INCREMENTAL LEARNING OF STOCHASTIC GRAMMARS WITH GRAPHICAL EM IN RADAR ELECTRONIC SUPPORT

Guillaume Latombe and Eric Granger[1]

Laboratoire d'imagerie, de vision et d'intelligence artificielle
Dépt. de génie de la production automatisée
École de technologie supérieure
Montreal, Canada

Fred A. Dilkes

Defence R&D Canada – Ottawa
Dept. of National Defence
Ottawa, Canada

ABSTRACT

Although Stochastic Context-Free Grammars (SCFGs) appear promising for recognition of radar emitters, and for estimation of their level of threat in Radar Electronic Support (ES) systems, well-known techniques for learning their production rule probabilities are computationally demanding, and cannot efficiently reflect changes in operational environments. Some techniques have been proposed for fast learning of SCFGs probabilities, yet, of those, only the HOLA technique can perform learning incrementally. In this paper, two incremental versions of the graphical EM (gEM) technique are proposed. The incremental gEM (igEM) and on-line incremental gEM (oigEM) allow for adapting production rule probabilities from new data, without having to retrain from the start on all accumulated training data. These new techniques are compared to HOLA using radar signal data. An experimental protocol has been defined such that the impact on performance of factors like the size of new data blocks for incremental learning, and the level of ambiguity of MFR grammars, may be observed. Results indicate that, contrary to HOLA, incremental learning of training data blocks with igEM and oigEM provides the same level of accuracy as learning from all cumulative data from scratch, even for small data blocks. As expected, incremental learning significantly reduces the overall time and memory complexities.Finally, it appears that while the computational complexity and memory requirements of igEM and oigEM may be greater than that of HOLA, they both provide a higher level of accuracy.

Index Terms— Radar electronic support, pattern recognition, stochastic grammars, incremental machine learning, graphical EM

1. INTRODUCTION

Radar Electronic Support (ES) involves the passive search for, interception, location, analysis and identification of radiated electromagnetic energy for military purposes. ES thereby provides valuable information for real-time situation awareness, threat detection, threat avoidance, and timely deployment of counter-measures [10]. Two critical functions of ES are the recognition of radar emitters associated with intercepted pulse trains, and the estimation of the instantaneous level of threat posed by these radars. The recent proliferation and complexity of electromagnetic signals encountered in modern environments is greatly complicating these functions.

[1]**Corresponding author**. École de technologie supérieure, 1100 rue Notre-Dame Ouest, H3C 1K3, Montreal, Qc, tel.: 1-514-396-8650, fax: 1-514-396-8650, e-mail: eric.granger@etsmtl.ca.

In conventional ES systems, radar signals are typically recognized using temporal periodicities within the pulse train in conjunction with histograms of the pulses in some parametric space, *e.g.*, carrier frequency, pulse repetition frequency, and pulse width. With the advent of automatic electronic switching designed to optimize radar performance, modern radars, and especially multi-function radars (MFR), are usually far too complex to be recognized in this way. MFRs will continuously and autonomously change their transmitted signals in response to various events in their dynamically-changing environment. In order to exploit the dynamic nature of many modern radar systems, advanced signal processing algorithms based on Stochastic Context Free Grammars (SCFGs) have been proposed for modeling the behavior of radar systems [9]. Such models can allow tracking the dynamic behaviour of radar emitter patterns, which can be exploited for recognition of radar emitters, and for estimation their respective level of threat.

Given prior knowledge of a radar's behavior, and a set of training sequences collected in the field, one challenge to the practical application of SCFGs is the task of learning probability distributions associated with the production rules. The most popular technique for learning the production rule probabilities is the Inside-Outside algorithm (IO) [1], but its application to real-world tasks is limited due to its time and memory complexity per iteration, and convergence time. Moreover, this technique cannot incrementally learn new information that may emerge as the operational environment evolves. In radar ES applications, new information from a battlefield or other sources often becomes available in blocks at different times. Incremental learning of SCFG probabilities is therefore an undisputed asset. In this context, incremental learning refers to the ability to adapt SCFG probabilities from new blocks of training sequences, without requiring access to training sequences used to learn the existing SCFG. To accelerate IO, Sato and Kameya [8] have developed a batch learning algorithm, called graphical EM (gEM), that re-estimates the probabilities of SCFG using a special arrangement of the data into support graphs, based on the result from a chart parsing algorithm. In addition, Oates and Heeringa [6] introduced an incremental learning algorithm called HOLA that optimizes the relative entropy between the distribution of rules obtained after having parsed the training set, and the distribution of rules obtained after having parsed a set generated by the grammar.

In this paper, two novel incremental derivations of the original gEM algorithm are proposed. The first one, called incremental gEM (igEM), is based on research by Neal and Hinton [4], whereas the second one, called on-line igEM (oigEM), is based on research by Sato and Ishii [7]. Both algorithms are compared to HOLA [6].

Given the need for a learning procedure that offers both accurate results and computational efficiency, the performance of these techniques is examined from several perspectives – perplexity, convergence time, time complexity, and memory complexity. The data sets used in our proof-of-concept computer simulations describe electromagnetic signals transmitted from fictitious multifunction radar systems. An experimental protocol has been defined such that the impact on performance of factors like the size of new data blocks for incremental learning, and the level of ambiguity of grammars, may be observed. Simulation results and complexity estimates are analyzed with ES applications in mind.

The rest of this paper is structured into four sections as follows. The next section provides some background information on grammatical modeling in the context of radar ES applications. In Section 3, the main features of igEM and oigEM are outlined. Then, the experimental protocol, data sets, and performance measures, used to compare these techniques are described in Section 4. Finally, the results of computer simulations and complexity estimates are presented and discussed in Section 5.

2. GRAMMATICAL MODELING IN RADAR ES

In radar ES applications, pulsed radar signals are generated by a MFR in reaction to its current operating environment. The algorithm controlling the function of a MFR is designed according to stochastic automata principles, and the state transitions within the automata are driven by the stochastic behavior of the targets [9]. It is often possible to partition a radar signal into words, which can be defined as static or dynamically-varying groups of pulses that a MFR emits in different states. When the radar is in a given state, it emits a particular sequence of words, whose number and structure vary according to the MFR.

At a word level, most MFR systems of interest have a natural and compact description in terms of a type of grammar called Context-Free Grammars (CFG) [2]. A CFG G consists of a vocabulary of *terminal* symbols along with a set of *non-terminal* symbols, and a set of *production rules*. Each production rule represents a substitution of the form $A \rightarrow \lambda$ where A is a nonterminal symbol and λ is a specified sequence containing terminal symbols, nonterminal symbols or both. A sequence of terminal symbols is in the *language* defined by G if it can be generated from a uniquely specified *starting non-terminal* by some sequence of substitutions permitted by the production rules. Grammatical modeling of a radar system's behavior is achieved if one assumes that symbols of the vocabulary correspond to words of a specific MFR, and that the language represents all possible combination of sequences that a radar could ever emit.

To allow for robust modeling of the random events, signal degradations, noise and other sources of uncertainty, an element of stochasticity can be introduced. A Stochastic CFG (SCFG) G_s is a CFG in which each production rule is assigned a *production probability* $\theta(A \rightarrow \lambda) \geq 0$, such that $\sum_\lambda \theta(A \rightarrow \lambda) = 1$ for $\forall A$. This results in a probability distribution over the generated language.

3. TECHNIQUES FOR FAST INCREMENTAL LEARNING OF SCFG PROBABILITIES

Suppose that Ω is a specified *training set* of sequences of terminal symbols. The problem of learning production probabilities of a SCFG G_s from Ω can be naturally formulated as a maximization of the joint likelihood $P(\Omega, \Delta_\Omega|G_s) = \prod_{x\in\Omega} P(x, \Delta_x|G_s)$ where Δ_x represents the set of derivation trees over a sequence $x \in \Omega$, and $P(x, \Delta_x|G_s)$ represents its overall probability. Here, Δ_Ω represents the set of all derivation trees over the sequences in Ω.

The Expectation-Maximization (EM) algorithm can be applied to learn SCFG probabilities [5] by optimizing $P(\Omega, \Delta_\Omega|G_s)$ with respect to the production probabilities $\theta(A \rightarrow \lambda)$. The E-step consists of computing the frequency $N(A \rightarrow \lambda, d_x)$ of each rule in each derivation tree $d_x \in \Delta_x$, and the probability of each derivation tree $P(x, d_x|G_s)$, in order to evaluate:

$$\eta(A \rightarrow \lambda) = \sum_{x\in\Omega} \frac{\sum_{d_x\in\Delta_x} N(A \rightarrow \lambda, d_x) P(x, d_x|G_s)}{P(x, \Delta_x|G_s)} \quad (1)$$

The M-step uses η to re-estimate the probabilities:

$$\theta'(A \rightarrow \lambda) = \frac{\eta(A \rightarrow \lambda)}{\sum_\mu \eta(A \rightarrow \mu)} \quad (2)$$

Note that η represents a vector of sufficient statistics for $\theta(A \rightarrow \lambda)$, meaning that all information about $\theta(A \rightarrow \lambda)$ is contained in $\eta(A \rightarrow \lambda)$.

The graphical EM algorithm (gEM), proposed by Sato [8] uses a special arrangement of the results from a CYK or Earley chart parser applied on the training data set, called support graphs, to accelerate the re-estimation of production rules. During the E-step, gEM computes $\eta(A \rightarrow \lambda)$ using the support graphs instead separately of computing the different elements of Eq. 1. The M-step simply applies Eq. 2. The rest of this section presents two derivations of gEM that are suitable for incremental learning in radar ES applications.

Incremental gEM (igEM): Neal and Hinton [4] have introduced an algorithm called *incremental EM*. According to this algorithm, the original dataset Ω is divided into blocks Ω_i, for $i = 1, ..., n$, and, for each block, the vectors of sufficient statistics are initialized to an initial guess η_i. After selecting a block Ω_i to be updated, the E-step computes η_i' on this block, sets $\eta_j' = \eta_j$ for $j \neq i$, and computes $\eta' = \sum_{k=1}^n \eta_k'$. The M-step re-estimates the probabilities that optimize likelihood given η'. Note that this algorithm is not incremental in the sense considered in this work, since all the data Ω must be available for the M-step. Blocks of data for training are selected either sequentially, at random, or through a special scheme for which the algorithm has not yet converged. The new igEM algorithm is a fully incremental approximation of the above algorithm applied to SCFG learning. Consider that the SCFG probabilities have previously been learned from a block of sequences Ω_1, and that the final value of η, referred to as η_1, is stored. Then, to learn a new block Ω_2, the E-step determines η from Ω_2, and then computes $\eta' = \eta + \eta_1$. The M-step re-estimates the probabilities θ using Eq. 2. Once convergence is reached, $\eta_2 = \eta'$ is stored, and another block may be learned.

On-line incremental gEM (oigEM): Sato and Ishii [7] have proposed an on-line version of the EM algorithm for normalized Gaussian networks. The batch EM algorithm for this task involves re-estimating a parameter using a combination of the weighted means f^* over n samples for specific functions f:

$$f^* = \frac{1}{n} \sum_{i=1}^n f(x_i) P(Z_i|x_i, \theta) \quad (3)$$

where $\{x_i\}$ is a set of observed variables and $\{Z_i\}$ is a set of unobserved ones. In our context, they correspond to the sequences and the associated derivation trees, respectively. The weighting probability $P(d_x|x, \theta)$ then corresponds to the probability of having a particular derivation tree d_x, knowing x and θ. The on-line version of this algorithm consists in computing an approximation of the weighted mean on the $i + 1^{th}$ sample based on the one on the i^{th} sample, according to:

$$\bar{f}^{(i+1)} = \bar{f}^{(i)} + \chi(i+1) \left(f(x_{i+1}) P(Z_{i+1}|x_{i+1}, \theta(i)) - \bar{f}^{(i)} \right) \quad (4)$$

where χ is a learning rate. This principle may be applied to gEM by identifying f with the frequency of the rules, N. Indeed, Bayes' theorem gives the relation between N^* and η for a particular production rule $A \rightarrow \lambda$:

$$N^*(A \rightarrow \lambda) = \frac{1}{|\Omega|} \sum_{x \in \Omega} \sum_{d_x \in \Delta_x} N(A \rightarrow \lambda, d_x) P(d_x|x, G_s) = \frac{\eta(A \rightarrow \lambda)}{|\Omega|} \tag{5}$$

Note that this algorithm is a sequential on-line algorithm that should be applied to each sequence of the training dataset, and from one iteration to the next until convergence is attained. The new oigEM algorithm is an adaptation of the above algorithm to incremental learning of SCFG probabilities. Suppose that training has already been successfully performed on the first block Ω_1, and that, for each rule $r = A \rightarrow \lambda$, $\tilde{N}(r)^{(1)}$ – the final value of $\tilde{N}(r)$ – is stored. In order to learn a new block Ω_2, $\tilde{N}(r)^{(2)}$ can be computed according to:

$$\tilde{N}(r)^{(i+1)} = \tilde{N}(r)^{(i)} + \chi(i+1) \left(\frac{\eta_{i+1}(r)}{|\Omega_{i+1}|} - \tilde{N}(r)^{(i)} \right) \tag{6}$$

and probabilities are re-estimated by replacing η by $\tilde{N}$ in Eq. 2.

The main difference between igEM and oigEM lies in the fact that oigEM is parametrized by learning rate χ. This parameter allows to tune the algorithm to be more or less adaptive for new information while training a SCFG. By setting $\chi(i+1) = \frac{\chi(i)}{1+\chi(i)}$, igEM and oigEM become identical.

4. EXPERIMENTAL METHODOLOGY

In order to characterize the performance of the techniques presented in Section 3, two fictitious MFR systems called Mercury and Pluto were considered [3]. The Mercury MFR can be in one of five functional states – Search (`S`), Acquisition (`Acq`), Non-Adaptive Track (`Na`), Range Resolution (`Rr`), and Track Maintenance (`Tm`). If the radar is in `S`, it can remain there, or move to `Acq` once a target is detected. The target acquisition cycle involves transitions from `Acq`, to `Na`, to `Rr`, and finally to `Tm`. The radar can remain in any of these states indefinitely. `Acq` or `Tm` can be abandoned at any point, at which time the radar returns to `S`. The Pluto MFR operates exactly the same way, except that it does not pass by the `Acq` state. A word-level CFG was designed for each MFR from its functional description. Mercury and Pluto respectively represent low- and high-ambiguity grammars.

The transition probabilities needed to design a SCFG were learned according to either igEM, oigEM and HOLA [6] techniques through computer simulation. A synthetic radar data set was generated for each MFR. It consisted of 300 sequences, each corresponding to a sequence of words that might be produced during one target detection, while switching through all internal states, starting and ending in `S` state. Mercury sequences had a size ranging from 108 to 1540 words, with an average of 588 words, while Pluto sequences had a size that ranged from 397 to 953 words, with an average of 644 words. The duration of each state was set using gaussian distributions. Prior to simulation trials, the data set for each MFR was partitioned into four equal parts - a training subset `Train` (Ω), a validation subset `Val`, and a test subset `Test`. Finally, the `Train` set for each MFR was subdivided into a certain number of blocks Ω_i of either 5, 10, 20, 25 or 50 sequences.

During each simulation trial with a block size $|\Omega_i|$, training was performed over several training iterations with the first block of data, until the log-likelihood for two consecutive iterations was lower than 0.001. Over-training was avoided using `Val`. Once convergence is reached for the first block, the second one is learned in the same way, and so on. For a new block, each trial was replicated for 10 different random initializations of the probabilities.

The average performance of techniques was compared in terms of the amount of resources required during training, measured with the overall time (T_{tot}) and memory (M_{tot}) complexity and the convergence time (I), and the accuracy of results, assessed using the perplexity (PP) – on the test sets. Definitions of these measures can be found in [3].

5. RESULTS AND DISCUSSION

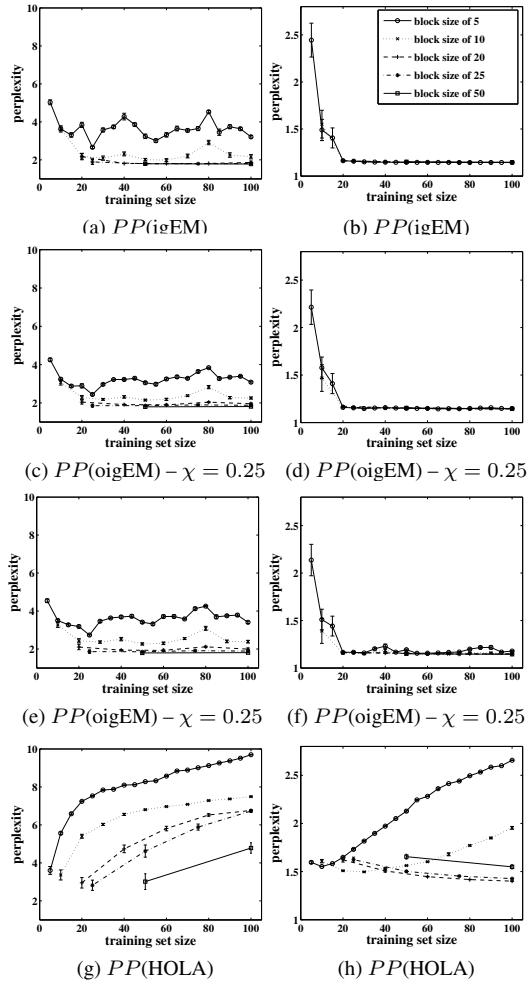
Fig. 1 shows the average perplexity achieved by a SCFG obtained by incremental learning with igEM, oigEM and HOLA for different block sizes $|\Omega_i|$ of the Mercury and Pluto data. Assuming that the environment is static, and that a block Ω_i is representative of the environment, performance depends heavily on its size $|\Omega_i|$. If Ω_i is large enough, the SCFG will be well defined. With Mercury data, the perplexity of a SCFG obtained from either igEM or oigEM tends towards the behaviour that could be achieved through batch learning on one single cumulative block of data, even for small block sizes. It appears that $|\Omega_i|$ = 25 sequences is required to reach the lowest possible perplexity. At that point, incremental learning with igEM or oigEM gives the same performance as batch learning with gEM[1]. In contrast, the perplexity of a SCFG obtained with HOLA never tends towards that of a SCFG trained though learning on a one single cumulative block.

The ambiguity of the Pluto grammar is greater, yet its complexity is lower. With Pluto data, the perplexity of a SCFG obtained by incremental learning with either igEM or oigEM also tends towards the behaviour of a SCFG obtained by batch learning with gEM. It appears that a $|\Omega_i|$ = 5 sequences is sufficient to reach the lowest possible perplexity. At that point, incremental learning with igEM or oigEM gives the same result as batch learning with gEM. In contrast, HOLA requires at least a training block size of 20 sequences to converge. The fact that HOLA converges for Pluto and not for Mercury may be linked to the lower number of production rules with Pluto. Tuning the learning rate χ of oigEM does not have a significant impact on the overall perplexity when using training blocks larger than 10 sequences. For training block sizes of 5 and 10 sequences, it appears that the perplexity is less stable if χ is high. Indeed, increasing the value of χ assign a greater importance to the new data in the learning process.magnitude.

The average convergence time on the Mercury data ranges from 4.6 to 5.2 iterations across block sizes for each algorithm. With the Pluto data (whose grammar is more ambiguous than that of Mercury), some significant differences appear. Increasing the block sizes from 5 to 50 multiply the convergence time by 2 for igEM and oigEM, while it divides it by 3 for HOLA. Finally, igEM and oigEM have similar convergence times when $\chi = 0.25$, but it tends to increase as χ grows beyond 0.25, since the higher χ is, the more probabilities are adjusted to new data.

The difference between gEM, igEM and oigEM lies only in the re-estimation equation, and has no influence on T and M, which are the same for these three algorithms, and have already been studied in [3]. An overall measure of the complexity needed to learn a block of training sequences would however reveal the impact on performance of incremental learning. The overall time and memory complexities needed to learn a block Ω_i are $T_{tot} = T \cdot \Gamma_T \cdot I$ and

[1]Note that igEM and oigEM are equivalent to gEM when used for batch learning of a single block of data. Therefore, the first point of each igEM and oigEM curve in Fig. 1 corresponds to the performance of gEM. That is one can compare the perplexity of a SCFG trained with gEM on one block of n sequences to that of a SCFG trained with igEM or oigEM on two blocks on $n/2$ sequences.



(a) PP(igEM) (b) PP(igEM)

(c) PP(oigEM) – $\chi = 0.25$ (d) PP(oigEM) – $\chi = 0.25$

(e) PP(oigEM) – $\chi = 0.25$ (f) PP(oigEM) – $\chi = 0.25$

(g) PP(HOLA) (h) PP(HOLA)

Fig. 1: Average perplexity versus training set size of SCFGs obtained by learning of blocks of Mercury data – (a), (c), (e), and (g) – and of Pluto data – (b), (d), (f), and (h). (Error bars are standard error of the sample mean).

$M_{tot} = M \cdot \Gamma_M \cdot I$, where Γ_T and Γ_M are multiplicative factors that depend on the size of $|\Omega_i|$, and T and M are the time complexity per iteration and the memory complexity. Consider that training has already been completed on a dataset Ω for gEM, divided into blocks Ω_i for the incremental algorithms, and that each new block was available once training was completed on the previous ones. Γ_T and Γ_M are presented in Table 1. Results indicate that incremental learning provides a considerable saving in terms of computational resources. In addition HOLA provides the lowest overall time and space complexities as it is bounded by the numbers of SFCG rules and is independent of Ω_i.

6. CONCLUSION

In this paper, two incremental alternatives to the gEM technique have been proposed for learning the production rule probabilities of SCFGs, in order to recognize multi-function radar (MFR) systems and estimate their states in ES applications. These two techniques are the incremental gEM (igEM), and the on-line incremental gEM (oigEM). As the original gEM, they rely on the pre-computation of data structures to accelerate the probability re-estimation process. However, they also allow integration of new blocks of training sequences, without re-training from the beginning using all previously accumulated data. The performance of these two techniques has been compared with HOLA in terms of resource allocation and accuracy. An experimental protocol has been defined to assess impact on performance of different sizes of the successive training blocks, and the level of ambiguity of MFR grammars. Proof-of-concept computer simulations have been performed on using synthetic radar data from different types of MFR systems.

	gEM	igEM and oigEM	HOLA
Γ_T	$\sum_{j=1}^{n}\sum_{i=1}^{j}\lvert\Omega_i\rvert$	$\sum_{i=1}^{n}\lvert\Omega_i\rvert$	constant
Γ_M	$\sum_{i=1}^{n}\lvert\Omega_i\rvert$	$\lvert\Omega_n\rvert$	constant

Table 1: Overall time and memory complexities.

Results indicate that incremental learning of data bocks with igEM and oigEM provides the same level of accuracy as learning from all cumulative data from scratch, even for small data blocks. As expected, incremental learning significantly reduces the overall time and memory complexities. The igEM and oigEM algorithms systematically provide a higher level of accuracy than HOLA, especially for small block of data. Unless the MFR system is modeled by a very ambiguous SCFG, these techniques can learn probabilities rapidly, at the expense of significantly higher memory resources. The execution time and memory requirements of HOLA are orders of magnitude lower than that of igEM and oigEM.

7. REFERENCES

[1] J. K. Baker. Trainable Grammars for Speech Recognition. In *Speech Communications Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, June 1979.

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

[3] G. Latombe, E. Granger, and F. A. Dilkes. Fast Incremental Learning of Grammatical Probabilities in Radar Electronic Support. Technical report, Defence R&D Canada - Ottawa, August 2006.

[4] R. M. Neal and G. E. Hinton. A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants. *Learning in graphical models*, pages 355–368, 1999.

[5] F. Nevado, J. A. Sanchez, and J. M. Benedi. Combination of Estimation Algorithms and Grammatical Inference Techniques to Learn Stochastic Context-Free Grammars. In *5th Int. Col., ICGI 2000, Proceedings*, pages 164–171, Lisbon, Portugal, 11-13 September 2000.

[6] T. Oates and B. Heeringa. Estimating Grammar Parameters Using Bounded Memory. In H. Fernau P. Adriaans and M. van Zaanen, editors, *Proc. of the Sixth Int. Col. on Grammatical Inference (ICGI)*, pages 185–198, Springer-Verlag Heidelberg, 2002.

[7] M. Sato and S. Ishii. On-line EM Algorithm for the Normalized Gaussian Network. *Neural Comp.*, 12(2):407–432, 2000.

[8] T. Sato and Y. Kameya. Parameter Learning of Logic Programs for Symbolic-Statistical Modeling. *Journal of Artificial Intelligence Research*, 2:391–454, 2001.

[9] N. Visnevski. *Syntactic Modeling of Multi-Function Radars*. PhD thesis, McMaster University, http://soma.ece.mcmaster.ca/~visnev/Resources/PHDThesis/, 2005.

[10] R. G. Wiley. *Electronic Intelligence: the Analysis of Radar Signals, 2nd ed.* Norwood, MA:Artech House, 1993.