhrase \rangle \\
&\quad \rightarrow w_1w_3w_5w_1|w_3w_5w_1w_3|w_5w_1w_3w_5.
\end{aligned}
$$

The $\langle AcqPhrase \rangle$ rules are

$$\langle AcqPhrase \rangle \rightarrow Q_1|Q_2|Q_3|Q_4|Q_5|Q_6.$$

The $\langle NATPhrase \rangle$ rules are

$$\begin{aligned} \langle NATPhrase \rangle &\rightarrow S_1 T_6 | Q_6 \\ S_1 &\rightarrow w_1 | w_2 | w_3 | w_4 | w_5. \end{aligned}$$

The range resolution rules are

$$\begin{aligned} \langle RR_1 Phrase \rangle &\rightarrow w_7 T_6 \\ \langle RR_2 Phrase \rangle &\rightarrow w_8 T_6 \\ \langle RR_3 Phrase \rangle &\rightarrow w_9 T_6. \end{aligned}$$

Finally, the track maintenance rules are

$$\begin{aligned} \langle TMPhrase \rangle &\rightarrow \langle FourWTrack \rangle | \langle ThreeWTrack \rangle \\ \langle FourWTrack \rangle &\rightarrow Q_6 | Q_7 | Q_8 | Q_9 \\ \langle ThreeWTrack \rangle &\rightarrow S_1 T_6 | S_2 T_7 | S_2 T_8 | S_2 T_9 \\ S_2 &\rightarrow S_1 | w_6 \\ S_1 &\rightarrow w_1 | w_2 | w_3 | w_4 | w_5. \end{aligned}$$

According to the Chomsky hierarchy discussed in Section II, this grammar is a CFG.

Characteristic Grammar: The characteristic grammar of the Mercury emitter must extend the deterministic grammar to accommodate for the possible uncertainties in the real life environment. These uncertainties are due to the errors in reception and identification of the radar words. Conceptually, this process can be described by the model of the stochastic erasure channel with propagation errors.

To accommodate for the channel impairment model, we have to make two modifications to the deterministic grammar of the Mercury emitter. First of all, the alphabet (16) has to be expanded to include the character—the character indicating that no reliable radar signal detection was possible

$$\begin{aligned} \mathcal{A}_c &= \{\emptyset\} \cup \mathcal{A} \\ &= \{\emptyset, \mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3, \mathrm{w}_4, \mathrm{w}_5, \mathrm{w}_6, \mathrm{w}_7, \mathrm{w}_8, \mathrm{w}_9\}. \end{aligned} \tag{17}$$

Finally, we introduce an additional level of indirection into the grammar by adding nine new nonterminals $W_1, \ldots, W_9$ and nine new production rules

$$W_i \rightarrow \emptyset | \mathrm{w}_1 | \mathrm{w}_2 | \mathrm{w}_3 | \mathrm{w}_4 | \mathrm{w}_5 | \mathrm{w}_6 | \mathrm{w}_7 | \mathrm{w}_8 | \mathrm{w}_9 \tag{18}$$

where $i = 1, \ldots, 9$. The reason for this modification will become apparent when we associate probabilities with the production rules of the grammar.

Stochastic Grammar: The channel impairment model has the following transition probability matrix:

$$\mathbf{P}_o = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4 \quad \mathbf{p}_5 \quad \mathbf{p}_6 \quad \mathbf{p}_7 \quad \mathbf{p}_8 \quad \mathbf{p}_9]^T \tag{19}$$

where $\mathbf{p}_i = [p_{i,j}]_{0 \le j \le 9}$ is the probability that the transmitted radar word i being inferred by the pulse train analysis layer of the EW receiver as radar word j via the noisy and corrupted observations.

The stochastic radar grammar can be obtained from the characteristic grammar by associating probability vectors of (19) with the corresponding productions of (18)

$$W_i \xrightarrow{\mathbf{p}_i} \emptyset | \mathrm{w}_1 | \mathrm{w}_2 | \mathrm{w}_3 | \mathrm{w}_4 | \mathrm{w}_5 | \mathrm{w}_6 | \mathrm{w}_7 | \mathrm{w}_8 | \mathrm{w}_9 \tag{20}$$

where $i = 1, \ldots, 9$. Thus, the complete stochastic grammar of the Mercury emitter is shown in Fig. 13.

Strictly speaking, this grammar should be called weighted grammar rather than stochastic. As shown by [23], a stochastic CFG must satisfy the limiting stochastic consistency criterion. However, [23] demonstrates that useful syntactic pattern recognition techniques apply equally well to both the stochastic, and the weighted CFGs. Therefore, we are not concerned with satisfying the stochastic consistency criterion at this point.

E. MFR and System Manager—Markov Modulated SCFG

From the previous section, the phrase scheduler and the radar controller are modeled by constructing the context-free backbone of the MFR grammar. The third

$$\begin{aligned}
< AcqPhrase > &\xrightarrow{1} Q_1|Q_2|Q_3|Q_4|Q_5|Q_6 \\
< NATPhrase > &\xrightarrow{1} S_1T_6|Q_6 \\
< RR_1Phrase > &\xrightarrow{1} W_7T_6 \\
< RR_2Phrase > &\xrightarrow{1} W_8T_6 \\
< RR_3Phrase > &\xrightarrow{1} W_9T_6 \\
< TMPhrase > &\xrightarrow{1} < FourWTrack > | < ThreeWTrack > \\
< FourWSearch > &\xrightarrow{1} W_1W_2W_4W_5|W_2W_4W_5W_1|W_4W_5W_1W_2|W_5W_1W_2W_ \\
< ThreeWSearch > &\xrightarrow{1} W_1W_3W_5W_1|W_3W_5W_1W_3|W_5W_1W_3W_5 \\
< FourWTrack > &\xrightarrow{1} Q_6|Q_7|Q_8|Q_9 \\
< ThreeWTrack > &\xrightarrow{1} S_1T_6|S_2T_7|S_2T_8|S_2T_9 \\
S_2 &\xrightarrow{1} S_1|W_6 \\
S_1 &\xrightarrow{1} W_1|W_2|W_3|W_4|W_5 \\
T_6 &\xrightarrow{1} W_6W_6W_6 \\
T_7 &\xrightarrow{1} W_7W_7W_7 \\
T_8 &\xrightarrow{1} W_8W_8W_8 \\
T_9 &\xrightarrow{1} W_9W_9W_9 \\
Q_i &\xrightarrow{1} W_iW_iW_iW_i \\
W_i &\xrightarrow{\mathbf{p}_i} \varnothing|w_1|w_2|w_3|w_4|w_5|w_6|w_7|w_8|w_9 \\
& \quad i = 1, \ldots, 9
\end{aligned}$$

Fig. 13. ***Weighted grammar of the Mercury emitter. This grammar, like its deterministic counterpart, is a CFG.***

component of the MFR as described in Section III-A is the system manager. The operation of the system manager is deciding on the optimal policy of operation for each time period, and which is modeled by assigning production rule probabilities to the context-free backbone. The evolution of the policies of operation is driven by the interaction between the targets and the radar, and the the production rules' probabilities conveniently represent the resource allocation scheme deployed.

The state space of MFR is constructed based on different tactic scenarios. Let $k = 0, 1, \ldots$ denote discrete time. The policies of operation x_k is modeled as a M-state Markov chain. Define the transition probability matrix as $A = [a_{ji}]_{M\times M}$, where $a_{ji} = P(x_k = e_i | x_{k-1} = e_j)$ for $i, j \in \{1, 2, \ldots, M\}$. For example, depending on the loading condition of the MFR, two states may be defined according to the amount of resources allocated to MFR's target searching and target acquisition functions. More specifically, one state may consider the scenario where the number of targets detected is low, and thus higher proportion of radar resources are allocated to search functions. The other state may consider the scenario where the number of targets detected is high, and resources are allocated to target acquisition and tracking functions.

In each state, the MFR will "speak" a different "language," which is defined by its state-dependent grammar. Denote the grammar at state e_i as $G_i = \{N_p \cup N_c, T_c, P_p^i \cup P_c^i, S\}$, and it is noted that the grammars' context-free backbone is identical for all i except the probability distributions defined over their production rules. Each state of MFR is characterized by its policy of operations, and which determines the resource allocation to the targets. Obviously, the more resource allocated to the target, the higher the threat MFR poses on the target. From this perspective, threat estimation of MFR is reduced to a state estimation problem.

One practical issue is that the signal generated by radar systems has finite length, and this finiteness constraint must be satisfied by the SCFG is the model is to be stable. We discuss this point by first defining the stochastic mean matrix.

Definition: Let $A, B \in N$, the stochastic mean matrix M_N is a $|N| \times |N|$ square matrix with its (A, B)th entry being the expected number of variables B resulting from rewriting A

$$M_N(A, B) = \sum_{\eta \in (N\cup T)^* s.t. (A\to\eta)\in P} P(A \to \eta) n(B; \eta)$$

where $P(A \to \eta)$ is the probability of applying the production rule $A \to \eta$, and $n(B; \eta)$ is the number of instances of B in η [52].

The finiteness constraint is satisfied if the grammar in each state satisfies the following theorem.

Theorem: If the spectral radius of M_N is less than one, the generation process of the SCFG will terminate, and the derived sentence is finite.

Proof: The proof can be found in [52].

IV. SIGNAL PROCESSING IN CFG DOMAIN

A. Overview of MFR Signal Processing at Two Layers of Abstractions

In the previous section, we have considered the representation problem where the MFR is specified as a Markov modulated SCFG, and its production rules derived from the radar words' syntactic structure. In this and the next section, we will deal with the signal processing of the MFR signal and present algorithms for state and parameter estimation. The signal processing problem is illustrated in Fig. 14, where it is decomposed into two layers of abstractions; the radar controller layer and the phrase scheduler.

The higher layer of processing is discussed in Section IV, where the state and parameter estimation are both processed in the CFG framework. Based on the estimated radar phrases passed from the radar control layer (hard decision is applied in the passing of radar phrases), the MFR's policies of operation is estimated by a hybrid of the Viterbi and the inside algorithm. In addition, maximum likelihood parameter estimator of the unknown system parameters will be derived based on the Expectation Maximization algorithm.

In Section V, a principled approach is discussed to deal with the signal processing of the NSE CFG. A synthesis procedure that converts an NSE CFG to its finite-state counterpart in polynomial time is introduced. The radar controller will be shown to be NSE, and its finite-state representation will be derived. Once the radar controller is

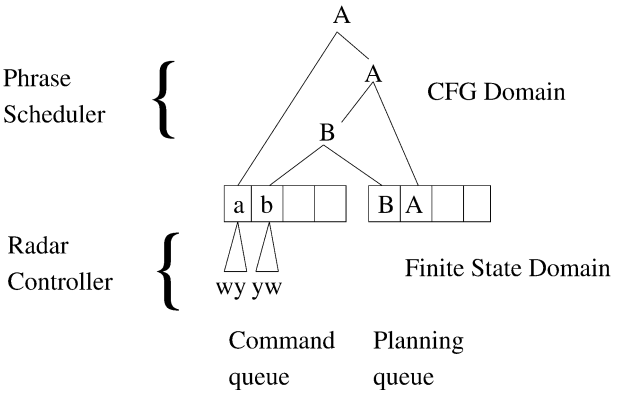


Fig. 14. ***Signal processing in two levels of abstractions, the word level and the phrase level.***

in its finite-state form, its signal processing could be performed with an FSA.

Following the state space notation introduced in the previous section, let $x_{0:n} = (x_0, x_1, \ldots, x_n)$ be the (unknown) state sequence, and $\gamma_{1:n} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ be the corresponding intercepted radar signal stored in the track file. Each $\gamma_k = (w_1, w_2, \ldots, w_{m_k})$ is a string of concatenated terminal symbols (radar words), and m_k is the length of γ_k.

In order to facilitate the discussion of the estimation algorithms, it is convenient to introduce the following variables:

- Forward variable:

$$f_i(k) = P(\gamma_1, \gamma_2, \ldots, \gamma_k, x_k = e_i)$$

- Backward variable:

$$b_i(k) = P(\gamma_{k+1}, \gamma_{k+2}, \ldots, \gamma_n | x_k = e_i)$$

- Inside variable:

$$\beta_j^i(k, p, q) = P\left(w_{pq} | A_{pq}^j, x_k = e_i\right)$$

- Outside variable:

$$\alpha_j^i(k, p, q) = P\left(w_{1(p-1)}, A_{pq}^j, w_{(q+1)m} | x_k = e_i\right)$$

where w_{pq} is the subsequence of terminals from pth position of γ_i to qth position, and A_{pq}^j is the nonterminal $A^j \in N_p$ which derives w_{pq}, or $A^j \overset{*}{\Rightarrow} w_{pq}$. Fig. 15 illustrates the probabilities associated with inside and outside variables.

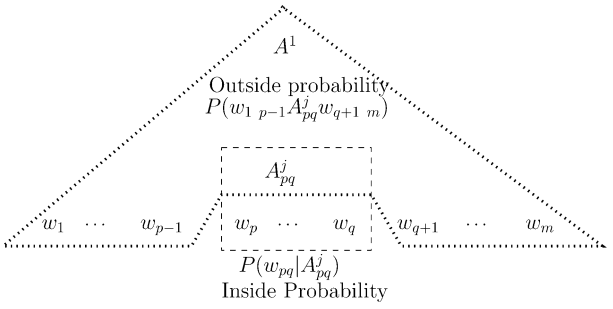


Fig. 15. *Inside and outside probabilities in SCFG.*

B. Bayesian Estimation of MFR's State via Viterbi and Inside Algorithms

The estimator of MFR's state at time k is

$$\hat{x}_k = \arg\max_i P(x_k = e_i | \gamma_{1:n})$$

and which could be computed using the Viterbi algorithm. Define

$$\delta_i(k) = \max_{x_0, x_1, \ldots, x_{k-1}} P(x_0, x_1, \ldots, x_k = i, \gamma_1, \gamma_2, \ldots, \gamma_k)$$

the Viterbi algorithm computes the best state sequence inductively as follows:

1) Initialization: $\delta_i(1) = \pi_i o_i(\gamma_1)$, for $1 \le i \le M$.
2) Induction:

$$\delta_i(k+1) = \max_{1 \le j \le M} \left[\delta_i(k) a_{ji}(\psi)\right] o_i(\gamma_{k+1}),$$

where $1 \le k \le n-1$ and $1 \le i \le M$

$$\psi_i(k+1) = \arg\max_{1 \le j \le M} \delta_i(k) a_{ji},$$

where $1 \le k \le n-1$ and $1 \le i \le M$.

3) Termination:

$$\hat{x}_n = \arg\max_{1 \le j \le M} \delta_j(n)$$

4) Path backtracking:

$$\hat{x}_k = \psi_{k+1}(\hat{x}_{k+1}), \quad k = n-1, n-2, \ldots, 1$$

where $o_i(\gamma_k)$ is the output probability of the string γ_k generated by the grammar G_i. An efficient way to calculate the probability is by the inside algorithm, a dynamic programming algorithm that inductively calculates the probability.

The inside algorithm computes the probability $o_i(\gamma_k)$ inductively as follows:

1) Initialization: $\beta_j(k, k) = P(A^j \to w_k | G_i)$.
2) Induction:

$$\beta_j(p, q) = \sum_{r,s} \sum_{d=p}^{q-1} P(A^j \to A^r A^s) \beta_r(p, d) \beta_s(d+1, q)$$

for $\forall j$, $1 \le p < q \le m_k$.

3) Termination: $o_i(\gamma_k) = \beta_1(1, m_k)$.

Running both the Viterbi and the inside algorithms, the posteriori distribution of the states given the observation could be computed.

C. MFR Parameter Estimation Using EM Algorithm

The Expectation Maximization (EM) algorithm is a widely used iterative numerical algorithm for computing maximum likelihood parameter estimates of partially observed models. Suppose we have observations $\Gamma_n = \{\gamma_1, \ldots, \gamma_n\}$ available, where n is a fixed positive integer. Let $\{P_\phi, \phi \in \Phi\}$ be a family of probability measures on $(\Omega, \mathcal{F})$ all absolutely continuous with respect to a fixed probability measure P_o. The likelihood function for computing an estimate of the parameter ϕ based on the information available in Γ_n is

$$\mathcal{L}_n(\phi) = \mathbf{E}_o\left[\frac{dP_\phi}{dP_o}|\Gamma_n\right]$$

and the maximum likelihood estimate (MLE) is defined by

$$\hat{\phi} \in \arg\max_{\phi\in\Phi} \mathcal{L}_n(\phi).$$

The EM algorithm is an iterative numerical method for computing the MLE. Let $\hat{\phi}_o$ be the initial parameter estimate. The EM algorithm generates a sequence of parameter estimates $\{\phi_j\}$ $j \in \mathbb{Z}^+$ as follows:

Each iteration of the EM algorithm consists of two steps:

Step 1) (Expectation-step) Set $\tilde{\phi} = \hat{\phi}_j$ and compute $Q(\cdot, \tilde{\phi})$, where

$$Q(\phi, \tilde{\phi}) = \mathbf{E}_{\tilde{\phi}}\left[\log\frac{dP_\phi}{dP_{\tilde{\phi}}}|\Gamma_n\right].$$

Step 2) (Maximization-step) Find

$$\hat{\phi}_{j+1} \in \arg\max_{\phi\in\Phi} Q(\phi, \phi_j).$$

Using Jensen's inequality, it can be shown (see Theorem 1 in [53]) that sequence of model estimates $\{\hat{\phi}_j\}$, $j \in \mathbb{Z}^+$ from the EM algorithm are such that the sequence of likelihoods $\{\mathcal{L}_n(\hat{\phi}_j)\}$, $j \in \mathbb{Z}^+$ is monotonically increasing with equality if and only if $\hat{\phi}_{j+1} = \hat{\phi}_j$.

In Section IV-B, MFR's state estimation problem was discussed. However, the algorithm introduced assumes complete knowledge of system parameters, i.e., the Markov chain's transition matrix and the SCFG's production rules. In reality, such parameters are often unknown. In this subsection, EM algorithm is applied to a batch of noisy radar signal in the track file, and system parameters are estimated iteratively.

In EM's terminology, the intercepted radar signal (radar words), $\gamma_{1:n}$, is the incomplete observation sequence, and we have complete data if it is augmented with $\{x_{0:n}, C_{1:n}\}$. $x_{0:n}$ is the state sequence of the MFR system. $C_{1:n} = (C_1, \ldots, C_n)$ is the number of counts each production rule is used to derive $\gamma_{1:n}$, and in particular, $C_k = (C^1(A \to \eta; \gamma_k), C^2(A \to \eta; \gamma_k), \ldots, C^M(A \to \eta; \gamma_k))$ for $k = 1, 2, \ldots, n$ and $C^i(A \to \eta; \gamma_k)$ for $i = 1, 2, \ldots, M$ is the number of counts grammar G_i applies the production rule $A \to \eta$ in deriving γ_k.

Denote the parameter of interest as $\Phi = \{a_{ji}, P^1(A \to \eta), P^2(A \to \eta), \ldots, P^M(A \to \eta)\}$, where $P^i(A \to \eta)$ is set of probabilities of all the production rules defined in grammar i, the complete-data likelihood is written as

$$\mathcal{L}_n(\phi) = \prod_{k=1}^{n} P(\gamma_k, C_k|x_k, \phi)P(x_k|x_{k-1}, \phi)P(x_o|\phi).$$

In order to facilitate the discussion of the EM algorithm, the following two variables are introduced:

$$\begin{aligned}\chi_i(k) &= P(x_k = e_i|\gamma_{1:n}) \\ &= \frac{f_i(k)b_i(k)}{\sum_{i=1}^{3} f_i(k)b_i(k)} \\ \xi_{ji}(k) &= P(x_k = e_j, x_{k+1} = e_i|\gamma_{1:n}) \\ &= \frac{f_j(k)a_{ji}o_i(\gamma_{k+1})b_i(k+1)}{\sum_{j=1}^{3}\sum_{i=1}^{3} f_j(k)a_{ji}o_i(\gamma_{k+1})b_i(k+1)}.\end{aligned}$$

The Expectation step of the EM algorithm yields the following equation:

$$\begin{aligned}&E_{\tilde{\phi}}(\log\mathcal{L}_n(\phi)) \\ &= \sum_{k=1}^{n}\sum_{x_k}\sum_{A^{x_k}}\sum_{T^{x_k}} E_{\tilde{\phi}}(C^{x_k}(A \to \eta; \gamma_k)) \\ &\quad \times \log P^{x_k}(A \to \eta)\chi_{x_k}(k) \\ &\quad + \sum_{k=1}^{n}\sum_{x_k}\sum_{x_{k-1}} \log\left(a_{x_k|x_{k-1}}\right)\xi_{x_{k-1}x_k}(k-1) \\ &\quad + \sum_{k=1}^{n}\sum_{x_0} \log P(x_0)\chi_{x_0}(k)\end{aligned}$$

where $E_{\tilde{\phi}}(C^{x_k}(A \to \eta; \gamma_k))$ can be computed using inside and outside variables [51].

The maximization step of the EM algorithm could be computed by applying Lagrange multiplier. Since the parameters we wish to optimize are independently separated into three terms in the sum, we can optimize the parameter term by term. The estimates of the probabilities of the production rules can be derived using the first term of the equation, and the updating equation is

$$P^{x_k}(A \to \eta) = \frac{\sum_{k=1}^{n} E_{\tilde{\phi}}(C^{x_k}(A \to \eta; \gamma_k))\chi_{x_k}(k)}{\sum_{\eta}\sum_{k=1}^{n} E_{\tilde{\phi}}(C^{x_k}(A \to \eta; \gamma_k))\chi_{x_k}(k)}.$$

Similarly, the updating equation of the transition matrix a_{ji} is

$$a_{ji} = \frac{\sum_{k=1}^{n-1} \xi_{ji}(k)}{\sum_{k=1}^{n-1} \chi_j(k)}$$

Under the conditions in [54], iterative computations of the expectation and maximization steps above will produce a sequence of parameter estimates with monotonically nondecreasing likelihood. For details of the numerical examples, the parameterization of the Markov chain's transition matrix by logistic model, and the study of the predictive power (entropy) of SCFGs, please see [55].

V. SIGNAL PROCESSING IN FINITE-STATE DOMAIN

In this section, we deal with the lower layer of signal processing as described in Section IV-A. Before we discuss finite-state modeling and signal processing of syntactic MFR models, we need to provide some additional definitions.

1) CFGs and Production Graphs: The property of self-embedding of CFGs introduced in Section II is not very easy to determine through a simple visual inspection of grammatical production rules. More precise and formal techniques of the self-embedding analysis rely on the concept of CFG *production graphs*.

Definition 5.1: A *production graph* $P(G)$ for a CFG $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is a directed graph whose vertices correspond to the nonterminal symbols from $\mathcal{E}$, and there exists an edge from vertex A to vertex B if and only if there is a production in Γ such that $A \to \alpha B \beta$.

Definition 5.2: A *labeled production graph* $P_l(G)$ for a CFG $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is a production graph $P(G)$ with the set of labels $lab(\Gamma)$ defined over the set of edges of $P(G)$ in the following way:

$$lab(A \to B) = \begin{cases} l & \text{if for every } A \to \alpha B \beta \in \Gamma, \alpha \neq \varepsilon, \beta = \varepsilon, \\ r & \text{if for every } A \to \alpha B \beta \in \Gamma, \alpha = \varepsilon, \beta \neq \varepsilon, \\ b & \text{if for every } A \to \alpha B \beta \in \Gamma, \alpha \neq \varepsilon, \beta \neq \varepsilon, \\ u & \text{if for every } A \to \alpha B \beta \in \Gamma, \alpha = \varepsilon, \beta = \varepsilon. \end{cases} \quad (21)$$

Note that the production graphs of Def. 5.1 and Def. 5.2 are not related to FSAs or FSLs described earlier. They are simply useful graphical representations of CFGs.

Let us consider an example grammar (reproduced from [56] with modifications)

$$\mathcal{A} = \{a, b, c, d\},$$
$$\mathcal{E} = \{S, A, B, C, D, E, F\},$$
$$\Gamma = \left\{ \begin{array}{l} S \to DA \\ A \to bEaB \\ B \to aE|S \\ C \to bD \\ D \to daC|a \\ E \to D|Cc|aF|Fc \\ F \to bd \end{array} \right\}. \quad (22)$$

The labeled production graph for this grammar is illustrated in Fig. 16.

Definition 5.3: A *transition matrix* $\mathbf{M}(G)$ for the labeled production graph $P_l(G)$ of a CFG $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is an $N \times N$ matrix whose dimensions are equal to the number of

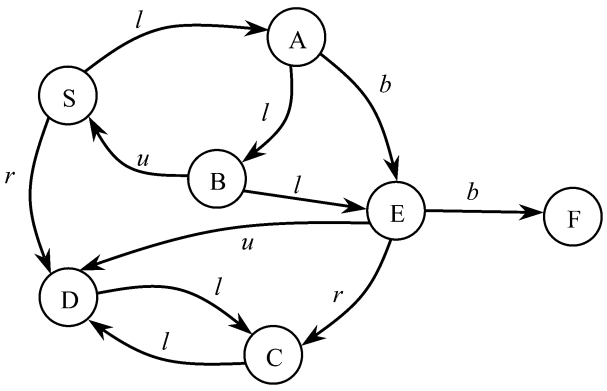


Fig. 16. *Production graph for the grammar in (22).*

nonterminal symbols in $\mathcal{E}$ (number of vertices in the production graph), and whose elements are defined as follows:

$$m_{i,j}(G) = \begin{cases} 0 & \text{if } A_i \to \alpha A_j \beta \notin \Gamma, \\ lab(A_i \to A_j) & \text{if } A_i \to \alpha A_j \beta \in \Gamma. \end{cases} \quad (23)$$

The transition matrix of the labeled production graph in Fig. 16 with respect to the vertex ordering $\{S, A, B, C, D, E, F\}$ has the following structure:

$$\mathbf{M}(G) = \begin{bmatrix} 0 & l & 0 & 0 & r & 0 & 0 \\ 0 & 0 & l & 0 & 0 & b & 0 \\ u & 0 & 0 & 0 & 0 & l & 0 \\ 0 & 0 & 0 & 0 & l & 0 & 0 \\ 0 & 0 & 0 & l & 0 & 0 & 0 \\ 0 & 0 & 0 & r & u & 0 & b \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (24)$$

In Section V-A, we will use production graphs and transition matrices in analysis of self-embedding property of CFG.

A. CFG-Based Finite-State Model Synthesis

This subsection is devoted to development of a procedure that allows to automatically synthesize a finite-state model of a DES using its CFG model as an input. We introduce a theoretical framework for determining whether a specified CFG of the system actually represents an FSL and provide an automated polynomial-time algorithm for generating the corresponding FSA.

This synthesis procedure consists of four basic steps.

1) **Test of self-embedding.** A CFG that is determined to be NSE describes an FSL (see Section II). Therefore, an FSA can be synthesized from this grammar.
2) **Grammatical decomposition.** First, the NSE CFG is broken down into a set of simpler FSGs.
3) **Component synthesis.** Once the grammar has been decomposed into a set of simpler grammars, an FSA can be synthesized for every one of these FSGs.
4) **Composition.** Finally, the components from the previous step are combined together to form a single FSA that is equivalent to the original NSE CFG of the MFR.

This procedure is based on combined results published in [41] and [57].

1) Verification of Non-Self-Embedding: As was mentioned earlier, if a CFG of a system is in the NSE form, this CFG has an equivalent finite-state representation. However, given an arbitrarily complex CFG, it is not possible to verify the NSE property of this grammar by simple visual inspection.

Table 3 Semiring Operations of Sum and Product

Sum						Product					
$+$	l	r	b	0	u	$\times$	l	r	b	0	u
l	l	b	b	l	l	l	l	b	b	0	l
r	b	r	b	r	r	r	b	r	b	0	r
b	b	b	b	b	b	b	b	b	b	0	b
0	l	r	b	0	u	0	0	0	0	0	0
u	l	r	b	u	u	u	l	r	b	0	u

In this section, we present a formal verification procedure of the NSE property of an arbitrary CFG. This procedure is based on the one described in [41], but it has been modified to suite the needs of the DES grammars.

Let us start by defining the concept of a *semi-ring* [58]:

Definition 5.4: A *semi-ring* is a set S together with addition "$+$" and multiplication "$\times$" operations defined over the elements of this set in such a way that they satisfy the following properties:

1) additive associativity: $(\forall e, g, f \in S)(e+g)+f = e+(g+f)$;
2) additive commutativity: $(\forall e, g \in S)e+g = g+e$;
3) multiplicative associativity: $(\forall e, g, f \in S)(e \times g) \times f = e \times (g \times f)$;
4) left and right distributivity: $(\forall e, g, f \in S)e \times (g+f) = (e \times g)+(e \times f)$ and $(e+g) \times f = (e \times f)+(g \times f)$.

Let us now define a semiring over the set of labels of production graphs of CFGs. This set of labels is introduced by Def. 5.2 and Def. 5.3. The *sum* and *product* operations of this semiring are listed in Table 3.

If $\mathbf{M}(G)$ is a transition matrix of the CFG G (see Def. 5.3), then, using the semiring operations of Table 3, we define the steady-state matrix of the production graph of this grammar as

$$\mathbf{M}^{\leq N}(G) = \sum_{i=1}^{N} [\mathbf{M}(G)]^i \quad (25)$$

where N is the dimension of the transition matrix $\mathbf{M}(G)$.

[41] have proven that if

$$\mathbf{diag}\left[\mathbf{M}^{\leq N}(G)\right]$$

does not contain labels "b," the corresponding grammar G is NSE. This demonstrates that the NSE property of a CFG can be verified in polynomial time.

To illustrate the application of this algorithm, let us revisit the example CFG (22). The labeled production graph for this grammar is shown in Fig. 16, and the transition matrix of this graph is given by (24).

$\mathbf{diag}[\mathbf{M}^{\leq N}(G)] = [l, l, l, l, l, 0, 0]$, therefore, CFG (22) is NSE. In the remainder of this section, we will describe a three-step procedure that accepts an NSE CFG and automatically synthesizes an FSA that is equivalent to this grammar.

2) Grammatical Decomposition: We start by introducing the concept of the grammatical $\oplus$-*composition*.

Definition 5.5: If $G_1 = (\mathcal{A}_1, \mathcal{E}_1, \Gamma_1, S_1)$ and $G_2 = (\mathcal{A}_2, \mathcal{E}_2, \Gamma_2, S_2)$ are two CFGs with $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$ and $\mathcal{E}_1 \cap \mathcal{A}_2 = \emptyset$, then $\oplus$-*composition* of these grammars is defined as

$$G = G_1 \oplus G_2 = (\mathcal{A}, \mathcal{E}, \Gamma, S)$$

where $\mathcal{A} = \mathcal{A}_1 \setminus \mathcal{E}_2 \cup \mathcal{A}_2$, $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$, $\Gamma = \Gamma_1 \cup \Gamma_2$, and $S = S_1$.

[41] have demonstrated that for any NSE CFG G there exist n FSGs $G_1, G_2, \ldots, G_n$ such that $G = G_1 \oplus G_2 \oplus \ldots \oplus G_n$. They have also shown that every FSG G_i of this decomposition is equivalent to some strongly connected component of the production graph $P(G)$.

The grammatical decomposition procedure consists of the following steps.

1) Let $P_1(G), P_2(G), \ldots, P_n(G)$ be n strongly connected components of the production graph $P(G)$. Then $\mathcal{E}_i$ of the FSG G_i is the same as the set of vertices in $P_i(G)$.
2) The set of terminal symbols of the FSG G_i is found through the following recursive relationship:

$$\begin{aligned} \mathcal{A}_n &= \mathcal{A} \\ \mathcal{A}_{n-1} &= \mathcal{A} \cup \mathcal{E}_n \\ &\vdots \\ \mathcal{A}_2 &= \mathcal{A} \cup \mathcal{E}_3 \cup \ldots \cup \mathcal{E}_n \\ \mathcal{A}_1 &= \mathcal{A} \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_n \end{aligned}$$

3) The set of grammatical production rules $\Gamma_i \subseteq \Gamma$ is defined as $\Gamma_i = \{A \to \alpha | A \in \mathcal{E}_i\}$.
4) Finally, the start symbol S_1 for the FSG G_1 is chosen as S_0 of the original NSE CFG G, and S_i for $i = 2, \ldots, n$ is chosen to be an arbitrary nonterminal from the corresponding set $\mathcal{E}_i$.

One of the most efficient procedures to decompose a directed graph into a set of strongly connected components involves Dulmage–Mendelsohn decomposition [59] of the production graph's adjacency matrix. This decomposition finds a permutation of the vertex ordering that renders the adjacency matrix into upper block triangular form. Each block triangular component of the transformed adjacency matrix corresponds to a strongly connected component of the production graph.

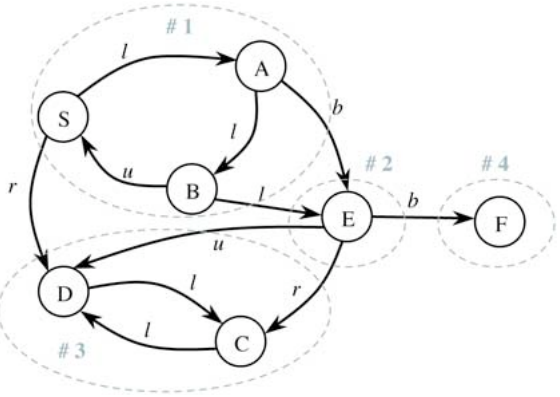


Fig. 17. ***Strongly connected components of the production graph shown in Fig. 16.***

Now consider an example of the decomposition procedure applied to grammar (22). Its production graph includes four strongly connected components shown in Fig. 17. The four FSG components of this CFG are

$$G_1 = \begin{pmatrix} \mathcal{A}_1 = \{a, b, c, d, C, D, E\}, \\ \mathcal{E}_1 = \{S, A, B\}, \\ \Gamma_1 = \left\{ \begin{matrix} S \to DA, \\ A \to bEaB, \\ B \to aE|S \end{matrix} \right\}, \\ S_1 = S \end{pmatrix}, \tag{26a}$$

$$G_2 = \begin{pmatrix} \mathcal{A}_2 = \{a, b, c, d, C, D, F\}, \\ \mathcal{E}_2 = \{E\}, \\ \Gamma_2 = \{E \to D|Cc|aF|Fc\}, \\ S_2 = E \end{pmatrix}, \tag{26b}$$

$$G_3 = \begin{pmatrix} \mathcal{A}_3 = \{a, b, c, d\}, \\ \mathcal{E}_3 = \{C, D\}, \\ \Gamma_3 = \left\{ \begin{matrix} C \to bD, \\ D \to daC|a \end{matrix} \right\}, \\ S_3 = \{X | X \in \mathcal{E}_3\} \end{pmatrix}, \tag{26c}$$

$$G_4 = \begin{pmatrix} \mathcal{A}_4 = \{a, b, c, d\}, \\ \mathcal{E}_4 = \{F\}, \\ \Gamma_4 = \{F \to bd\}, \\ qS_4 = F \end{pmatrix} \tag{26d}$$

where, in (26c), X may represent either of the nonterminals contained in $\mathcal{E}_3$.

3) Synthesis of Finite-State Components: The next step involves synthesis of individual FSAs for each of the FSGs $G_1, G_2, \ldots, G_n$ obtained at the step of grammatical decomposition. This is a straightforward mechanical procedure well described in the literature [23], [25], [26], [35], [40], [42].

The FSA for the example grammars (26) are shown in Fig. 18. They have the following structure:

$$\Lambda_1 = \begin{pmatrix} \Sigma_1 = \mathcal{A}_1, \\ Q_1 = \mathcal{E}_1 \cup Q_1', \\ \delta_1 = \delta(\Gamma_1), \\ q_0 = \{S\}, \\ F_1 = H \in Q_1' \end{pmatrix}, \tag{27a}$$

$$\Lambda_2 = \begin{pmatrix} \Sigma_2 = \mathcal{A}_2, \\ Q_2 = \mathcal{E}_2 \cup Q_2', \\ \delta_2 = \delta(\Gamma_2), \\ q_0 = \{E\}, \\ F_2 = H \in Q_2' \end{pmatrix}, \tag{27b}$$

$$\Lambda_3 = \begin{pmatrix} \Sigma_3 = \mathcal{A}_3, \\ Q_3 = \mathcal{E}_3 \cup Q_3', \\ \delta_3 = \delta(\Gamma_3), \\ q_0 = \{\{X\} | X \in \mathcal{E}_3\}, \\ F_3 = H \in Q_3' \end{pmatrix}, \tag{27c}$$

$$\Lambda_4 = \begin{pmatrix} \Sigma_4 = \mathcal{A}_4, \\ Q_4 = \mathcal{E}_4 \cup Q_4', \\ \delta_4 = \delta(\Gamma_4), \\ q_0 = \{F\}, \\ F_4 = H \in Q_4' \end{pmatrix} \tag{27d}$$

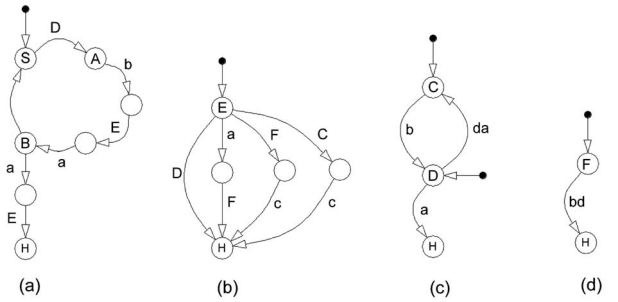


Fig. 18. ***Components of the FSA for the grammar (22). States that are not labeled in these graphs are considered intermediate and have no direct representation in the set of nonterminals of the grammars. (a) Corresponds to the grammar (26a). (b) Corresponds to the grammar (26b). (c) Corresponds to the grammar (26c). (d) Corresponds to the grammar (26d).***

where Q_i' are the sets of intermediate states required for construction of the automaton i that are not present in the set of nonterminals of the corresponding grammar G_i.

Note that we present here simplified versions of FSAs highlighting only the important structural aspects. Specifically, in Fig. 18(c) and (d) transitions labeled with nonterminals "*da*" and "*bd*," when rigorously treated, require the insertion of an intermediate state. Also, before constructing an FSA, the grammar should have been converted to the unit production free form so that every edge in the graphs in Fig. 18 corresponds to the generation of a single nonterminal. We will suppress these intermediate steps in the rest of this paper, and, without loss of generality, will adopt a slightly simplified form of FSA representation.

4) Composition of the FSA: The final step of the FSA synthesis procedure for a given NSE CFG involves composition of the FSA components obtained at the previous step. A recursive "depth-first" algorithm that performs this operation was developed by [57] and its modified version was presented by [56]. Here we present an alternative, "breadth-first" algorithm.

The FSA composition procedure is formalized in terms of two algorithms presented below. The main procedure "createFSA" initializes the composition operation and calls the function "expandFSA" that performs the actual FSA composition. We will illustrate this procedure by an example composing FSA components shown in Fig. 18.

Algorithm 1 Procedure "createFSA" creates an FSA

1) **procedure** CREATEFSA$(S_0, \Lambda_1, \ldots, \Lambda_n)$ ▷ Creates an FSA from components $\Lambda_1, \ldots, \Lambda_n$
2) $\Sigma \leftarrow \{S_0\}$ ▷ Initializing set of transitions
3) $Q \leftarrow \{q', H\}$ ▷ Adding intermediate states
4) $\delta \leftarrow \{\delta(q', S_0) = \{H\}\}$ ▷ Adding initial transition
5) $q_0 \leftarrow q'$ ▷ Setting initial state
6) $F \leftarrow \{H\}$ ▷ Setting terminal states
7) $\Lambda \leftarrow (\Sigma, Q, \delta, q_0, F)$ ▷ Initializing the FSA
8) $\Lambda \leftarrow$ expandFSA$(\Lambda, \Lambda_1, \ldots, \Lambda_n)$ ▷ Calling the expansion procedure
9) **end procedure**

Algorithm 2 Function "expandFSA" inserts FSA components into the FSA Λ

1) **function** EXPANDFSA$(\Lambda, \Lambda_1, \ldots, \Lambda_n)$ ▷ Inserts FSA components into Λ
2) **for all** Λ_i and $\alpha \in \Sigma$ **do**
3) **if** $\alpha \in Q_i$ **then** ▷ If transition matches a state
4) $q_{\text{from}} \leftarrow \arg_q(\delta(q_\alpha, \alpha))$ ▷ Saving *from* state

```
5)            q_to ← δ(q_from, α)          ▷ Saving to state
6)            Σ ← Σ \ α ∪ Σ_i              ▷ Expanding set of
                                               transitions
7)            Q ← Q ∪ Q_i                  ▷ Expanding set of states
8)            δ ← δ ∪ δ_i                  ▷ Appending transition
                                               structure
9)            for all δ(q_j, β) == q_from do
10)               δ(q_j, β) ← α            ▷ Rerouting q_from
                                               inputs
11)           end for
12)           for all δ(q_j, β) ∈ F_i do
13)               δ(q_j, β) ← q_to         ▷ Rerouting q_to
                                               inputs
14)           end for
15)           Q ← Q \ F_i                  ▷ Removing term.
                                               states of Λ_i
16)           Q ← Q \ q_from               ▷ Removing q_from state
17)       end if
18)   end for
19)   return Λ ▷ Returning the complete FSA Λ
20) end function
```

The initialization step involves creation of a "dummy" FSA that contains two states—an intermediate state and a terminal state. It also contains one transition from the intermediate state to the terminal state on symbol $S_0 = S$ from the original NSE CFG (22). This FSA is shown in Fig. 19(a).

The function "expandFSA" accepts the "dummy" FSA as well as four FSA components shown in Fig. 18. It then transforms the "dummy" FSA into a real automaton by consecutively inserting FSA components and rewiring transitions.

The step-by-step composition procedure is illustrated in Fig. 19(b)–(e). First, the FSA shown in Fig. 18(a) is inserted into the FSA in Fig. 19(a) instead of the transition labeled *S*. The resulting intermediate FSA is shown in Fig. 19(b). Next, all the *E*-transitions are replaced with the FSA in Fig. 18(b). The resulting intermediate FSA is shown in Fig. 19(c). Then, all the *F*-transitions are replaced with the FSA in Fig. 18(d). The resulting intermediate FSA is shown in Fig. 19(d). Finally, all the *C*- and *D*-transitions are replaced with the FSA in Fig. 18(c). The final automaton equivalent to the grammar (22) is shown in Fig. 19(e). Note that the labels of the states in Fig. 19 are not the unique state identifiers. These labels are shown to illustrate the composition procedure and to provide linkage with the states of the original FSA shown in Fig. 18.

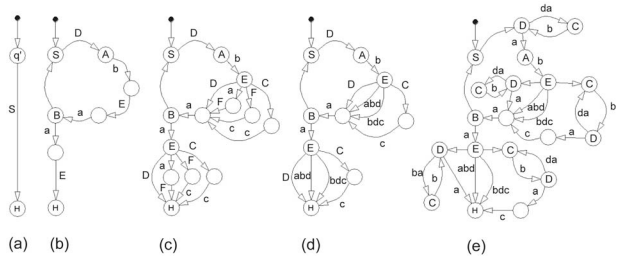


Fig. 19. ***Synthesis procedure of the FSA for the example CFG (22).***

B. State Machine Synthesis of the Mercury Radar Controller

As stated in [23], the analysis of stochastic and weighted grammars must be performed using their characteristic counterparts. However, since the characteristic grammar is so close to the original deterministic grammar, we can perform the NSE property test directly on the deterministic CFG.

Following the verification procedures as described in Section V-A1, the diagonal elements of the steady-state analysis matrix can be shown to be

$$\mathbf{diag}\left(\mathbf{M}^{\leq N}(G)\right) = [o \;\; o \;\; l \;\; l \;\; l \;\; l \;\; l \;\; l \;\; o \;\; \ldots \;\; o]^T$$

and which confirms that the Mercury's radar controller grammar is an NSE CFG. Therefore, an FSA of the radar controller can be synthesized from the grammar of Fig. 13.

Using the Dulmage–Mendelsohn decomposition, 29 strongly connected components of the production graph of the CFG of the Mercury emitter were obtained, and the production graph is illustrated in Fig. 20. As shown in Section II, each strongly connected component of the production graph corresponds to an FSG. FSAs for each of these FSGs are shown in Figs. 21 and 22. The complete state machine of the Mercury emitter can be obtained by applying the FSA composition operation.[3]

The final step of the synthesis procedure involves transformation of the deterministic state machine of the Mercury emitter into a stochastic model, taking into account the probability distributions determined by the structure of the stochastic grammar shown in Fig. 13. At this stage, the probabilistic elements of the problem that led to the development of the stochastic radar grammar [e.g., the channel impairment probability distribution (19)] are brought into the structure of the radar model. This conversion procedure is illustrated in Section II in (12)–(15). The procedure is essentially a simple mechanical operation of converting the Mealy state machine to the Moor automaton and assigning the probabilities of transitions as shown in (13), and the probabilities of observations as demonstrated in (15).

C. Finite-State Signal Processing

The finite-state model of the radar controller grammar is based on the HMM. Once this model is obtained, the

[3]Due to the very large size of the final Mercury state machine, we do not include it in this paper.

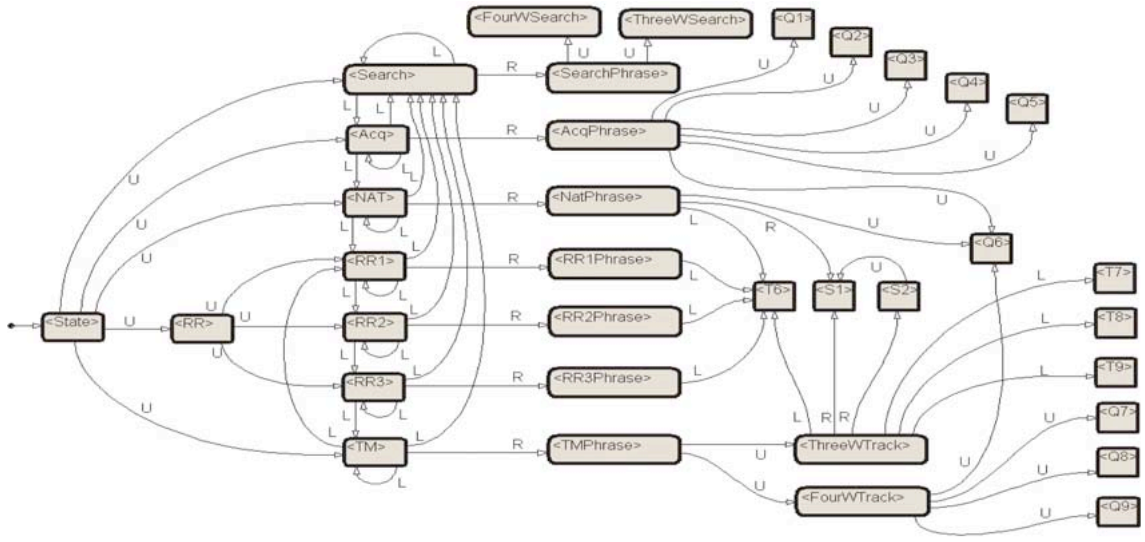


Fig. 20. ***Production graph of the Mercury grammar.***

forward-backward algorithm also known as *HMM-filter* can be applied to statistical signal processing based on this model. This algorithm is well-known and studied in detail in [27] and [44].

VI. CONCLUSION

The main idea of this paper is to model and characterize MFRs as a string generating device, where the control rules are specified in terms of an SCFG. This is unlike modeling of targets, where hidden Markov and state space models are adequate [14], [24]. The threat of the MFR is recognized as its policy of operation, and it is modeled as a Markov chain modulating the probabilities of the MFR's production rules. The paper shows how a large scale dynamical system such as MFRs is expressed by a compact representation, and demonstrates the flexibility of translating from one representation to another if the self-embedding property is satisfied. Based on the SCFG representation, a maximum likelihood sequence estimator is derived to evaluate the threat poses by the MFR, and a maximum likelihood parameter estimator is derived to infer the unknown system parameters with the EM algorithm. Since SCFGs are multitype Galton–Watson branching processes, the algorithms proposed in this paper can be viewed as filtering and estimation of a partially observed multitype Galton–Watson branching processes. For details of numerical examples of the constructed model, and the study of the predictive power (entropy) study, please see [55].

Several extensions of the ideas in this paper are worth considering.

1) The algorithms studied in this paper are inherently off-line. It is of interest to study stochastic approximation algorithms for adaptive learning of the MFR grammar and real-time evaluation of the threat. For HMMs, such real-time state and parameter estimation algorithms are well known [60].
2) In this paper we have constructed SCFG models for the MFR radar as it responds to the dynamics of a target. Recall from Fig. 6 that in this paper we are interested in electronic warfare from the target's point of view. An interesting extension of this paper that we are currently considering is optimizing the trajectory of the target to maximize the amount of information obtained from the CFG. Such trajectory optimization can be formulated as a stochastic control problem involving an SCFG (or equivalently a Galton–Watson branching process).
3) The SCFG signal processing algorithms presented in this paper consider an iid channel impairment model. It is important to extend this to Rayleigh fading channels. Sequential Monte Carlo method, such as particle filtering, can be applied to cope with fading channel.
4) In this paper we deal exclusively with optimizing the radar signal at the word level. Analogous to cross layer optimization on communication system [61], cross layer optimization can be applied to radar signal processing at the pulse and word level.
5) This paper deals exclusively with modeling and identifying MFRs in open loop. That is, we do not model the MFR as a feedback system which optimizes its task allocation in response to the target. See [62] for a Lagrangian MDP formulation of radar resource allocation. Modeling, identifying, and estimating an MFR in closed loop is a challenging task and will require sophisticated real time processing (See point 1 above). ■

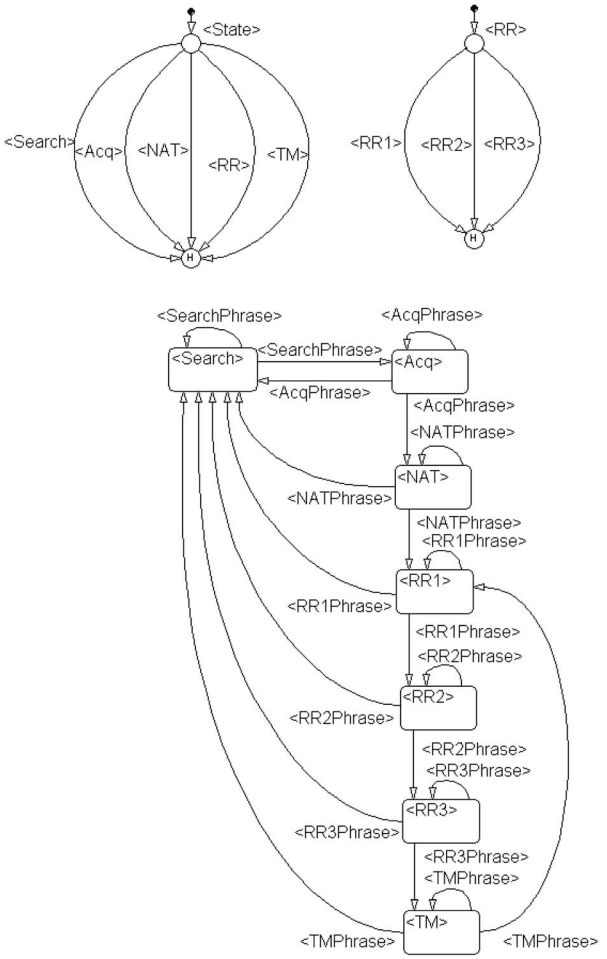


Fig. 21. ***Mercury state machine components.***

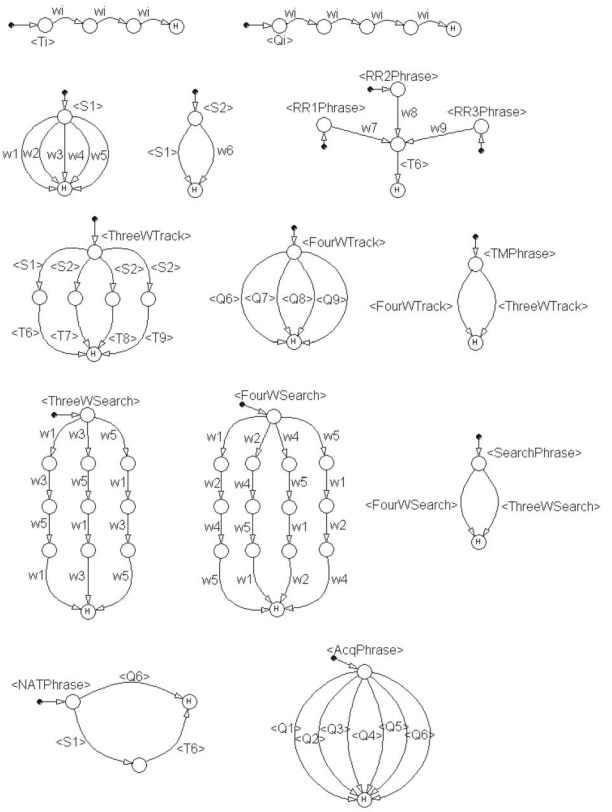


Fig. 22. ***Mercury state machine components.***

APPENDIX

A. Functional Specification of the "Mercury" Emitter

This appendix contains a sanitized version of a textual intelligence report describing the functionality of the emitter called "*Mercury*".[4]

1) General Remarks: The timing of this emitter is based on a crystal-controlled clock. Each cycle of the clock is known as a crystal count (Xc) and the associated time interval is the clock period. All leading edge emission times and dead-times can be measured in crystal counts (integer multiples of the clock period).

Most of the information below relates to search, acquisition, and tracking functions only. Missile engagement modes (launching, guidance, and fusing) can also be fit into the structure below, but with some modifications.

[4]The specification of this emitter was provided by Dr. Fred A. Dilkes of Defence R&D Canada. It is based on specifications of some real-life anti-aircraft defense radars, but has been altered and declassified before the release.

2) Radar Words: The timing of this emitter is dictated by a substructure called a *word*. Words occur sequentially in the pulse train so that one word begins as the previous word is ending. There are nine distinct words, denoted by $w_1, \ldots, w_9$. Each has the same length (on the order of several milliseconds), and is associated with a fixed integer number of crystal counts. For the purpose of this paper we will consider all words of the radar distinguishable from each other.

3) Time-Division Multiplexing—Phrases and Clauses: This system is an MFR capable of engaging five targets in a time-multiplexed fashion using structures called clauses and phrases. A phrase is a sequence of four consecutive words. A clause is a sequence of five consecutive phrases (see Fig. 23).

Each phrase within a clause is allocated to one task, and these tasks are independent of each other. For instance, the radar may search for targets using phrases 1, 3, and 4, while tracking two different targets using phrases 2 and 5.

4) Search-While-Track Scan: One of the generic functional states of the radar is a search scan denoted by <*FourWSearch*>. In the <*FourWSearch*> scan, the words

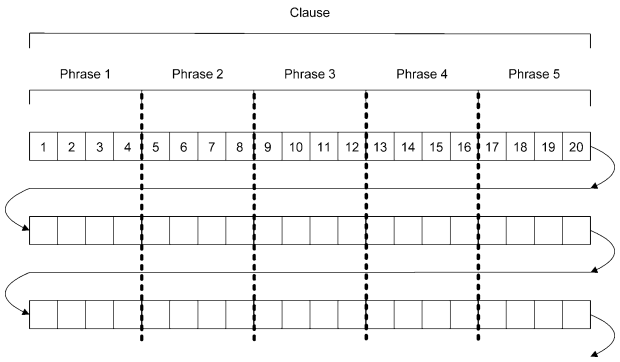


Fig. 23. ***The output sequence of this radar is formed so that the clauses follow each other sequentially. As soon as the last word of the last phrase of a clause is emitted, the first word of the first phrase of the new clause follows. Although the process is linear in time, it is very convenient to analyze the radar output sequence as a two-dimensional table when clauses are stacked together not horizontally, but vertically. In that case, boundaries of phrases associated with multiplexed tasks align, and one can examine each multiplexed activity independently by reading radar output within one phrase from top to bottom.***

are cycled through the quadruplet of words $w_1 - w_2 - w_4 - w_5$. The radar will complete one cycle (four words) for each beam position as it scans in space. This is done sequentially using all unoccupied word positions and is not dictated by the clause or phrase structure. (Note that the radar does not have to start the cycle with W1 at each beam position; it could, for instance, radiate $w_4 - w_5 - w_1 - w_2$ or any other cyclic permutation at each beam position.)

It is possible for the entire system to operate in a search-only state in which no target tracks are maintained during the search. However, <*FourWSearch*> can also be multiplexed with target tracking functions. In the latter case, some of the words within each clause are occupied by target tracking and will not engage in search functions. Only the available phrases (those that are not occupied) are cycled through the quadruplet of words. Since the number of beam positions in the scan is fixed, the rate at which the radar is able to search a given volume of space is proportional to the number of available words; as a result, simultaneous tracking increases the overall scan period.

The radar has another scan state called <*ThreeWSearch*>. This is similar to <*FourWSearch*> except that it uses only a triplet of words $w_1 - w_3 - w_5$ (and dwells on each beam position with only three words). It can also be multiplexed with automatic tracking.

5) Acquisition Scan: When the radar search scan detects a target of interest, it may attempt to initiate a track. This requires the radar scan to switch from one of the search behaviors to one of the acquisition patterns. All of the acquisition scans follow these steps sequentially.

a) *Switch from Search to Acquisition*: The switch from search to acquisition begins with all available words being converted to the same variety of word: one of $w_1, \ldots, w_6$, chosen so as to optimize to the target Doppler shift. Words that are occupied with other tracks continue to perform their tracking function and are not affected by the change from search to acquisition. The available words perform one of several scan patterns in which each beam position dwells only for the period of one word.

b) *Nonadaptive track*: Then, one of the available phrases becomes designated to track the target of interest. This designation will perpetuate until the track is dropped. Correspondingly, either the last three or all four of the words within that designated phrase become associated with the track and switch to w_6 (a nonadaptive track without range resolution). The remaining available words continue to radiate in the variety appropriate to the target Doppler.

c) *Range resolution*: At this point the radar has angular track resolution but still suffers from range ambiguities. After some variable amount of time, the first word in the designated phrase will hop between words w_7, w_8, and w_9, in no predictable order. It will dwell on each of those varieties of words only once in order to resolve the range ambiguity, but dwell-time for each variety is unpredictable.

d) *Return from Acquisition to Search*: Finally, once the radar has established track, it is ready to terminate the acquisition scan. Thereafter, until the track is dropped, either the last three or all four words of the designated phrase will be occupied with the track and will not be available for search functions or further acquisitions. The radar then returns to one of the search-while-track functions. All occupied words maintain their tracks and all available words (possibly including the first word of the designated track phrase) execute the appropriate scan pattern.

Only one acquisition can be performed at any given time.

6) Track Maintenance: Each track is maintained by either the last three or all four words of one of the phrases. Those words are considered occupied and cannot participate in search or acquisition functions until the target is dropped. The radar performs range tracking by adaptively changing amongst any of the high pulse repetition frequency words $(w_6, \ldots, w_9)$ in order to avoid eclipsing and maintain their range gates.

Occasionally, the system may perform a range verification function on the track by repeating the range resolution steps described above.

Acknowledgment

The authors would like to thank Dr. Fred Dilkes and Dr. Pierre Lavoie of the Defense Research and Development Canada for providing useful feedback on the material presented in this paper. We would like to thank Dr. Dilkes for providing the specification of the Mercury emitter that was used as an example of modeling of a realistic MFR.

REFERENCES

[1] R. G. Wiley, *Electronic Intelligence: The Analysis of Radar Signals*. Norwood, MA: Artech House, 1993.

[2] N. J. Whittall, "Signal sorting in ESM systems," *IEE Proc. F*, vol. 132, pp. 226–228, 1985.

[3] D. R. Wilkinson and A. W. Watson, "Use of metric techniques in ESM data processing," *IEE Proc. F*, vol. 132, pp. 229–232, 1985.

[4] J. Roe and A. Pudner, "The real-time implementation of emitter identification for ESM," in *Proc. IEE Colloq. Signal Processing in Electronic Warefare*, 1994, pp. 7/1–7/6.

[5] J. A. V. Rogers, "ESM processor system for high pulse density radar environments," *IEE Proc. F*, vol. 132, pp. 621–625, 1985.

[6] S. Sabatini and M. Tarantino, *Multifunction Array Radar System Design and Analysis*. Norwood, MA: Artech House, 1994.

[7] M. I. Skolnik, *Introduction to Radar Systems*. McGraw-Hill, 2002.

[8] N. Chomsky, "Three models for the description of language," *IRE Trans. Inf. Theory*, vol. IT-2, no. 3, pp. 113–124, Sep. 1956.

[9] N. Chomsky and G. A. Miller, "Finite state languages," *Inf. Control*, vol. 1, no. 2, pp. 91–112, May 1958.

[10] N. Chomsky, "On certain formal properties of grammars," *Inf. Control*, vol. 2, no. 2, pp. 137–167, Jun. 1959.

[11] N. Chomsky, "A note on phrase structure grammars," *Inf. Control*, vol. 2, no. 4, pp. 393–395, Dec. 1959.

[12] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals*. New York: Wiley-IEEE, 1999.

[13] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[14] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

[15] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Comput. Vis. Image Understanding*, vol. 73, pp. 428–440, 1999.

[16] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Patttern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 852–872, Aug. 2000.

[17] W. Zhu and J. Garcia-Frias, "Modeling of bursty channels using stochastic context-free grammars," in *Proc. 55th Vehicular Technology Conf. (VTC 2002)*, pp. 355–359.

[18] W. Zhu and J. Garcia-Frias, "Stochastic context-free grammars and hidden Markov models for modeling of bursty channels," *IEEE Trans. Veh. Technol.*, vol. 53, no. 3, pp. 666–676, May 2004.

[19] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[20] Y. Sakakibara, "Grammatical inference in bioinformatics," *IEEE Trans. Patttern Anal. Mach. Intell.*, vol. 27, no. 7, pp. 1051–1062, Jul. 2005.

[21] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[22] E. Rivas and S. R. Eddy, "The language of RNA: A formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, pp. 334–340, 2000.

[23] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[24] Y. Bar-Shalom and X. R. Li, *Estimation and Tracking: Principles, Techniques, and Software*. Norwood, MA: Artech House, 1993.

[25] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*, vol. 1, *Parsing*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[26] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley, 1979.

[27] N. Visnevski, V. Krishnamurthy, S. Haykin, B. Currie, F. Dilkes, and P. Lavoie, "Multi-function radar emitter modelling: A stochastic discrete event system approach," in *Proc. IEEE Conf. Decision and Control (CDC'03)*, pp. 6295–6300.

[28] N. A. Visnevski, F. A. Dilkes, S. Haykin, and V. Krishnamurthy, *Non-self-embedding context-free grammars for multi-function radar modeling—Electronic warfare application*, 2005, pp. 669–674.

[29] A. Wang, V. Krishnamurthy, F. A. Dikes, and N. A. Visnevski, "Threat estimation by electronic surveillance of multifunction radars: A stochastic context free grammar approach," presented at the IEEE Conf. Decision and Control, San Diego, CA, 2006.

[30] A. Wang and V. Krishnamurthy, "Threat estimation of multifunction radars: Modeling and statistical signal processing of stochastic context free grammars," presented at the IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, HI, 2007.

[31] K. Lari and S. J. Young, "The estimation of stochastic context free grammars using the inside-outside algorithm," *Comput. Speech Lang.*, vol. 4, pp. 35–56, 1990.

[32] M. I. Miller and A. O'Sullivan, "Entropies and combinatorics of random branching processes and context-free languages," *IEEE Trans. Inf. Theory*, vol. 38, no. 4, pp. 1292–1310, Jul. 1992.

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

[34] J. K. Baker, "Trainable grammars for speech recognition," in *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, 1979, pp. 547–550.

[35] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, Tools*. Reading, MA: Addison-Wesley, 1986.

[36] A. Salomaa, *Theory of Automata*. Elmsford, NY: Pergamon Press, Ltd., 1969.

[37] A. Salomaa, *Formal Languages*. New York: Academic, 1973.

[38] A. Salomaa, *Computation and Automata*. Cambridge, U.K.: Cambridge Univ. Press, 1985.

[39] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*, vol. 2, *Compiling*. Englewood Cliffs, NJ: Prentice-Hall, 1973.

[40] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. Reading, MA: Addison-Wesley, 2001.

[41] M. Anselmo, D. Giammarresi, and S. Varricchio, "Finite automata and non-self-embedding grammars," in *Proc. 7th Int. Conf. Implementation and Application of Automata (CIAA'02)*, pp. 47–56.

[42] K. S. Fu, *Syntactic Methods in Pattern Recognition*. New York: Academic, 1974.

[43] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov Models: Estimation and Control*. New York: Springer-Verlag, 1995.

[44] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1518–1569, Jun. 2002.

[45] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[46] D. C. Schleher, *Electronic Warfare in the Information Age*. Norwood, MA: Artech House, 1999.

[47] N. A. Visnevski, "Syntactic modeling of multi-function radars," Ph.D. dissertation, McMaster Univ., 2005.

[48] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley, 2001.

[49] P. L. Bogler, *Radar Principles With Applications to Tracking Systems*. New York: Wiley, 1990.

[50] T. Chiu, "The development of an intelligent radar tasking system," *Proc. 1994 2nd Australian and New Zealand Conf. Intelligent Information Systems*, 1994, pp. 437–441.

[51] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[52] Z. Chi, "Statistical properties of probabilistic context-free grammars," *Comput. Linguistics*, vol. 25, pp. 131–160, 1999.

[53] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.*, vol. 39, pp. 1–38, 1977.

[54] C. Wu, "On the convergence properties of the EM algorithm," *Ann. Stat.*, vol. 11, no. 1, pp. 95–103, 1983.

[55] A. Wang and V. Krishnamurthy, "Signal interpretation of multifunction radars: Modeling and statistical signal processing with stochastic context free grammar," *IEEE Trans. Signal Process.*, 2006.

[56] F. Dilkes and N. Visnevski, "Non-self-embedding context-free grammars for electronic warfare," Defence Research & Development Canada, Ottawa, ON, Canada, Tech. Rep. DREO TM 2004-157, Oct. 2004.

[57] M. Nederhof, *Regular Approximations of CFLs: A Grammatical View* ser. Advances in Probabilistic and Other Parsing Technologies. Boston, MA: Kluwer, 2000, ch. 12, pp. 221–241.

[58] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*, ser. EATCS Monographs on Theoretical Computer Science, W. Brauer, G. Rozenberg, and A. Salomaa, Eds. New York: Springer-Verlag, 1986.

[59] A. Pothen and C.-J. Fan, "Computing the block triangular form of a sparse matrix," *ACM Trans. Math. Softw.*, vol. 16, no. 4, pp. 303–324, Dec. 1990.

[60] V. Krishnamurthy and G. Yin, "Recursive algorithms for estimation of hidden Markov models and autoregressive models with Markov regime," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 458–476, Feb. 2002.

[61] F. Yu and V. Krishnamurthy, "Optimal joint session admission control in integrated WLAN and CDMA cellular network," *IEEE/ACM Trans. Mobile Comput.*

[62] J. Wintenby and V. Krishnamurthy, "Hierarchical resource management in adaptive airborne surveillance radars," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, no. 2, pp. 401–420, Apr. 2006.

ABOUT THE AUTHORS

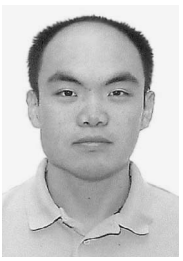
Nikita Visnevski (Member, IEEE) was born in Krasnodar, Russia, in 1974. He did his undergraduate studies in computer and software engineering at the St. Petersburg Academy of Aerospace Engineering, St. Petersburg, Russia (1991–1995). He received the M.S. degree in electrical engineering from Texas Tech University, Lubbock, in 1997 and the Ph.D. degree in electrical engineering from McMaster University, Hamilton, ON, Canada in 2005.

In 1996–1997 he worked at the Defense and Space department of the Boeing Company in Seattle, WA, and the Scientific Research Laboratory of Ford Motor Company in Dearborn, MI. During this period he focused his research on the application of artificial neural network based intelligent systems to diagnostics and control of air- and ground-based vehicles. From 1997 until 2000 he worked as an application specialist at the Controls Design Automation department of The MathWorks, Inc. (Natick, MA) where he focused on software design and development for Simulink and Real Time Workshop products. From 2000 until 2001 he worked as an application development team leader for The Solect Technology Group (Toronto, ON, Canada) where he supervised a mobile IP software development team and developed mobile Internet and voice over IP applications. Since August 2005 he has been a staff scientist and system architect at the Global Research Center of the General Electric Company, Niskayuna, NY. He is a member of the Intelligent Embedded Systems Laboratory of the Electronics and Energy Conversion Organization. His primary research focus areas include intelligent and cognitive embedded systems as well as large-scale systems architectures.

Vikram Krishnamurthy (Fellow, IEEE) was born in 1966. He received the B.S. degree from the University of Auckland, New Zealand, in 1988, and the Ph.D. degree from the Australian National University, Canberra, in 1992.

Since 2002, he has been a Professor and Canada Research Chair at the Department of Electrical Engineering, University of British Columbia, Vancouver, Canada. Prior to 2002, he was a chaired Professor in the Department of Electrical and Electronic Engineering, University of Melbourne, Australia, where he also served as Deputy Head of the department. His current research interests include stochastic modeling of biological ion channels, stochastic optimization and scheduling, and statistical signal processing. He is Coeditor (with S. H. Chung and O. Andersen) of the book *Biological Membrane Ion Channels—Dynamics Structure and Applications* (Springer-Verlag, 2006).

Dr. Krishnamurthy has served as Associate Editor for several journals including IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS AEROSPACE AND ELECTRONIC SYSTEMS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS B, IEEE TRANSACTIONS ON NANOBIOSCIENCE, and SYSTEMS AND CONTROL LETTERS.



Alex Wang was born in 1979 in Taiwan. He received the B.S. degree (with honours) in engineering physics with a commerce minor and the M.S. degree in electrical and computer engineering from the University of British Columbia, Vancouver, Canada, in 2003 and 2005, respectively. He is currently working toward the Ph.D. degree in statistical signal processing, under the supervision of Dr. Vikram Krishnamurthy, at the University of British Columbia.

His research interests include radar signal processing, syntactic pattern recognition, and uncertain reasoning.

Simon Haykin (Fellow, IEEE) received the B.Sc. (First-class Honours), Ph.D., and D.Sc. degrees in electrical engineering from the University of Birmingham, Birmingham, U.K.

Currently, he holds the title of Distinguished University Professor in the ECE Department at McMaster University, Hamilton, ON, Canada. He is the author of numerous books, including the most widely used books: *Communication Systems* (4th ed., Wiley), *Adaptive Filter Theory* (4th ed., Prentice-Hall), *Neural Networks: A Comprehensive Foundation* (2nd ed., Prentice-Hall), and the newly published *Adaptive Radar Signal Processing* (Wiley), as well as numerous refereed journal papers.

Dr. Haykin is a Fellow of the Royal Society of Canada, recipient of the Honourary Degree of Doctor of Technical Sciences from ETH, Zurich, Switzerland, and the Henry Booker Gold Medal from URSI, as well as other prizes and awards.