Syntactic Modeling of Multi-Function Radars

By

Nikita A. Visnevski, M.Sc., B.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

©Copyright by Nikita Visnevski, July 2005

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Syntactic Modeling of Multi-Function Radars

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

DOCTOR OF PHILOSOPHY (2005) McMaster University

(Department of Electrical and Comuter Engineering) Hamilton, Ontario

TITLE: Syntactic Modeling of Multi-Function Radars

AUTHOR: Nikita Visnevski, M.Sc., B.Sc.

SUPERVISOR: Simon Haykin

NUMBER OF PAGES: xv, 144

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Abstract

The problem of radar modeling is of critical importance to Electronic Warfare applications such as radar recognition and threat analysis. As modern radar signals become steadily more complex, so do the issues associated with radar modeling and signal processing. Traditional, data-centric approaches to radar signal processing can no longer cope with the increasing complexity of radar signals. The main contribution of this thesis is the novel, model-centric approach to radar signal processing that utilizes methods from the theory of formal languages and syntactic pattern recognition.

In this thesis, we focus our attention on modeling of Multi-Function Radars (MFRs) – the class of radars that currently presents the greatest challenges to the radar signal processing community. The characteristic feature of MFRs is the complex hierarchical signal structure often utilized by these radars. This complexity in MFR signals makes the classic radar signal processing techniques inadequate.

We consider MFRs as stochastic discrete event systems that are "communicating" information using some stochastic formal languages. We then show how these languages can be modeled by grammars that can be derived using *a priori* information available in the databases of electronic intelligence. We also demonstrate how these grammars can capture the complex MFR signal structures and exploit the relationships between the internal processes within MFRs and signals emitted by these radars. We refer to this MFR modeling approach as "syntactic modeling".

We also take advantage of the hierarchical nature of the MFR signals and develop a layered radar model where processing of related features of radar signals is confined to a certain modeling layer, and only the information relevant to the next layer of radar signal processing in propagated forward. This hierarchical radar modeling approach enables to keep the complexity of MFR models manageable.

We demonstrate the applicability of the developed approach using computer simulations of synthetic MFR signals and provide two complete case studies demonstrating how the principles developed in this thesis can be applied to modeling of real-life MFRs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Acknowledgements

I am deeply indebted to my supervisor, Prof. Simon Haykin, for providing continuous support and guidance over the entire period of my study. I would also like to express my gratitude to Dr. T. Kirubarajan and Dr. Tamás Terlaky for agreeing to take part in my Ph. D. committee and for providing me with important and useful feedback on my progress.

I am also thankful to the Canadian Department of Defence and the Ottawa branch of Defence Research & Development Canada for their financial support and continuous guidance. Specifically, I would like to thank Dr. Pierre Lavoie and Dr. Fred A. Dilkes for their close involvement and assistance in this work. Without the specifications of multi-function radars provided by Dr. Dilkes this thesis would not be possible.

I am deeply grateful to Dr. Vikram Krishnamurthy of the University of British Columbia for his part in this project and for hosting me on several occasions at his laboratory in Vancouver. Dr. Krishnamurthy's input helped shape the direction of this project and greatly influenced my views of the subjects of modeling and signal processing.

I would like to thank Brian Currie for his hard work and dedication to building the multi-function radar simulator. Data generated by this simulator was essential to the pulse train analysis experiments presented in this thesis. Brian's effort in DRDC contract management and his active participation in software design and analysis ensured continuous and steady progress of our research effort.

Last but by no means least, I am deeply thankful to Diana Zaft for proof-reading and correcting this thesis and for continuous emotional support.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Contents

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables x

List of Algorithms xi

List of Acronyms xii

List of Contributions to the Literature xiv

1 Introduction 1
- 1.1 Background . . . 2
- 1.2 Multi-Function Radars . . . 3
- 1.3 The need for model-centric principle . . . 4
  - 1.3.1 Deficiencies in modern Electronic Intelligence . . . 4
  - 1.3.2 The model-centric principle . . . 6
  - 1.3.3 The model-centric Electronic Intelligence . . . 8
- 1.4 Syntactic Multi-Function Radar modeling . . . 8
  - 1.4.1 What does "syntactic" mean? . . . 9
  - 1.4.2 The essence of syntactic radar modeling . . . 10
- 1.5 Overview of the thesis . . . 10
  - 1.5.1 Main contributions . . . 10
  - 1.5.2 Scope and assumptions . . . 12
  - 1.5.3 Organization . . . 13

I Essential elements of the model-centric approach 15

2 Hierarchical model of Multi-Function Radar signals 16
- 2.1 "Divide and conquer" principle . . . 17

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.2 Layered radar signal architecture 17
2.3 Summary 21

3 Elements of syntactic modeling 22
3.1 Formal languages 24
3.2 Transformational grammars 24
3.3 Finite-State Languages and Automata 27
3.4 Context-Free Languages and Context-Free Grammars 29
3.5 Stochastic languages and stochastic grammars 32
3.6 Summary 35

4 Markov chains and Hidden Markov Models 37
4.1 Markov chains 37
4.2 Hidden Markov Models 39
4.3 Summary 44

II Syntactic radar modeling 46

5 Word-level modeling of Multi-Function Radars 47
5.1 Verification of Non-Self-Embedding 49
5.2 Grammatical decomposition 51
5.3 Synthesis of finite-state components 53
5.4 Composition of the Finite-State Automaton 55
5.5 Summary 57

6 Pulse-level modeling of Multi-Function Radars 59
6.1 The problem of pulse train analysis 60
6.2 Syntactic pulse-level model 63
6.3 Summary 66

7 Model-centric Electronic Threat Libraries 67
7.1 Derivation constraints for MFR grammars 68
7.2 Design cycle for the model-centric ETL 69
7.3 Model accuracy considerations 72
7.4 Summary 75

III Experiments and Applications 76

8 Word-level modeling experiments 77
8.1 Case study: Mercury emitter 77
8.1.1 Mercury phrase structure grammar 77
8.1.2 Mercury state machine synthesis 85

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8.2 Case study: Pluto emitter 91
8.2.1 Pluto phrase structure grammar 91
8.2.2 Pluto state machine synthesis 98
8.3 Summary 103

9 Pulse train analysis application **106**
9.1 Application of the Viterbi Algorithm 107
9.2 Simulation Results 109
9.3 Summary 113

10 Discussion **114**
10.1 Contributions to the literature 114
10.2 Conclusion 116

A Functional specification of the "Mercury" emitter **118**

B Functional specification of the "Pluto" emitter **124**

C Event-driven pulse train analysis **129**

Bibliography **134**

Nomenclature **140**

Index **142**

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

2.1 A three-layer hierarchical radar signal structure 18
2.2 Radar output sequence 19
2.3 Layered signal structure of another radar 20

3.1 The Chomsky hierarchy of formal languages. 26
3.2 Example of a finite-state automaton 29
3.3 Production graph 31
3.4 Derivation sequence for stochastic grammars 34

4.1 Example of a Markov chain 38
4.2 Example of a Hidden Markov Model 40

5.1 Production graph 50
5.2 Strongly-connected components of the production graph 52
5.3 Components of the finite-state automaton 54
5.4 Synthesis of the Finite-State Automaton 57

6.1 Sample Multi-Function Radar words 60
6.2 Binary channel model 62
6.3 Pulse interval Markov chain 64

7.1 Cyclic and acyclic components of the production graph 68
7.2 Design cycle for the model-centric Electronic Threat Libraries . 70
7.3 System analysis considerations 73

8.1 High level functionality of the Mercury emitter 80
8.2 Deterministic grammar of the Mercury emitter 82
8.3 Stochastic channel impairment model 83
8.4 Characteristic grammar of the Mercury emitter 84
8.5 Weighted grammar of the Mercury emitter 86
8.6 Production graph of the Mercury grammar 88
8.7 Transition matrix for the Mercury grammar 89
8.8 Production graph analysis matrix for the Mercury grammar . . 90
8.9 Mercury state machine components (Part 1) 93

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8.10 Mercury state machine components (Part 2) 94
8.11 High level functionality of the Pluto emitter 96
8.12 Deterministic grammar of the Pluto emitter 97
8.13 Characteristic grammar of the Pluto emitter 99
8.14 Weighted grammar of the Pluto emitter 100
8.15 Production graph of the Pluto grammar 101
8.16 Transition matrix for the Pluto grammar 102
8.17 Production graph analysis matrix for the Pluto grammar 102
8.18 Pluto state machine components 104

9.1 Example of the trellis path of the Viterbi algorithm 109
9.2 Pulse train analysis simulation results 112

A.1 Signal structure of the Mercury radar 120
A.2 Mercury output sequence 121

B.1 Signal structure of the Pluto radar 126

C.1 Event-driven pulse train analysis principle 132
C.2 Pulse train analysis simulation results 133

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

3.1 Deterministic grammars, production rules, and languages. 27

5.1 Semi-ring operations of sum and product. 50

8.1 Mercury emitter phrase structure 79
8.2 Connected components of the Mercury production graph 92
8.3 Pluto emitter phrase structure 95
8.4 Connected components of the Pluto production graph 105

9.1 Summary of the pulse train analysis experiments 111

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Algorithms

1 Procedure "createFSA" 56
2 Function "expandFSA" 56

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Acronyms

AAD	Anti-aircraft Defence
CFG	Context-Free Grammar
CFL	Context-Free Language
CSG	Context-Sensitive Grammar
CSL	Context-Sensitive Language
DES	Discrete Event System
ELINT	Electronic Intelligence
ETL	Electronic Threat Libraries
EW	Electronic Warfare
FSA	Finite-State Automata
FSG	Finite-State Grammar
FSL	Finite-State Language
HMM	Hidden Markov Model
LPI	Low-Probability-of-Intercept
MFR	Multi-Function Radar
MIMO	Multiple Input Multiple Output
NAT	Non-Adaptive Track
NSE	Non-Self-Embedding
OOM	Observable Operator Model
PAA	Phased Array Antenna

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

PPI	Pulse to Pulse Interval
PRI	Pulse Repetition Interval
PRF	Pulse Repetition Frequency
RG	Regular Grammar
RL	Regular Language
RWR	Radar Warning Receiver
SCFG	Stochastic Context-Free Grammar
SDR	Software-Defined Radio
SFSG	Stochastic Finite-State Grammar
SFSL	Stochastic Finite-State Language
SRL	Stochastic Regular Language
TDM	Time Division Multiplexing
TM	Track Maintenance
TOA	Time-Of-Arrival
UG	Unrestricted Grammar
UL	Unrestricted Language

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Contributions to the Literature

1. N. Visnevski, S. Haykin, F. Dilkes, V. Krishnamurthy, and P. Lavoie. Syntactic modeling of multi-function radars. *IEEE Signal Processing Mag.*, to appear, Jan. 2006.

2. N. Visnevski, F. Dilkes, P. Lavoie, V. Krishnamurthy, and S. Haykin. Hidden Markov models for radar pulse train analysis in electronic support systems. *IEEE Trans. Aerosp. Electron. Syst.*, submitted, Dec. 2004.

3. N. Visnevski, F. Dilkes, V. Krishnamurthy, S. Haykin, and P. Lavoie. A special case of the radar pulse train analysis algorithm for static radar signal templates. *IEEE Trans. Aerosp. Electron. Syst.*, in preparation.

4. N. Visnevski, F. Dilkes, S. Haykin, and V. Krishnamurthy. Non-self-embedding context-free grammars for multi-function radar modeling - electronic warfare application. In *Proc. IEEE International Radar Conference (IRC05)*, pages 669–674, Arlington, VA, USA, May 2005.

5. N. Visnevski, S. Haykin, V. Krishnamurthy, F. Dilkes, and P. Lavoie. Hidden Markov models for radar pulse train analysis in electronic warfare. In *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP05)*, pages V597–V600, Philadelphia, PA, USA, Mar. 2005.

6. 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 Conference on Decision and Control (CDC03)*, pages 6295-6300, Maui, Hawaii, USA, Dec. 2003.

7. S. Haykin, N. Visnevski, F. Dilkes, V. Krishnamurthy, and P. Lavoie. Stochastic context-free grammars for multi-function radars: Modeling, joint classification, and state-estimation. *Proc. IEEE*, in preparation.

8. S. Haykin, N. Visnevski, F. Dilkes, V. Krishnamurthy, and P. Lavoie. Stochastic context-free grammars for multi-function radars: Modeling, joint classification, and state-estimation. In *Proc. Workshop on Defense*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Applications in Signal Processing (DASP'2005), Homestead, Utah, USA, Mar. 2005.

9. F. Dilkes and N. Visnevski. Non-self-embedding context-free grammars for electronic warfare. Technical Report DREO TM 2004-157, Defence Research & Development Canada, Ottawa, Canada, Oct. 2004.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction

One of the main functions of Electronic Warfare (EW) addressed in this thesis is that of passive sensing of signals from radars with the objective of estimating the potential threats posed by hostile forces. Naturally, the EW equipment is often located on board of an aircraft or a ship that is a potential target of the enemy radar. This thesis is focused on the problems that EW equipment designers face when analyzing hostile radar emissions.

As radar signals become steadily more complex, so do the problems of EW. Functions such as counter-targeting, jamming, and threat assessment rely heavily on Electronic Intelligence (ELINT) resources to provide *a priori* signal information for various known radar systems.

The intelligence information about radar emitters is typically represented by volumes of parameterized data records that describe radar transmit frequency as well as stable or semi-stable modes of radar operation. They are annotated by lines of text explaining when, why, and how a signal change from one mode to another may occur. We refer to this as a *data-centric* ELINT representation.

Traditional approaches to ELINT are inadequate considering the complexity of modern emitters. Specifically, intelligence gathering is often done by different recording hardware, and various ELINT analysts can annotate recorded data in different ways, creating a serious problem of non-homogeneity of the intelligence data.

The non-homogeneous nature of the data-centric ELINT results in loss of information during the design of the signal processing units of tactical systems. The sub-optimal performance of these systems is a manifestation of such design limitations. This is particularly problematic in the modern EW environment where some of the most dangerous threat systems are very complex and highly agile.

This thesis presents a novel approach to time-domain radar modeling and signal processing. To resolve the issue of non-homogeneous intelligence data, and to address the growing complexity of the radar systems, we propose a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

model-centric approach to ELINT.

We show how the application of a simple "divide and conquer" strategy provides an effective means of radar signal complexity management through separation of time-domain radar signals into hierarchical layers. We demonstrate how mathematical models of each hierarchical layer can be derived using data-centric ELINT records. These mathematical models form the homogeneous basis for the model-centric ELINT Electronic Threat Libraries and open new avenues for radar signal processing that were not available from the data-centric approaches.

Specifically, such important applications as radar emitter recognition, radar state estimation, prediction of radar behavior, and radar pulse train analysis can potentially benefit from the flexible modeling approach that we present in this thesis.

To the best of our knowledge, this is the first unclassified thesis written on the subject of model-centric ELINT. The novelty of this work is viewed by the author as its most significant contribution.

1.1 Background

Electronic Warfare (EW) can be broadly defined as any military action with the objective to control the electromagnetic spectrum (Adamy 2001; Adamy 2002; Schleher 1999; Schlesinger 1986; Vakin, Shustov, and Dunwell 2001). As Schleher (1999) states: *"Every element of EW is based upon accurate, timely, and focused intelligence"*. One of the most important sources of intelligence for EW is Electronic Intelligence (ELINT) – intelligence focused on radar emitter signals and other forms of electromagnetic emission (not including communications).

Contemporary ELINT is a very sophisticated concept that involves interception (Wiley 1985) and analysis (Wiley 1993) of radar signals, integration of this information with other non-electronic sources of intelligence (technical reports, publications, manufacturer and design specifications, etc.), as well as the intelligence-derived products such as data bases. These data bases represent the *a priori* information about the threats, and play a critical role in all elements of EW.

The complexity and agility of modern radars dictates the need for massive intercept data recordings by various ELINT equipment in order to obtain a complete picture about the radar functionality. Since the amount of information stored in ELINT data bases is prohibitively large for real-time data manipulations, radar signal processing units of tactical systems rely on compact Electronic Threat Libraries that are derived from ELINT data bases (Schleher 1999). These Electronic Threat Libraries can be loaded by the tactical equipment prior to military operations, and contain the essential information about

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

potential electromagnetic threats (Wiley 1993). Due to the data-centric nature of ELINT, the Electronic Threat Libraries used in the modern EW applications are also *data-centric*, in the sense that they contain threat parameters and their acceptable limits. These Electronic Threat Libraries are the main focus of this thesis.

1.2 Multi-Function Radars

The concept of a *Multi-Function Radar (MFR)* dates back to 1955 when an electronically-controlled Phased Array Antenna was first developed (Skolnik 1990; Skolnik 2002). Then, for the first time, the need of mechanical drives to control the radar beam orientation disappeared.

Phased Array Antennas consist of a large number of small antenna elements geometrically placed at certain distances from each other (usually in the form of a grid) to enable beam forming using the effect of superposition of phases of electromagnetic waves. Each antenna element is connected with an electronically controlled phase shifter that introduces a time delay into propagating waves. Since the phase shifter of each antenna element can be steered individually, the shape and direction of the antenna beam can be controlled rapidly and with a much greater efficiency than by any mechanical means (Skolnik 1990; Skolnik 2002).

The rapid and accurate beam steering capability, combined with high performance digital control, allows multiple radar functions to be performed virtually simultaneously (Skolnik 1990) (hence the term Multi-Function Radar). The principle of Time Division Multiplexing often employed by MFRs enables them to track multiple targets, perform complete hemispherical search, guide weapons, and even engage in communication activities at the same time (Skolnik 1990). MFRs often utilize sophisticated waveforms and Pulse to Pulse Interval (PPI) scheduling to optimize their performance. This level of sophistication places very serious demands on the field of EW.

Until recently, manufacturing and maintenance costs have been an important factor limiting widespread utilization of MFRs. However, recent developments on the microwave, computing, software, as well as signal processing fronts provide a new level of commercial viability for MFRs that demands serious attention from the EW community.

Advances in the solid-state electronics and Multiple Input Multiple Output antenna technology lower the cost of the microwave front. Single chip micro controllers replace digital phase shift computers that used to occupy several cubic meters of space. Adaptive beam forming techniques such as the one proposed by Griffiths and Jim (1982) enable unprecedented precision and efficiency. Finally, the advances in the area of Software-Defined Radio (Tuttlebee 2002a; Tuttlebee 2002b) clearly demonstrate the advantage of fully

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

software-controlled military equipment. One can easily see that in the software-controlled network-centric warfare of tomorrow, MFRs will become more and more software-defined, and therefore more difficult to deal with.

Target-radar interaction and the need for radar modeling

An important aspect of EW is the radar-target interaction. This interaction can be examined from two entirely different viewpoints - the viewpoint of the radar, and the viewpoint of the target.

From the radar designer's viewpoint, the primary goal of the radar is to detect the target and to identify its critical parameters. Therefore, to the radar, the target represents an uncooperative, uncontrollable system that needs to be modeled in order to apply the inference techniques.

On the contrary, EW is on the side of the potential radar target. Therefore, the radar itself becomes a problem. The target does not directly influence the radar, but it has the need to control its behavior based on the understanding of the processes that happen within the radar. For example, a military aircraft can engage into maneuvers if it is likely that the radar is guiding a weapon to intercept that aircraft. Therefore, the radar now becomes a subject of modeling.

1.3 The need for model-centric principle

In this section, we propose, motivate, and explain the model-centric approach to EW, as well as analyze its advantages. We do so by addressing the following fundamental questions:

- What are current deficiencies of EW?
- What does *"model-centricity"* mean?
- What are the main elements of the model-centric EW?

1.3.1 The nature of the modern threats and why ELINT cannot cope

EW is rooted in a period when radars were relatively primitive systems that included a high power, high frequency emitter that radiated a series of electromagnetic pulses, a sensitive receiver that detected the reflected electromagnetic energy, and a couple of mechanical drives that could position the antenna in the vertical and horizontal planes.

The goal of the radar systems was to detect and track targets. Specific tasks were performed by radars using dedicated *modes* of operation, and the electromagnetic signatures of the modes were easily distinguishable.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The data-centric or parameterized model of ELINT and Electronic Threat Libraries was quite adequate for these threats. The EW system could identify the radar and determine its operation mode by simply matching the electromagnetic signature of the radar against the available entries in Electronic Threat Libraries.

For certain tasks such as emitter recognition, data-centric radar signal processing principle still is and perhaps will always remain the tool of choice due to its simplicity and efficiency. For example, radar transmit frequency is an important characterization of the radar emitter that indicates the class of the radar system. However, other aspects of modern ELINT and radar signal processing require new approaches that go beyond the traditional parameterized approaches. This thesis focuses on the modern approaches to the EW problem.

As Wiley (1993) notes, the primary objective of radar system designers has always been the detection of targets. However, in response to advances in the EW field, new objectives of radar design now include both the satisfactory radar operation in the presence of jamming and electronic countermeasures, and minimization of the likelihood of the intercept.

Rapid advances both on the microwave front and on the computing and software fronts, have given rise to Multi-Function Radars (MFRs) and Low-Probability-of-Intercept radars, which enable radar designers to achieve all three design objectives at once, to a much greater extent than was possible in the past. The radars can locate and track multiple targets as well as guide multiple weapons simultaneously. Mechanical drives have been almost completely replaced by electronic software-controlled beam steering using Phased Array Antenna technology. The radar designers employ sophisticated techniques to lower the probability of intercept of radar signals. Complex waveforms and algorithmically-controlled pulse sequences (scheduled Pulse to Pulse Intervals) add an additional degree of intricacy to the task of radar signal processing.

In the present day, in the EW community, there is a growing concern that ELINT and some areas of radar signal processing do not keep up with the developments on the radar front. Lavoie (2001) has stated that the basic assumptions of conventional EW are no longer valid. Radar signals are no longer stable and radar modes *"date back to the early days of radar when an operator would change the signal by manually switching to another electrical circuit"* (Lavoie 2001).

In addition to the point identified in (Lavoie 2001), we have to remember the data-centric nature of ELINT and its derivative threat libraries. Given the complexity and agility of modern radar emitters, the data collection process of ELINT has to be more comprehensive. This introduces data base scalability and management problems. As Wiley (1993) sates: *"There are so many parameters that may be used to describe a signal, and so many different ELINT*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

signals to describe, that it becomes a problem to maintain up-to-date and accessible files suitable for the several users of such data".

Data-centric approaches are highly demanding in terms of storage capacity, yet provide no insight into the data and give no explanation of the underlying logical and physical processes. Data-centric approaches are also highly demanding in terms of analysis since large volumes of data have to be processed. In addition, no matter how much data is collected, there is never a guarantee that this data completely represents a complex system. Therefore, the data-centric approaches to ELINT simply cannot cope with the new reality. Alternative approaches are required.

To realize the urgency of the situation, one can look at trends in modern warfare development. The concept of the network-centric warfare implies heavy dependency on software-defined warfare infrastructure. With this come additional degrees of scalability, flexibility, and agility of electronic threats. In order to respond to them properly, one has to understand the essence of the underlying processes. Model-centric approach with all its advantages described in Section 1.3.2 appeals as the very effective tool for the job.

1.3.2 The model-centric principle

The model-centric principle, in contrast to the data-centric one, is centered around the idea of modeling. A model is generally defined as *"a simplified or idealized description or conception of a particular system, ... that is put forward as a basis for theoretical or empirical understanding, or for calculations, predictions, etc."* (Oxford English Dictionary). The primary purpose of models is to improve our understanding of a given system with the aim of making sound and wise decisions about the underlying processes that models represent.

West and Harrison (1997) identify the three most important properties that a good model should offer:

Description is the property that supports the meaning of the model. It includes simplicity, completeness and the relative significance of the elements that comprise the model as well as their interactions and impacts on each other.

Control implies our ability to influence the behavior of the underlying system. Alternatively, if the system cannot be influenced, the model of such a system should at least provide mechanisms for controlling the decisions that we make in regards to the system's behavior.

Robustness is the property that supports the flexibility and efficiency in adjustment of the model when faced with some unforseen or unexpected circumstances.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Description is perhaps the only property that the data-centric principle supports to some extent. On the other hand, only the model-centric principle accommodates for control and robustness.

The three properties described above are supported through the model structure $\mathcal{M}$ (West and Harrison 1997):

$$\mathcal{M} = \{\mathcal{C}, \mathcal{F}, \mathcal{Q}\} \tag{1.1}$$

where:

$\mathcal{C}$ is the conceptual basis of the model,

$\mathcal{F}$ is the model form, and

$\mathcal{Q}$ is the quantitative basis of the model.

The model structure (1.1) is inherently hierarchical. The conceptual form $\mathcal{C}$ is an abstract representation of a model made of basic scientific principles. Therefore, it is very stable and should rarely change. The model form $\mathcal{F}$ represents the model qualitatively, describing basic model terms and relationships. Different models having the same conceptual basis may differ at the qualitative level. Finally, the quantitative basis of the model $\mathcal{Q}$ defines a particular instance of that model's representation. It typically contains model parameters, that are tunable to support flexibility and robustness.

The structure (1.1) and the properties that it supports, give model-centric approaches the following advantages over the data-centric ones:

Compactness. Models are compact representations of the data sequences that they can produce. One can view modeling as a form of data compression. This alleviates the demands for storage of the data-centric approaches.

Accuracy. A well-validated model developed with insight into the physical principles of the system and the environment provides more accurate coverage of the system's functionality than disjoint sets of recorded observations.

Simulation. The critical role of simulation in EW has been demonstrated by Adamy (2002). Modeling is an integral part of simulation. In fact, simulation becomes a straightforward mechanical task once a practical model of the system is available.

Analysis. One may argue that modeling originated in response to analysis requirements. The types of models often dictate the analysis approaches and complexity. The reverse is also true. Bayesian models, for instance, stand out among all other types of models due to their important relationship with statistical inference (Bernardo and Smith 2000; Pearl 1988; Robert 2001; West and Harrison 1997).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The four items listed above demonstrate significant advantages of model-centricity when compared to data-centric principles. However, model-centric approaches have their own limitations. An important one is the model complexity. It is natural for the model designer to try to make the model as close to reality as possible. The question is, how one can keep the model complexity under control, and what design tradeoffs are required to do so? In this thesis, we attempt to exploit the advantages of the model-centric approaches to EW and to address the issues of model complexity management.

1.3.3 The essence of the model-centric Electronic Intelligence

The central elements of the model-centric ELINT and Electronic Threat Libraries are models of electronic threats such as radars. There are two main aspects of this approach. The first is the modeling itself, which we address in more detail in Section 1.4. The second is the issue of model complexity management. In this section, we look at how the well-known engineering principle of "divide and conquer" can be utilized to keep the complexity of electronic threat models under control.

We observe that the modern electronically-agile software-controlled radars use sophisticated, hierarchical signal structures in order to optimize their performance. For instance, MFRs employ Time Division Multiplexing to engage multiple targets at once. Radar pulses are arranged in groups, which are dynamically scheduled by radar control software. A single Pulse to Pulse Interval (PPI) in radar signals currently contains very little information about what task the radar is performing. Rather, static or dynamically varying groups of pulses may provide clues about the state of the radar. Models that address all the aspects of radar functionality starting from a single PPI and including radar state transitions, tracking, search and guidance algorithms, etc. are not practical and could prove prohibitively complex.

To address the issue of electronic threat model complexity, we propose a layered, hierarchical model of radar signal organization. The layers can be identified according to certain characteristics, and the single radar model can be broken into a series of smaller models that represent radar signals at particular layers. The concept of radar signal data layers and their implication on model complexity management could be viewed as another important contribution of this thesis.

1.4 Syntactic Multi-Function Radar modeling

The goal of this section is to propose, motivate, and explain the syntactic modeling approach to Multi-Function Radars (MFRs), as well as to analyze

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

its advantages. We do so by addressing the following questions:

- What does the concept of *"syntactic modeling"* mean?
- Why is syntactic MFR modeling a promising approach?

1.4.1 What does "syntactic" mean?

Discrete event systems and formal languages

The theory of formal languages has originated in response to the needs of the natural language processing researchers (Chomsky 1956; Chomsky 1959b; Chomsky and Miller 1958), but has quickly found applications in various other areas of engineering and computer science. It has been shown to be one of the most useful modeling frameworks for Discrete Event Systems (Cassandras and Lafortune 1999).

The very basic building block of any language is an *alphabet*. The Webster's New Universal Unabridged Dictionary defines alphabet (from Greek *alphábētos*) as *"any system of characters or signs with which a language is written"*. The *language* itself is defined as *"any systems of formalized symbols and rules for their combination and use, by means of which information can be transferred"*.

Sequences of symbols build up *words*. Words, combined together according to some rules form *phrases*. Phrases, forming an independent and logically complete block of information, make up a *sentence*.

The relationship between formal languages and Discrete Event Systems is quite strong. An underlying set of events is associated with any Discrete Event System. This set of events can be considered as an alphabet of the language of this system (Cassandras and Lafortune 1999). The sequences of events build up words and phrases of that language. Therefore, one might ask the following fundamental question – "given a certain language, how can we build a system that *speaks* this language?"

Grammars and syntax analysis

The notions of words and sentences require an expansion of the set of building blocks for a language mentioned above. The *vocabulary* (or *lexicon* from Greek *lexicón* - use of words) of the language is a set of all words of the language (possibly infinite). The language itself is often an infinite set of phrases and sentences formed out of words from its vocabulary.

The set of rules of the language is referred to as *grammar* (or *syntax* from Greek *sýntaxis* - an arranging in order). Most importantly, for practical formal languages, grammars are finite sets of rules. In other words, grammars are efficient finite representations of infinite sets of strings (languages).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In the information-theoretic sense, lexical and syntactic constraints of the language reduce the measure of uncertainty (decrease entropy) of the content of the text in that language to facilitate its recognition and processing. Deller, Hansen, and Proakis (1999) define the concept of *language processing* as follows: *"Language processing is generally concerned with the attempt to recognize a large pattern (sentence) by decomposing it into small sub-patterns according to rules that reduce entropy"*.

In light of the definitions above, formal language processing involves at least 2 major phases:

Lexical analysis – validation and extraction of the lexical content

Syntax analysis – validation and extraction of the grammatical structure

1.4.2 The essence of syntactic radar modeling

In Section 1.2, we have discussed the complicated structure of Multi-Function Radars (MFRs), and the need of MFR modeling from the target viewpoint. Our findings reported in (Visnevski, Krishnamurthy, Haykin, Currie, Dilkes, and Lavoie 2003) demonstrate that due to their complex, software-controlled nature, MFRs fit quite well into the stochastic Discrete Event Systems framework. Therefore, the radar viewed from the target side is a perfect example of a stochastic Discrete Event System.

In Section 1.4.1 we have defined the concept of formal languages in the context of Discrete Event System modeling. It seems natural to suggest that formal languages are a perfect match for MFR models. Therefore, syntactic radar modeling involves uncovering the grammatical structure of the MFR languages, and employing the techniques of syntactic pattern recognition (Fu 1974; Fu 1982), including lexical and syntax analysis, in radar signal processing.

1.5 Overview of the thesis

In this section, we state the main contributions of the thesis, define the scope, specify the important assumptions, and provide an overview of the thesis organization.

1.5.1 Main contributions

This thesis presents the following contributions to the field of radar modeling and radar signal processing:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Model-centric principle. To resolve the issue of non-homogeneous intelligence data, and to address the growing complexity of the radar systems, we propose a *model-centric* approach to ELINT. This is a novel approach that promises significant advantages over the traditional data-centric or parametric ELINT organization.

We demonstrate how mathematical models for radar emitters can be derived using data-centric ELINT records. These mathematical models form a homogeneous basis for the model-centric ELINT and open new avenues for radar signal processing that traditional, data-centric approaches did not support.

Syntactic radar modeling. The novelty of this work rests in the view of complex radar emitters as abstract Discrete Event Systems that broadcast messages using some stochastic formal language. The most widely known and powerful model for formal languages is a grammar. Therefore, we can model radar languages with stochastic grammars, and process radar messages using methods from the theory of syntax analysis, which we also view as a strong and novel contribution.

Syntactic modeling has been used in the past primarily in pattern recognition applications. Most notably, in computer language parsing (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Aho and Ullman 1973; Hopcroft, Motwani, and Ullman 2001), natural language processing (Deller, Hansen, and Proakis 1999), and in the area of bioinformatics and genomic sequencing (Baldi and Brunak 2001; Durbin, Eddy, Krogh, and Mitchison 1998). Most of these tasks are performed in the off-line fashion and do not address real-time processing requirements of EW. To the best of our knowledge, this is the first time syntactic modeling in the form presented in this thesis is applied to the on-line signal processing problems outside of the problem domains mentioned above.

Our approach of radar model complexity management based on the *"divide and conquer"* principle applied to the hierarchical radar signal architecture is an additional contribution.

After conducting a comprehensive search of the open literature on the subject of modeling in EW, we realized the apparent lack of research efforts that address shortcomings of the data-centric nature of ELINT. The data-centric approach remains the dominant approach to many real-life EW problems. The difficulties that ELINT encounters when dealing with sophisticated electronic threats are recognized by a number of authors (Lavoie 2001; Wiley 1993). The EW community has developed an appreciation for the potentials that model-centric approaches provide (Lavoie 2001). However, to the best of our knowledge, this is the first unclassified thesis written on the subject of model-centric ELINT.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1.5.2 Scope and assumptions

The challenges presented to EW by the levels of sophistication of modern electronic threats are significant. They cannot all be addressed in one thesis. Therefore it is important for us to determine the scope of the exposition and to specify important assumptions made when presenting the results of our investigation.

The level of detail in the intelligence reports and ELINT data base entries may vary quite significantly from one radar to another. Detailed design and manufacturer specifications may be available for some radars, whereas only sparse and noisy data recordings may be available for the others. The amount of *a priori* information available to EW analysts usually dictates the modeling approaches.

In the case when the knowledge base about the radar is limited, the radar models have to be inferred from sparse observation data. This approach is usually called *blind model inference*. The greatest challenge of this approach is to ensure that the maximum amount of information about radar operation is extracted from the data.

Alternatively, in case of a large *a priori* knowledge base about the radar, the models can be developed fairly accurately using some formal synthesis technique. This is commonly referred to as the approach of *structural model synthesis*. The challenge of this approach is to maximize utilization of available information about the radar during the process of model synthesis.

We have chosen the approach of structural model synthesis for two reasons. As stated in Section 1.3, ELINT data bases are already overloaded with data about radars. The *a priori* information about radars is there, but extraction, utilization, and management of this information has become a problem. Therefore, the structural model synthesis approach is the approach that can currently make the biggest impact on the field of EW. Thus, without any intentions to diminish the significance of the blind model inference approach, we are leaving it out of scope of this thesis. We hope that this will keep the exposition more focused and concrete.

Another important aspect of EW is radar identification. Before the radar model can become useful to radar signal processing equipment, the radar signals have to be correctly classified, and the proper model identified. Unfortunately, the systematic approach to this problem is currently not available (Lavoie 2001), but some heuristic algorithms have been shown to work in a satisfactory fashion (Matuszewski and Paradowski 1998; Ray 1998).

It is often advantageous to approach radar identification from the data-centric (parameterized) viewpoint. For instance, such parameters as transmit frequency, signal power, dominant Pulse to Pulse Intervals, and beam width can often provide emitter classification hints. The data-centric nature of the emitter classification task suggests that it does not really fit into the scope of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

this investigation. Therefore, throughout this thesis we assume that the radar classification task has been solved successfully.

EW systems often operate in the conditions of non-stationarity of the environment in which radar signal observations are recorded. This demands tracking the variations of statistical radar model parameters. We recognize both the importance and the complexity of this task. In fact, this seems like an ideal application for the particle filtering technique (Arulampalam, Maskell, Gordon, and Clapp 2002; Crisan and Doucet 2002; Djuric, Kotecha, Zhang, Huang, Ghirmai, Bugallo, and Miguez 2003) since this technique specifically focuses on non-stationary non-Gaussian tracking problems. However, in order to keep the research efforts manageable, we intentionally leave it as one of the potential directions for future investigations.

One of the important aspects of modeling is model validation. Given a model and the *a priori* information about the system's operation, one must be able to find out how closely the system's model reflects the real system's behavior. This would require the model designer to compare simulated outputs of the model with the real recorded data of the radar. Unfortunately, given the classified nature of the military radar systems, these recordings were not available for analysis, or the results of the validation could not be published in the open literature. Therefore, we will also leave this matter for future investigations.

Finally, this thesis is focused on modeling of Multi-Function Radars (MFRs). For simplicity, we will refer to MFRs as just *"radars"*. However, it is important to understand that most of the principles in the thesis really apply to MFRs since conventional radar systems are already adequately represented in the data-centric Electronic Threat Libraries. It is the MFR that is the driving force of this research effort, so when we say *"radar"* we really mean *"Multi-Function Radar"*.

1.5.3 Organization

The rest of the material presented in this thesis is separated into three independent parts. In Part I, we present the necessary elements and building blocks of the proposed syntactic modeling methodology. Part II develops the modeling methodology itself. In Part III, we apply the developed modeling methodology to specific real-life radar modeling and signal-processing problems.

Part I includes Chapters 2-4. In Chapter 2, we address the issues of complexity of radar models and propose a hierarchical modeling architecture that allows to keep radar model complexity under control. Chapter 3 introduces fundamental elements of the theory of syntactic modeling and syntax analysis. Chapter 4 presents the background on Markov chains and Hidden Markov Models and establishes the link between syntactic modeling and Hidden Markov Models. This lays the foundation for the modeling methodology

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

to follow.

Part II develops the syntactic radar modeling methodology based on the material presented in Part I. It includes Chapters 5-7. Chapters 5 and 6 develop modeling techniques for two critical layers of the radar modeling hierarchy proposed in Part I. Although both layers of modeling are very important, the emphasis of this thesis is on the so called *word-level* modeling described in Chapter 5. *Pulse-level* models are playing a supporting role to the word-level modeling techniques. Therefore, we discuss them in Chapter 6 after word-level modeling is presented. Chapter 7 presents a global overview of the model-centricity in Electronic Intelligence and demonstrates how the proposed modeling techniques fit together and form a homogeneous basis for advanced Electronic Threat Libraries.

Part III includes Chapters 8 and 9. In Chapter 8, we demonstrate how the proposed modeling technique can be applied to modeling of real-life Multi-Function Radars. Chapter 9 looks at an important application of syntactic modeling - the task of pulse train analysts.

Chapter 10 summarizes the main contributions of this thesis and offers some concluding remarks. Detailed specification of two real-life Multi-Function Radars is provided in Appendices A and B. Finally, a promising variation of the pulse train analysis algorithm presented in Chapter 9 is briefly discussed in Appendix C.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Part I

Essential elements of the model-centric approach

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

Hierarchical model of Multi-Function Radar signals

In Chapter 1, we have identified the level of model complexity associated with modern Multi-Function Radars (MFRs) as the serious bottleneck of the model-centric approach to Electronic Intelligence (ELINT). The level of sophistication of modern radars is such that radar signals can no longer be viewed as stable sequences of pulses. Pulse Repetition Intervals (PRIs) are typically scheduled by intricate radar control software. In addition, multiple independent operations performed by MFRs are often multiplexed in time. This makes the task of modeling modern radars overwhelmingly difficult. In this chapter, we propose a novel hierarchical modeling approach that helps maintaining radar model complexity at the manageable level. To the best of our knowledge, this approach to radar modeling has not been considered in the open literature in the past. In addition to constraining model complexity, this approach also offers the benefits of model modularity and compactness.

This chapter is organized as follows. In Section 2.1, we briefly introduce the principle of "divide and conquer" and discuss its applicability to the problem of radar modeling.

In Section 2.2, we demonstrate how the principle of "divide and conquer" can be applied to MFR modeling to develop a hierarchical radar modeling approach that keeps model complexity under control. We propose a three-layer hierarchical model, and identify the first two layers of this model as particularly important for the Electronic Warfare (EW) applications. Later in this thesis, we develop syntactic modeling techniques that represent these levels of the hierarchy.

Finally, Section 2.3 summarizes the major results of this chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.1 "Divide and conquer" principle

By definition, "divide and conquer" is a problem-solving methodology that involves partitioning a difficult problem into subproblems that are relatively simple to tackle, solving the subproblems, and then combining the solutions to the subproblems into a solution for the original problem.

We observe that in terms of model complexity "divide and conquer" technique translates to reducing the scope of the model components, while simultaneously increasing their level of detail with the aim of keeping the model confidence at its maximum.

In terms of MFR modeling from the EW standpoint, the principle of "divide and conquer" is best applied to the structure of the observed radar signals. One of the characteristic features of MFRs is their complex signal organization. We would like to exploit this signal organization by creating hierarchical models of MFR signals. The scope of each hierarchical modeling layer will be limited, while the level of detail involved at each layer could be maintained high without sacrificing the overall model confidence.

We should also keep in mind that advantages of the "divide and conquer" approach come with a cost. Often, hard-decision thresholds have to be employed at the boundaries of hierarchical model components. This could potentially result in some loss or distortion of information propagated through the model. However, the maintainability and modularization benefits of hierarchical models are usually much more desirable.

2.2 Layered radar signal architecture

As was previously observed, Multi-Function Radars employ sophisticated signal structures and computer-controlled Pulse to Pulse Interval (PPI) scheduling. Fig. 2.1 illustrates an example signal structure of one of the MFRs (a detailed description of this emitter is given in Appendix A). This emitter has a three-layer hierarchical signal structure that is common to many MFRs.

This radar employs special entities called *words* that are represented by a certain sequence of pulses. Every word consists of five segments (A – E) and words differ from each other only through PPIs of pulses in section B.

Words, combined in *phrases*, are dependent on the current radar function or state of operation. This particular MFR is capable of engaging five different targets using Time Division Multiplexing (TDM). This is achieved through *clauses* which consist of five phrases associated with five independent tasks of the MFR. The illustration of the radar signal evolution in time is shown in Fig. 2.2.

In Fig. 2.3, we present a hierarchical signal structure of another MFR. This emitter is described in detail in Appendix B. Although structurally different

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

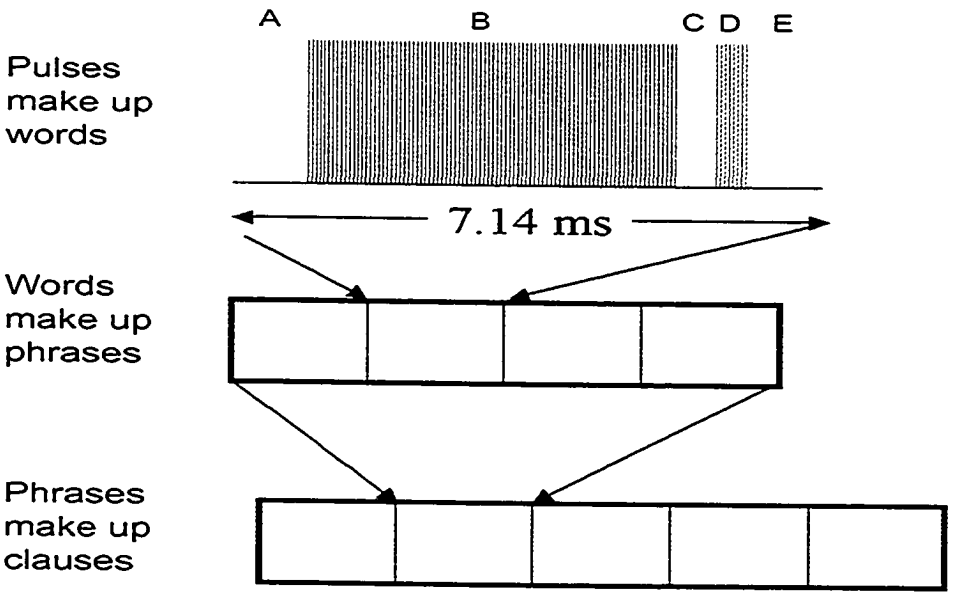

Figure 2.1: A common three-layer hierarchical radar signal structure. Pulse sequences are arranged into groups according to specific patterns. These groups are called *words*. For this particular emitter, words are all of equal length of $7.14ms$, but in general they can be of variable length. Words are made up of five sections (A - E). Sections A, C, and E are dead times of known duration. Section B is a fixed PPI pulse-Doppler sequence, and section D is a scheduled PPI synchronization burst. *Phrases* are made up of several words grouped together (four in the case of this emitter). One phrase is commonly associated with a single task like search or track. *Clauses* are made up of phrases, and the number of phrases in one clause determines the number of tasks the radar can perform simultaneously. A clause is a product of Time Division Multiplexing in MFR operation. This particular radar has five phrases within one clause which means that it can perform five different operations simultaneously.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

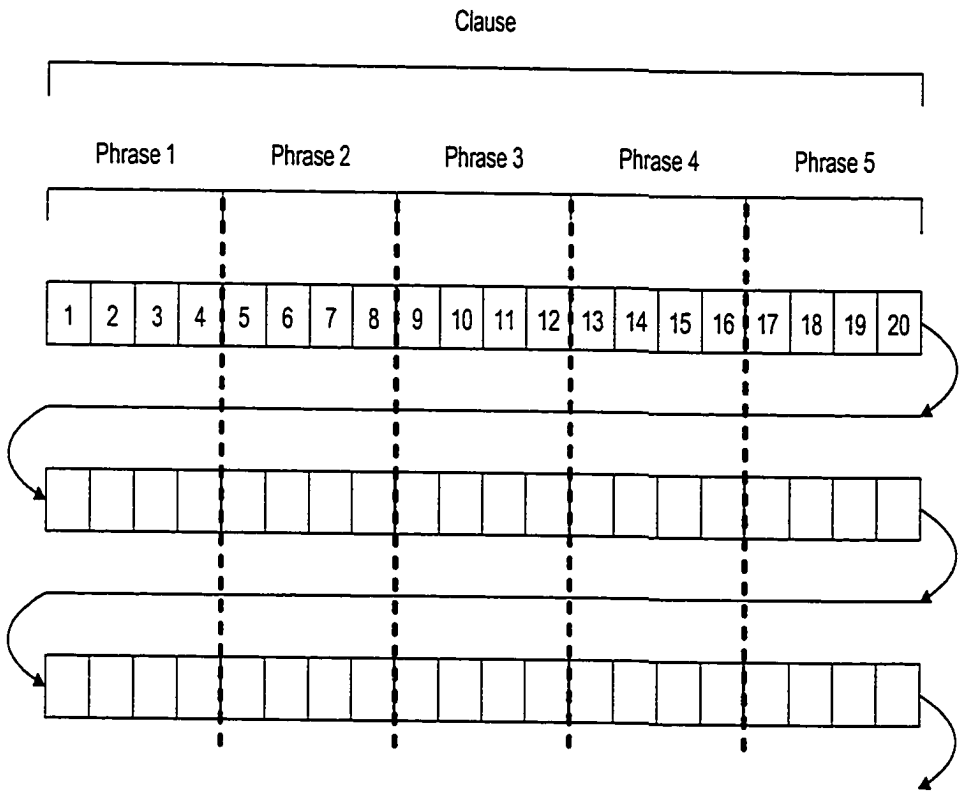

Figure 2.2: Radar output sequence for the emitter with the signal structure of Fig. 2.1.

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.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

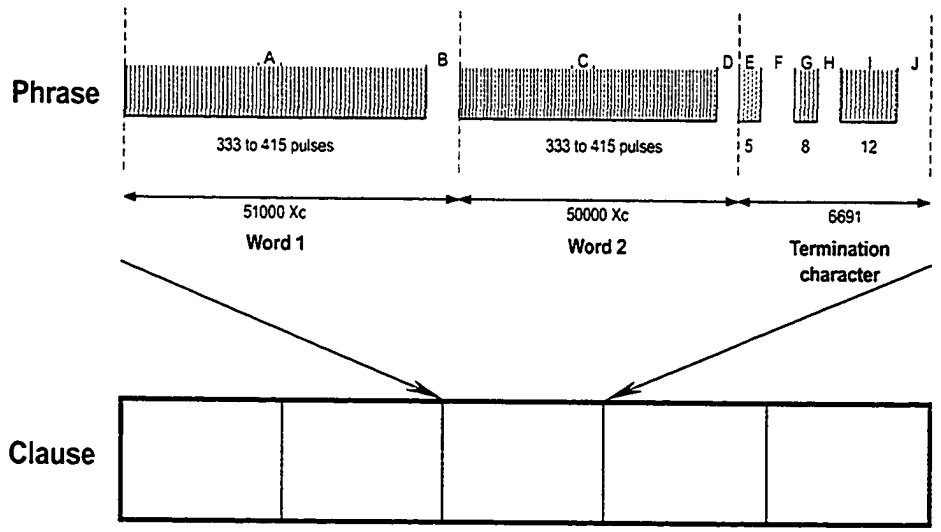

Figure 2.3: Layered signal structure of another radar.
This emitter employs words, phrases, and clauses in its signal organization. A phrase is constructed using 2 words followed by a single termination character. Words within the phrase can be different, but the termination character is always the same. The words of this emitter are of different length, but the duration of the phrase is fixed (107,691 crystal clock counts (Xc), which is $\approx 10.23ms$). To accommodate for the differences in word length, dead time pads (regions B and D) of variable duration are employed. A termination character is a sequence of scheduled PPIs and consists of five distinct regions (E - J). Regions E, G, and I contain 5, 8, and 12 fixed PPI pulses, respectively. Regions F, H, and J are dead times of known length. This particular radar has five phrases within one clause which means that it can perform five different operations simultaneously. The output sequence of this radar is formed in a similar fashion to the sequence of the radar emitter considered earlier.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

from Fig. 2.1, the signals of both emitters share many common features.

It is obviously very hard to develop a singe model for either of these Multi-Function Radars given the described structure of their signals. However, the task becomes more manageable if we develop a layered, hierarchical model following the radar's signal structure illustrated in Fig. 2.1 and 2.3. In other words, we will consider MFRs at three levels - the *pulse-level*, the *word-level*, and the *phrase-level*. We then will combine the three levels of modeling to form a single hierarchical model of the radar.

The scope of the pulse-level model will be limited to the structural aspects of radar words. Such factors and PPI schedule, word segments and their order will be taken into consideration. The pulse-level models will be establishing a mapping between the physical radar pulses and the radar word id.

The scope of the word-level model will be limited to capturing the target-specific behavior of the radar. Pulses will not feature in this model. It will be developed on the conceptual level and will incorporate the dynamic relationships between the MFR state and the sequence of words generated in this state. From the EW stand point, this is the most important level of modeling since it deals with the internal dynamics of the radar.

The scope of the phrase-level model will be limited to the coordination of the Time Division Multiplexing and multitasking aspects of the MFR. This level of modeling is important for MFR simulation, but is of limited use to EW. Therefore, it will not be addressed in detail in this thesis.

2.3 Summary

In this chapter, we examined the issues related to complexity of the model-centric approach proposed in Chapter 1. We have closely examined the structure of signals employed by Multi-Function Radars and defined a modeling hierarchy that consists of three distinct levels - pulse-level, word-level, and phrase-level. Each modeling level of this hierarchy is self-contained, and the amount of information propagated from one level to the next is restricted to the necessary minimum.

This layered hierarchy allows to reduce the modeling scope at every layer, while simultaneously increasing the level of detail at every modeling level without sacrificing model confidence and validity. This approach enables us to keep the overall level of complexity of Multi-Function Radar models manageable.

In Part II of this thesis, we will develop syntactic modeling approaches to two out of three hierarchical levels identified in this chapter - pulse-level and word-level. These two layers of the modeling hierarchy are most relevant to the domain of Electronic Warfare.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

Elements of syntactic modeling

This chapter presents important elements from the theory of syntactic modeling, syntactic pattern recognition, and syntax analysis. The background material of this chapter covers critical elements of the model-centric approach proposed in Chapter 1.

In conventional terms, the definitions and notations used in this chapter follow the theory of formal languages and computational linguistics (Aho and Ullman 1972; Hopcroft and Ullman 1979). 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 this thesis, we propose to view the signals of Multi-Function Radars as such collections of strings. This allows interpreting these radars as systems that communicate information in some formal language.

Set-theoretic definitions of formal languages, although mathematically most accurate, do not provide any practical means of analyzing the strings from these languages. 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**.)

Parsing is also often called the problem of *syntax analysis*.

The finite-dimensional models of languages that help answer 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*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Grammars are by no means unique models of a given language. In fact, infinitely many grammatical models can be developed for a single language. Therefore, it is not surprising that generative and transformational grammars of one language can be very different models, tuned to specific needs of a particular application.

The goal of the generative grammar is to derive valid strings that are guaranteed to be in a given language. Consequently, the structure of the generative grammar can be very complicated with a large number of checks and balances in the grammatical structure that ensures validity of generated strings. The drawback of this rigorous grammatical structure is that it is often impractical for the parsing applications since the structural complexity of generative grammars often translates into computational complexity of parsing algorithms (Aho and Ullman 1972; Aho and Ullman 1973).

On the other hand, the goal of transformational grammars is to lower computational costs of algorithms of determining the membership of any given string in a language. This can often be achieved by relaxing some generative constraints of the grammar. This may lead to a structurally simpler grammar that represents a superset of the original language. Consequently, transformational grammar may generate strings that are not in the original language.

Since the goal of this thesis stated in Chapter 1 is to develop a modeling methodology for Multi-Function Radars that supports radar signal processing, we will focus our attention on transformational grammars only.

The rest of this chapter is organized as follows. In Section 3.1, we define the concept of formal languages. Section 3.2 presents mathematical structure of transformational grammars and discusses different classes of grammars and classes of languages that these grammars are commonly associated with.

Section 3.3 looks at one class of grammars that is called Finite-State Grammars (FSG) and establishes the relationship between these grammars and Finite-State Automata. Finite-State Grammars describe a class of Finite-State Languages that will play a critical role in syntactic radar modeling.

Section 3.4 examines another class of grammars that is called Context-Free Grammars. In general these grammars describe the class of Context-Free Languages that is much more broad than the class of Finite-State Languages. However, a special subclass of Context-Free Grammars called Non-Self-Embedding (NSE) Context-Free Grammars is known to be a class of compact models of Finite-State Languages. This compactness of Non-Self-Embedding Context-Free Grammars will also play a very important role in the following chapters when we present a complete Multi-Function Radar modeling methodology.

In Section 3.5 we will discuss stochastic equivalents of transformational grammars and establish an important relationship between Stochastic Finite-State Grammars and Hidden Markov Models. This is an important link that will be exploited later in developing stochastic models of Multi-Function

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Radars.

Finally, Section 3.6 summarizes the presented material and provides an overview of the literature that contains more in-depth treatment of the subjects addressed in this chapter.

3.1 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

$$\mathcal{A}^+ = \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots\} \quad (3.1)$$

$$\mathcal{A}^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \ldots\} \quad (3.2)$$

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 3.1.1 *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, $\varnothing$, $\mathcal{A}$, and $\mathcal{A}^*$ are also languages.

3.2 Transformational grammars

The definition of the formal language (Def. 3.1.1) is extremely broad and therefore, has very limited practical application. A more useful way of defining formal languages is through the use of *transformational grammars* (or simply *grammars* (Chomsky 1956; Chomsky 1959a; Chomsky 1959b; Chomsky and Miller 1958)).

Definition and examples

Definition 3.2.1 *A* deterministic transformational grammar G *is a four-tuple*

$$G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0) \quad (3.3)$$

where:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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

$\mathcal{E}$ *is the set of non-terminal symbols of the grammar;*

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

S_0 *is the staring non-terminal.*

In general, Γ is a partially defined function of type

$$\Gamma : (\mathcal{A} \cup \mathcal{E})^* \rightarrow (\mathcal{A} \cup \mathcal{E})^*. \tag{3.4}$$

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 thesis, unless specified otherwise, we will write non-terminal 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. 3.1.1 provides a set-theoretic definition of a formal language. Now, using Def. 3.2.1 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:

$$\begin{aligned} \mathcal{A} = &\quad \{a, b\} \quad S_0 \rightarrow aS_1|b \\ \mathcal{E} = &\quad \{S_0, S_1\} \quad S_1 \rightarrow bS_0|a \end{aligned} \tag{3.5}$$

These are some of the strings that are in this language, and examples of how they can be derived:

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.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

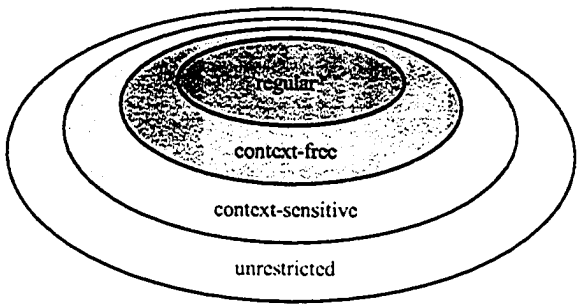

Figure 3.1: The Chomsky hierarchy of formal languages.

Chomsky hierarchy of transformational grammars

In Def. 3.2.1, the production rules of the grammar are given in a very general form. Chomsky (1959b) 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 transformational grammars:

RG *Regular Grammars.* 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 non-terminal only, and the right-hand side could be either one terminal or one terminal followed by one non-terminal. 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 (FSGs).*

CFG *Context-Free Grammars.* 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 non-terminal only, whereas the right-hand side can be any string.

CSG *Context-Sensitive Grammars.* 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 non-terminal S are dependent on its context α_1 and α_2.

UG *Unrestricted Grammars.* Any production rules of the form $\alpha_1 S \alpha_2 \rightarrow \gamma$ are allowed. Here $\alpha_1, \alpha_2, \gamma \in (\mathcal{A} \cup \mathcal{E})^*$. The unrestricted grammars are also often referred to as *type-0 grammars* due to Chomsky (Chomsky 1959b).

Figure 3.1 illustrates the Chomsky 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 Context-Sensitive

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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)

Table 3.1: Deterministic grammars, production rules, and languages.

Language (CSL), and Regular Language (RL) is a special case of Context-Free Language (CFL). Table 3.1 provides a condensed summary of the classes of transformational grammars, their production rule structures, and classes of languages that they define. More detailed treatment of the Chomsky hierarchy is given by Durbin, Eddy, Krogh, and Mitchison (1998).

Syntactic modeling of MFRs developed in this thesis will make extensive use of FSGs and CFGs. CSGs and UGs will not be used in our modeling approach.

3.3 Finite-State Languages and Automata

In this section, we will look in more detail at the Finite-State Languages (FSLs) which are often referred to as RLs. We will review such concepts as regular sets, regular expressions, regular grammars, and Finite-State Automata (FSA). We will use these concepts when developing radar pulse train models in Chapther 6 and the pulse train analysis algorithm in Chapter 9.

Finite-State Languages and regular expressions

Definition 3.3.1 *Let $\mathcal{A}$ be a finite alphabet. We define a* regular set over $\mathcal{A}$ *in the following way (Aho and Ullman 1972):*

1. *$\varnothing$ (the empty set) is a regular set over $\mathcal{A}$;*
2. *$\{\varepsilon\}$ is a regular set over $\mathcal{A}$;*
3. *$(\forall a \in \mathcal{A})\{a\}$ is a regular set over $\mathcal{A}$;*
4. *if $\mathcal{P}$ and $\mathcal{Q}$ are regular sets over $\mathcal{A}$, then so are $\mathcal{P} \cup \mathcal{Q}$, $\mathcal{PQ}$, and $\mathcal{P}^*$;*
5. *nothing else is a regular set.*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In other words, the subset of $\mathcal{A}^*$ is regular *iff* it is $\varnothing$, $\{\varepsilon\}$, or $\{a\}$ for some $a \in \mathcal{A}$, or it can be obtained from these sets by union, concatenation, or closure. The languages that are regular sets defined by Def. 3.3.1 are called *Regular Languages.*

Definition 3.3.2 Regular expressions over $\mathcal{A}$ *and the regular sets they* denote *are defined in the following way (Aho and Ullman 1972):*

1. $\varnothing$ *is a regular expression denoting the regular set* $\varnothing$*;*
2. ε *is a regular expression denoting the regular set* $\{\varepsilon\}$*;*
3. $a \in \mathcal{A}$ *is a regular expression denoting the regular set* $\{a\}$*;*
4. *if* p *and* q *are regular expressions denoting regular sets* $\mathcal{P}$ *and* $\mathcal{Q}$*, respectively, then* $(p+q)$ *is a regular expression denoting* $\mathcal{P} \cup \mathcal{Q}$*,* pq *is a regular expression denoting* $\mathcal{P}\mathcal{Q}$*, and* $(p)^*$ *is a regular expression denoting* $\mathcal{P}^*$*;*
5. *nothing else is a regular expression.*

It is worth mentioning that regular expressions denoting regular sets are not unique. In fact, each regular set has an infinite number of regular expressions that denote it, but each regular expression can only denote one regular set (see (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Hopcroft, Motwani, and Ullman 2001; Hopcroft and Ullman 1979; Salomaa 1973)). For example, the language defined in (3.5) can be denoted by the following regular expression:

$$(ab)^*(b + aa)$$

We now can make an important observation that Regular Grammars and regular expressions are two equivalent representations of the Regular Languages[1].

Relationship between Regular Languages and Finite-State Automata

Definition 3.3.3 *A* Finite State Automaton *(FSA)* Λ *is a five-tuple*

$$\Lambda = (Q, \Sigma, \delta, q_0, F)\,, \tag{3.6}$$

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;*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

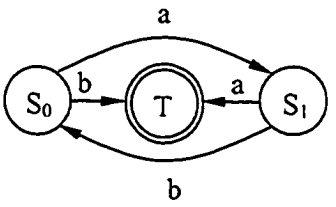

Figure 3.2: FSA equivalent to the grammar example (3.5). State S_0 is the starting state, and T is an accepting state, as indicated by the double circle.

F is the set of final (accepting) states of the FSA ($F \subset Q$).

Finite-State Automata were shown to be equivalent to Regular Grammars, Regular Languages, and regular expressions (see (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Hopcroft and Ullman 1979; Salomaa 1969; Salomaa 1973; Salomaa 1985)). In fact, using Def. 3.2.1 and Def. 3.3.3 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 (3.5) is shown in Fig. 3.2.

3.4 Context-Free Languages and Context-Free Grammars

The next, less restricted member of the Chomsky hierarchy of transformational grammars is the *Context-Free Grammar (CFG)*. Languages that can be accepted by FSA are limited in terms of strings that they can contain. The most famous example of a language that cannot be accepted by FSA is the language of *palindromes*[2]. It was shown to be a CFL (Hopcroft and Ullman 1979). A simple language of palindromes can, for example, be defined by the following set of production rules:

$$P \rightarrow bPb|aPa|b|a|\varepsilon\,, \tag{3.7}$$

and an example string from this language is *bababaaababab*. According to Table 3.1, the grammar in (3.7) is a CFG.

CFGs are often associated with tree-like graphs instead of FSA since the dependency between the elements of the strings of the CFL are nested (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Aho and Ullman 1973; Hopcroft,

[1]See (Aho and Ullman 1972; Hopcroft and Ullman 1979) for more mathematically rigorous treatment of this statement.

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Motwani, and Ullman 2001; Hopcroft and Ullman 1979). Due to this fact, the task of processing the strings from a Context-Free Language is a more computationally complex procedure than that of a Regular Language. On the other hand, Anselmo, Giammarresi, and Varricchio (2003) have shown that CFGs could be more compact descriptions of the Regular Languages than Regular Grammars or regular expressions. 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 Finite-State Automata.

As Fig. 3.1 clearly demonstrates, Regular Languages are a proper subset of the class of Context-Free Languages. However, given a general Context-Free Grammar, one cannot tell if this grammar describes a Regular Language or a Context-Free Language (this task was shown to be undecidable (Hopcroft, Motwani, and Ullman 2001)). We will now look at the property of *self-embedding* of the CFGs and see how this property helps in determining the class of the languages described by such CFGs.

Non-Self-Embedding Context-Free Grammars

Definition 3.4.1 *A Context-Free Grammar* $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ *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* $\alpha, \beta \neq \varepsilon$ *being any string of terminal and nonterminal symbols.*

For example, the nonterminal symbol P in the palindrome grammar (3.7) is such a self-embedding nonterminal, and the Context-Free Grammar of palindromes is self-embedding.

Definition 3.4.2 *A Context-Free Grammar* $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ *is* Non-Self-Embedding (NSE) *if there exists no nonterminal symbols for which the condition of the Def. 3.4.1 can be satisfied.*

Chomsky (1959a) has demonstrated that if a Context-Free Grammar is Non-Self-Embedding, it generates a Finite-State Language. In Chapter 5, we will describe an algorithm to verify the Non-Self-Embedding property of Context-Free Grammars, and show how to obtain Finite-State Automata for these grammars.

Context-Free Grammars and production graphs

The property of self-embedding of Context-Free Grammars introduced above 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*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

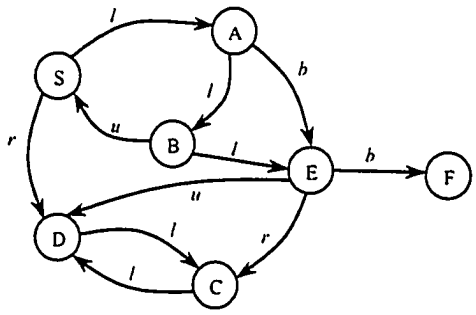

Figure 3.3: Production graph for the grammar in (3.9).

Definition 3.4.3 *A* production graph $P(G)$ *for a Context-Free Grammar* $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ *is a directed graph whose vertices correspond to the non-terminal 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* $A \rightarrow \alpha B \beta$.

Definition 3.4.4 *A* labeled production graph $P_l(G)$ *for a Context-Free Grammar (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:*

$$lab(A \rightarrow B) = \begin{cases} l & \text{if for every } A \rightarrow \alpha B \beta \in \Gamma,\ \alpha \neq \varepsilon,\ \beta = \varepsilon, \\ r & \text{if for every } A \rightarrow \alpha B \beta \in \Gamma,\ \alpha = \varepsilon,\ \beta \neq \varepsilon, \\ b & \text{if for every } A \rightarrow \alpha B \beta \in \Gamma,\ \alpha \neq \varepsilon,\ \beta \neq \varepsilon, \\ u & \text{if for every } A \rightarrow \alpha B \beta \in \Gamma,\ \alpha = \varepsilon,\ \beta = \varepsilon. \end{cases} \tag{3.8}$$

Note that the production graphs of Def. 3.4.3 and Def. 3.4.4 are not related to Finite-State Automata of Finite-State Languages described earlier. They are simply useful graphical representations of CFGs.

Let us consider an example grammar (reproduced from (Dilkes and Visnevski 2004) with modifications):

$$\begin{aligned} \mathcal{A} &= \{a, b, c, d\}\ , \\ \mathcal{E} &= \{S, A, B, C, D, E, F\}\ , \\ \Gamma &= \left\{ \begin{array}{l} S \rightarrow DA \\ A \rightarrow bEaB \\ B \rightarrow aE | S \\ C \rightarrow bD \\ D \rightarrow daC | a \\ E \rightarrow D | Cc | aF | Fc \\ F \rightarrow bd \end{array} \right\}. \end{aligned} \tag{3.9}$$

The labeled production graph for this grammar is illustrated in Fig. 3.3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Definition 3.4.5 *A* transition matrix $\mathbf{M}(G)$ *for the labeled production graph* $P_l(G)$ *of a Context-Free Grammar* $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ *is an* $N \times N$ *matrix whose dimensions are equal to the number of non-terminal symbols in* $\mathcal{E}$ *(number of vertices in the production graph), and whose elements are defined as follows:*

$$m_{i,j}(G) = \begin{cases} 0 & \text{if } A_i \rightarrow \alpha A_j \beta \notin \Gamma, \\ lab(A_i \rightarrow A_j) & \text{if } A_i \rightarrow \alpha A_j \beta \in \Gamma. \end{cases} \tag{3.10}$$

The transition matrix of the labeled production graph in Fig. 3.3 with respect to the vertex ordering $\{S, A, B, C, D, E, F\}$ has the following structure:

$$\mathbf{M}(G) = \begin{bmatrix} 0 & l & 0 & 0 & r & 0 & 0 \\ 0 & 0 & l & 0 & 0 & b & 0 \\ u & 0 & 0 & 0 & 0 & l & 0 \\ 0 & 0 & 0 & 0 & l & 0 & 0 \\ 0 & 0 & 0 & l & 0 & 0 & 0 \\ 0 & 0 & 0 & r & u & 0 & b \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.11}$$

In Chapter 5, we will use production graphs and transition matrices in analysis of self-embedding property of CFGs.

3.5 Stochastic languages and stochastic grammars

A number of practical applications contain certain amounts of uncertainty that are often represented by probabilistic distributions. For example, radar signals are typically observed in the noisy environment where signal interference may cause observation sparseness. These factors require extension of the concepts described above into the domain of stochastic languages.

Stochastic grammars

Definition 3.5.1 *A* weighted grammar G_w *is a five-tuple*

$$G_w = (\mathcal{A}, \mathcal{E}, \Gamma, P_w, S_0) \tag{3.12}$$

where:

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

$\mathcal{E}$ *is the set of non-terminal symbols of the grammar;*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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 non-terminal.*

Here is a simple example of a weighted grammar:

$$\begin{aligned} S_0 &\xrightarrow{9} aS_1 \\ S_0 &\xrightarrow{1} b \\ S_1 &\xrightarrow{2} bS_0 \\ S_1 &\xrightarrow{8} a \end{aligned} \tag{3.13}$$

This grammar has been obtained from grammar (3.5) by associating with its productions the set of weights $P_w = \{(9,1),(2,8)\}$. Note that the set of weights P_w does not have to be normalized.

Definition 3.5.2 *A* stochastic grammar G_s *is a five-tuple*

$$G_s = (\mathcal{A}, \mathcal{E}, \Gamma, P_s, S_0) \tag{3.14}$$

where $\mathcal{A}$, $\mathcal{E}$, Γ, *and* S_0 *are the same as in Def. 3.5.1, 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:

$$\begin{aligned} S_0 &\xrightarrow{0.9} aS_1 \\ S_0 &\xrightarrow{0.1} b \\ S_1 &\xrightarrow{0.2} bS_0 \\ S_1 &\xrightarrow{0.8} a \end{aligned} \tag{3.15}$$

This grammar has been obtained from grammar (3.5) by applying to its productions the probability distributions $P_s = \{(0.9,0.1),(0.2,0.8)\}$. Note that, like in the case of any proper probability distribution, the elements of the set P_s must be normalized (the sum of all probabilities associated with the productions originating from the same non-terminal must be equal to one).

Characteristic grammars

Stochastic and weighted grammars are classified and analyzed on the basis of their underlying *characteristic grammars* (Fu 1974; Fu 1982). A characteristic grammar G_c is obtained from the stochastic grammar G_s (weighted grammar

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

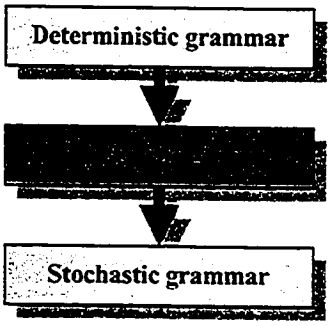

Figure 3.4: Derivation procedure for the stochastic grammars. First, a deterministic grammar for the system is constructed. Then, after considerations of possible sources of uncertainties, the deterministic grammar is modified into a characteristic grammar that accommodates for these uncertainties. Finally, a probability distribution is assigned to the characteristic grammar, yielding a stochastic grammar of the system.

G_w) by removing the probability distribution P_s (set of weights P_w) from the grammar definition.

If the resulting characteristic grammar is a Finite-State Grammar, the stochastic grammar is called Stochastic Finite-State Grammar (SFSG). If the characteristic grammar is Context-Free Grammar, the stochastic grammar is referred to as Stochastic Context-Free Grammar (SCFG). For example, grammar (3.15) is a SFSG, and grammar (3.5) is its characteristic grammar.

Characteristic grammars play important roles in deriving syntactic models of real-life systems. The typical procedure is illustrated in Fig. 3.4. 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.

Stochastic Regular Languages, Markov chains, and Hidden Markov Models

Just as Finite-State Automata constitute one of the representation forms of Finite-State Languages, discrete-state discrete-time Markov chains are naturally considered the equivalent representations of the Stochastic Finite-State Languages (Cassandras and Lafortune 1999). This representation has been successfully utilized in bioinformatics and computational genomics (Baldi and Brunak 2001; Durbin, Eddy, Krogh, and Mitchison 1998) as well as in natural language and speech processing (Deller, Hansen, and Proakis 1999).

Hidden Markov Models (HMMs) are particularly suitable for representing stochastic languages of the finite-state discrete-event systems observed in noisy environments. They enable to separate the uncertainty in the model attributed to the observation process from the uncertainties associated with the system's

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

functionality.

We will examine Markov chains and Hidden Markov Models in more detail in Chapter 4, and in Chapters 5 and 6, we will use HMMs as realizations of syntactic models of Multi-Function Radars.

3.6 Summary

This chapter presented essential elements of the theory of syntactic modeling. These elements contribute to the foundation of the model-centric Multi-Function Radar modeling methodology developed in this thesis.

We have started by looking at the concept of formal languages and their finite-dimensional models called grammars. We then reviewed two most useful classes of grammars – Finite-State Grammars and Context-Free Grammars. These grammars will be cornerstones of syntactic radar modeling developed in the chapters that follow.

We have also presented stochastic equivalents of formal languages and examined the relationships between languages and Finite-State Automata as well as stochastic languages and Hidden Markov Models. We will explore these relationships when building levels of the Multi-Function Radar model hierarchy in the rest of the thesis.

We must stress the point that this chapter only presented a brief overview of the most essential elements of syntactic modeling that were directly relevant to the radar modeling methodology developed in this thesis. Readers interested in a more rigorous treatment of the subject of syntactic modeling are encouraged to consult some of the sources listed at the end of this thesis. Specifically, the original publications by N. Chomsky (Chomsky 1956; Chomsky 1959a; Chomsky 1959b; Chomsky and Miller 1958) provide an excellent overview of the linguistic and grammatical hierarchies presented in this chapter.

Syntactic modeling and elements of syntax analysis are covered in great detail in (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Aho and Ullman 1973; Hopcroft, Motwani, and Ullman 2001; Hopcroft and Ullman 1979; Salomaa 1973). An in-depth treatment of Finite-State Automata and their relationship to Finite-State Languages and Finite-State Grammars is available in (Aho and Ullman 1972; Hopcroft, Motwani, and Ullman 2001; Hopcroft and Ullman 1979; Salomaa 1969; Salomaa 1985).

Elements of syntactic pattern recognition, stochastic languages, and stochastic grammars receive detailed treatment in (Fu 1974; Fu 1982).

Applications of stochastic languages and stochastic grammars to speech recognition and bioinformatics are presented in (Baldi and Brunak 2001; Deller, Hansen, and Proakis 1999; Durbin, Eddy, Krogh, and Mitchison 1998).

Finally, alternative models of formal languages that are structurally distinct from grammars and have some advantages in their descriptive power of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

formal languages were introduced by Lindenmayer (Lindenmayer 1968; Lindenmayer 1971; Lindenmayer 1975). They are commonly referred to as *L-systems*. Grammars from the Chomsky hierarchy presented in this chapter were shown to be properly contained within the class of L-systems (Goos and Hartmanis 1974; Rozenberg and Salomaa 1980; Salomaa 1981). This makes L-systems a more general and more complex class of models of formal languages.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Markov chains and Hidden Markov Models

In the previous chapter, we introduced some important elements of syntactic modeling. We identified discrete-state discrete-time Markov chains and Hidden Markov Models (HMMs) as important elements of syntactic modeling – they can be viewed as realizations of Stochastic Finite-State Languages.

In this chapter, we will introduce some elements from the theory of Markov chains and HMMs that will be used in the rest of this thesis. We will examine the concept of discrete-space discrete-time Markov chains and define HMMs on the basis of these Markov chains.

The rest of this chapter is organized as follows. Section 4.1 introduces Markov chains and defines them mathematically. Section 4.2 defines HMMs and describes some inference and analysis techniques that will be used later in the thesis. Finally, Section 4.3 summarizes the presented material and provides an overview of the literature that contains more in-depth treatment of the subjects addressed in this chapter.

4.1 Markov chains

A discrete-state discrete-time Markov chain can be defined as a stochastic timed automaton (Cassandras and Lafortune 1999):

Definition 4.1.1 *A* discrete-time Markov chain *defined over a discrete state space is a tuple*

$$\gamma = (\mathbf{A}, \pi), \tag{4.1}$$

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*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

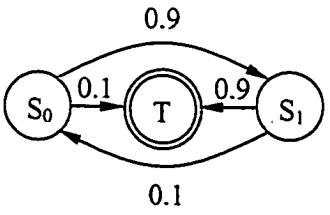

Figure 4.1: Example of a Markov chain for the Stochastic Finite-State Grammar (4.2). Note that Markov chains only capture the transition dynamics of the grammar since the terminal symbols of the grammar do not feature in the Markov chain structure.

N is the number of states in the Markov chain.

We will illustrate the relationship between Stochastic Finite-State Grammars and Markov chains by looking at the transition structure of the grammar (3.15) that we reproduce here for convenience:

$$G_s = \left(\begin{array}{rcl} \mathcal{A} & = & \{a,b\}, \\ \mathcal{E} & = & \{S_0,S_1\}, \\ \Gamma & = & \left\{ \begin{array}{lcl} S_0 & \stackrel{0.9}{\rightarrow} & aS_1, \\ S_0 & \stackrel{0.1}{\rightarrow} & b, \\ S_1 & \stackrel{0.1}{\rightarrow} & bS_0, \\ S_1 & \stackrel{0.9}{\rightarrow} & a \end{array} \right\}, \\ S_0 & & \end{array} \right). \tag{4.2}$$

We can construct a Markov chain that will reflect the transitions within Γ of (4.2) as

$$\gamma = \left(\mathbf{A} = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0.9 \\ 0 & 0 & 0 \end{bmatrix}, \pi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right), \tag{4.3}$$

where $\mathbf{A}$ and π are defined with respect to the state ordering $\{S_0, S_1, T\}$ as shown in Fig. 4.1.

The example above illustrates a strong parallel between Finite-State Automata (FSA) in the case of deterministic grammars, and Markov chains in the case of stochastic ones. However, Markov chains defined by Def. 4.1.1 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 Stochastic Finite-State Grammars discussed in Chapter 3. In the next section, we will see how extensions of Markov chains called Hidden Markov Models address this issue.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.2 Hidden Markov Models

Generally speaking, Hidden Markov Models (HMMs) are Markov chains indirectly observed through a noisy process (Cassandras and Lafortune 1999; Elliott, Aggoun, and Moore 1995; Ephraim and Merhav 2002; Rabiner and Juang 1993; Rabiner 1989). The underlying Markov chain can have either a continuous or a discrete state space. In the MFR modeling problem, we will only deal with the discrete state spaces.

Definition 4.2.1 *A* Hidden Markov Model λ *is a three-tuple*

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi), \tag{4.4}$$

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 Hidden Markov Models relate to Stochastic Finite-State Grammars, we would like to revisit the grammar (4.2). The Markov chain for this grammar is defined by (4.3). Now we can extend this chain bringing in the alphabet $\mathcal{A}$ of the grammar (4.2) 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. 4.1. Def. 4.2.1 associates the observation probability matrix $\mathbf{B}$ with the states of the chain, whereas Stochastic Finite-State Grammars 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* (Hopcroft and Ullman 1979). Therefore, to accommodate for the structural constraints of the Hidden Markov Model, the Markov chain in Fig. 4.1 has to be converted to the Moore machine form as described in detail in (Hopcroft and Ullman 1979).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

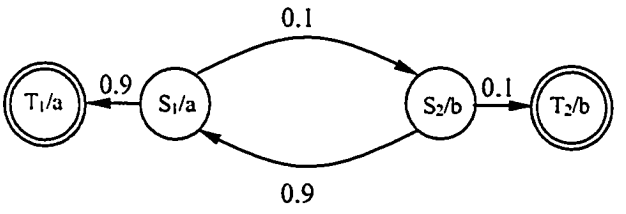

Figure 4.2: Example of a Hidden Markov Model for the Stochastic Finite-State Grammar (4.2). Each state is labeled by two symbols separated by a slash. The first symbol identifies the state of the system, and the second determines the output produced by the system in this state. To accommodate for the terminal symbols of the grammar (4.2) through the use of the observation probability matrix $\mathbf{B}$, the structure of the Markov chain in Fig. 4.1 had to be transformed to the Moore machine. Consequently, the underlying Markov chain of this Hidden Markov Model has different set of discrete states $\{S_1, S_2, T_1, T_2\}$.

The resulting HMM has the following structure:

$$\lambda = \left(\mathbf{A} = \begin{bmatrix} 0 & 0.1 & 0.9 & 0 \\ 0.9 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \pi = \begin{bmatrix} 0.9 \\ 0 \\ 0 \\ 0.1 \end{bmatrix} \right), \quad (4.5)$$

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. 4.2, and columns of $\mathbf{B}$ are defined with respect to the ordering $\{a, b\}$.

We will now examine some important properties of Hidden Markov Models and discuss two useful HMM-based inference techniques relevant to the problem of syntactic modeling of Multi-Function Radars.

Left-to-right and *ergodic* HMMs

Most useful HMMs can be classified into two distinct categories of models according to the structure of the transition probability matrix $\mathbf{A}$ of the underlying Markov chain (Rabiner and Juang 1993):

Left-to-right HMMs. For left-to-right HMMs all the states of the underlying Markov chains can be separated into two groups - the transient states, and the absorbing states. Typically, the chain starts in a particular state and as time progresses, it moves from one state to another, never returning to the already visited states, until it finally reaches one of the absorbing states. At that point transitions cease.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Ergodic HMMs. For ergodic HMMs every state of the underlying Markov chain can be reached from every other state in a finite number of time steps.

Two fundamental problems associated with HMMs

Given the HMM Def. 4.2.1, our primary focus will be on solving the following two fundamental problems (Rabiner and Juang 1993):

1. Given a sequence of observations $O = o_1 o_2 \dots o_T$ and the model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, how do we compute the probability $P(O|\lambda)$?
2. Given a sequence of observations $O = o_1 o_2 \dots o_T$ and the model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, how do we choose the corresponding sequence of states $q_1, \dots, q_T$ of the model λ that best explains the observations?

Computing $P(O|\lambda)$ with the forward algorithm

The general solution of the problem of computing the probability $P(O|\lambda)$ is given by the *forward algorithm* (Rabiner and Juang 1993). Let us define a *forward variable*

$$\alpha_t(i) = P(o_1, \dots, o_t, q_t = i|\lambda) \tag{4.6}$$

as the probability of the partial observation sequence $o_1, \dots, o_t$ and the state i at time t, given the model λ.

For the N-state HMM the forward algorithm is given by the following induction (Rabiner and Juang 1993):

1. Initialization:
$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N, \tag{4.7}$$
where $b_i(o_1)$ is the corresponding element of the $\mathbf{B}$ matrix.
2. Induction:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \quad \begin{matrix} 1 \leq j \leq N \\ 1 \leq t \leq T-1, \end{matrix} \tag{4.8}$$
where a_{ij} is the corresponding element of the $\mathbf{A}$ matrix, and T is the length of the discrete observation sequence.
3. Termination:
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{4.9}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Scaling in the forward algorithm

The forward algorithm implementation in the form presented above is not practical due to possibility of numerical divergence. As time progresses, the forward variable α becomes very small and quickly runs out of precision bounds of any digital computer. To avoid this, the forward variables are scaled at every time step by the factor (Rabiner and Juang 1993):

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}. \tag{4.10}$$

Using scaling factors (4.10), the practical value of the scoring probability can be calculated as follows (Rabiner and Juang 1993):

$$\log\left[P(O|\lambda)\right] = -\sum_{t=1}^{T} \log c_t. \tag{4.11}$$

Finding most likely state sequence with the Viterbi algorithm

The Viterbi algorithm finds the *maximum likelihood* estimate of the sequence of HMM states $Q = q_1, \ldots, q_T$ for the given observation sequence $O = o_1, \ldots, o_T$. This algorithm is an implementation of the Bellman's dynamic programming paradigm (Bertsekas 1996; Bertsekas 2001; Viterbi 1967).

One of the most serious issues of dynamic programming that inhibits its practical application is the so-called "curse of dimensionality". This issue is strongly related to tractability of the problem under study, and to computational complexity of the algorithms developed to solve the problem according to dynamic programming. The general question that one may pose is: *"how do we ensure that a particular implementation of the dynamic programming principle designed to solve a specific problem does not burn unnecessarily computing time and resources without any hope of ever producing a meaningful result?"*

To answer this question, we have to address the issue of mathematical tractability studied in detail by computer scientists. This theory classifies all computational problems into two main classes - *decidable* and *undecidable* problems. Decidable problems are those problems for which a computer algorithm can be developed, and undecidable problems are those for which no such algorithm exists.

The decidable problems are then classified into *tractable* and *intractable* ones. They are further sub-divided into classes of $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{NP}$-complete problems meaning, respectively, problems that can be solved in polynomial time, nondeterministic polynomial problems, and problems to which any $\mathcal{NP}$ problem can be reduced via a polynomial-time reduction operation (Cormen,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Leiserson, and Rivest 1998; Hopcroft, Motwani, and Ullman 2001; Knuth 1997). For problems in the $\mathcal{NP}$ class, there exists no polynomial-time algorithm that can solve them; but a polynomial-time algorithm can be found that can verify if any "guessed" solution is valid.

Only the problems of class $\mathcal{P}$ are considered tractable (Cormen, Leiserson, and Rivest 1998; Hopcroft, Motwani, and Ullman 2001). This brings us back to the question of tractability of dynamic programming posed earlier. The key to answering this question is in determining the class of the problem that the intended implementation is trying to solve. If the problem we are trying to address is in class $\mathcal{P}$, then the given implementation of the dynamic programming principle is acceptable.

For the case of the Viterbi algorithm applied to HMM processing, the exact upper bound complexity measure of $\mathcal{O}[LM^2]$ is shown in (Durbin, Eddy, Krogh, and Mitchison 1998; Rabiner and Juang 1993). Here L is the length of the observation data sequence, and M is the number of states of the HMM. Since this upper complexity bound is a polynomial in both L and M,by definition, the problem of finding the most likely state sequence of an HMM with the Viterbi algorithm belongs to the class $\mathcal{P}$ and is therefore computationally tractable.

Furthermore, additional reduction of complexity of this algorithm applied to the Multi-Function Radar signal processing problem can be achieved through careful analysis and utilization of the structure of the problem. In Section 9.1, we show how a variation of the Viterbi algorithm with an almost linear complexity in both L and M can be achieved.

Let us begin the introduction of the Viterbi algorithm with the definitions of some fundamental variables. For a given state i of the HMM, let us define the highest probability (score) at time t, which accounts for the first t observations (Rabiner and Juang 1993):

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P\left[q_1,\ldots,q_t = i, o_1,\ldots,o_t|\lambda\right] . \tag{4.12}$$

This score can be evaluated recursively as

$$\delta_{t+1}(j) = \max_i[\delta_t(i)a_{ij}]b_j(o_{t+1}) , \tag{4.13}$$

where a_{ij} and $b_j(o_{t+1})$ are, respectively, corresponding elements of the $\mathbf{A}$ and $\mathbf{B}$ matrices.

In addition to the score, the actual arguments that maximize (4.13) have to be accumulated as $\psi_t(j)$. These arguments will be used in the backtracking phase of the algorithm to retrieve the best state sequence.

For the N-state HMM the Viterbi algorithm is given by the following recursion (Rabiner and Juang 1993):

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \tag{4.14}$$
$$\psi_1(i) = 0, \tag{4.15}$$

where δ is the initial score, and ψ is the initial index of the maximum likelihood state.

2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t), \tag{4.16}$$
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \tag{4.17}$$
$$2 \leq t \leq T, \quad 1 \leq j \leq N,$$

where δ_t is the intermediate score, and ψ_t is the index of the maximum likelihood state at time t.

3. Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)], \tag{4.18}$$
$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)], \tag{4.19}$$

where P^* is the final score, and q_t^* is the the maximum likelihood state at time T.

4. Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1, \tag{4.20}$$

where q_t^* is the the maximum likelihood state at time t.

Scaling in the Viterbi algorithm

The Viterbi algorithm presented above suffers from the same numerical divergence problems as the forward algorithm presented earlier in this chapter. However, the problem can easily be solved by employing logarithms of scores $\delta_t(i)$ instead of the scores themselves. No numerical scaling is required in this case (Rabiner and Juang 1993).

4.3 Summary

In this chapter, we have introduced Markov chains and Hidden Markov Models (HMMs) and shown that HMMs can be considered as Markov chains indirectly observed in the noisy environment. We have also demonstrated the links

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of Markov chains and HMMs to Stochastic Finite-State Grammars. Markov chains can capture dynamics of the production rules of Stochastic Finite-State Grammars, but fail to accommodate for the terminal symbols of the grammar. HMMs, on the other hand, can include the terminal symbols of the grammar, but require conversion of the underlying Markov chain to a Moore machine form. Such conversion can easily be performed in an automatic fashion using a polynomial-time Mealy-to-Moore conversion procedure described in (Hopcroft and Ullman 1979). This makes Hidden Markov Models ideal tools for modeling Stochastic Finite-State Languages.

We also considered two fundamental problems of statistical inference commonly associated with HMMs – the problem of scoring an observation sequence (finding the probability that this sequence was generated by a given HMM), and the problem of finding the most likely sequence of HMM states given an observation sequence. We then discussed two algorithms that solve these problems. The problem of sequence scoring can be solved by the forward algorithm, and the problem of finding the most likely sequence of states can be solved using the Viterbi algorithm. We will apply these statistical inference algorithms in the context of radar signal processing later in this thesis.

This chapter only covered the minimal amount of material necessary for the development of the syntactic radar modeling framework. More detailed coverage of Markov chains and Hidden Markov Models can be found in (Cassandras and Lafortune 1999; Elliott, Aggoun, and Moore 1995; Ephraim and Merhav 2002; Rabiner and Juang 1993; Rabiner 1989). In addition to these resources, several very powerful applications of HMMs (biological sequence analysis and speech processing) are described in (Baldi and Brunak 2001; Deller, Hansen, and Proakis 1999; Durbin, Eddy, Krogh, and Mitchison 1998).

Curious readers may find the new theory of Observable Operator Models (OOMs) published by Jaeger (2000) very promising. In this paper, OOMs were shown to properly contain the class of Hidden Markov Models. They were also shown to be more powerful statistical models able to capture the properties of certain stochastic processes that HMMs cannot address.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Part II

Syntactic radar modeling

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

Word-level modeling of Multi-Function Radars

In Chapter 2, we introduced a hierarchical modeling approach to Multi-Function Radars (MFRs) and identified word-level models as the ones that hold the clue to the internal processes happening within the radar. The concept of a *radar word* was defined as a fixed (static) or dynamically varying sequence of radar pulses that forms a consistent pattern within the radar signal observations. Therefore, radar words, in syntactic terms, can be viewed as elements of the *alphabet* (or the *vocabulary*) of the radar.

One of the main contributions of our work is the consideration of the dynamics of radar signals on the word-level from the standpoint of a linguistic formalism (Fu 1974; Fu 1982; Hopcroft and Ullman 1979). We view MFRs as discrete event systems that "speak" some known, or partially known, formal language (Cassandras and Lafortune 1999). The sequences of observations of MFR signals are viewed as strings from this language corrupted by the measurement noise. Using the *a priori* intelligence about a particular MFR signal, we can derive a grammar that describes the radar language. We refer to such a modeling approach as *syntactic modeling* of Multi-Function Radars. Syntactic models are compact formal representations that can form a homogeneous basis for modeling complex radar dynamics.

Radar systems, as any digital finite-memory systems, are expected to behave in a causal or Markovian fashion, and, as a consequence, one expects the language associated with MFRs to be a Finite-State Language (FSL). At the same time, some radar systems may exhibit a very complex signal behavior that cannot be easily captured by the FSL definitions. Indeed, it may be simpler and more natural for the ES analyst to represent them as Context-Free Grammars (CFGs) (Aho and Ullman 1972; Fu 1982; Hopcroft and Ullman 1979) - a very flexible framework that accommodates most of the features of interest for this application.

Although CFGs may provide a compact representation of radar languages,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

they are associated with computationally complex signal-processing algorithms (Aho and Ullman 1972; Durbin, Eddy, Krogh, and Mitchison 1998; Fu 1982; Hopcroft and Ullman 1979). 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 (Visnevski, Krishnamurthy, Haykin, Currie, Dilkes, and Lavoie 2003)), but the associated signal-processing techniques are much less computationally demanding (see discussion on complexity of syntactic parsing algorithms in (Durbin, Eddy, Krogh, and Mitchison 1998)). It would therefore be advantageous to model MFRs using CFGs, but to perform radar signal processing on their finite-state equivalents.

In this chapter, we present a procedure that allows to automatically synthesize a finite-state model of an MFR using its CFG model as an input. We introduce a theoretical framework for determining whether a specified Context-Free Grammar of the radar actually represents a Finite-State Language and provide an automated polynomial-time algorithm for generating the corresponding finite-state automaton.

This synthesis procedure consists of four basic steps:

1. **Test of self-embedding.** A Context-Free Grammar that is determined to be Non-Self-Embedding describes a Finite-State Language (see Chapter 3). Therefore, a finite-state automaton can be synthesized from this grammar.

2. **Grammatical decomposition.** First, the Non-Self-Embedding Context-Free Grammar is broken down into a set of simpler Finite-State Grammars.

3. **Component synthesis.** Once the grammar has been decomposed into a set of simpler grammars, a finite-state automaton can be synthesized for every one of these Finite-State Grammars.

4. **Composition.** Finally, the components from the previous step are combined together to form a single Finite-State Automaton that is equivalent to the original Non-Self-Embedding Context-Free Grammar of the Multi-Function Radar.

The rest of this chapter is organized as follows. In Section 5.1, we present a polynomial-time algorithm that verifies an important property of a Context-Free Grammar - the property of self-embedding. Section 5.2 presents the decomposition procedure that allows to break any Non-Self-Embedding Context-Free Grammar into a set of Finite-State Grammars. Section 5.3 demonstrates how to synthesize a set of Finite-State Automata for these Finite-State Grammars. Section 5.4 describes the composition step of the procedure. Finally, Section 5.5 offers some concluding remarks.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.1 Verification of Non-Self-Embedding

As mentioned in Chapter 3, if the Context-Free Grammar (CFG) of the radar is in the Non-Self-Embedding (NSE) form, the 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.

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 (Anselmo, Giammarresi, and Varricchio 2003), but it has been modified to suite the needs of the radar grammars.

Let us start by defining the concept of a *semi-ring* (Kuich and Salomaa 1986):

Definition 5.1.1 *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:* $(\forall e, g, f \in S)\,(e \times g) \times f = e \times (g \times f)$, *and*
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 semi-ring over the set of labels of production graphs of CFGs. This set of labels is introduced by Def. 3.4.4 and Def. 3.4.5. The *sum* and *product* operations of this semi-ring are listed in Table 5.1[1].

If $\mathrm{M}(G)$ is a transition matrix of the CFG G (see Def. 3.4.5), then, using the semi-ring operations of Table 5.1, we define the steady-state matrix of the production graph of this grammar as

$$\mathrm{M}^{\leq N}(G) = \sum_{i=1}^{N} [\mathrm{M}(G)]^i \,, \tag{5.1}$$

where N is the dimension of the transition matrix $\mathrm{M}(G)$[2].

[1] We observe that mathematically this is a curious-looking structure that is more than just a semi-ring. It contains identity elements for both addition and multiplication operations, but does not contain inverse elements that are required for this structure to be called a *ring*.

[2] Remember that the elements of the transition matrix are the labels of the edges of the production graph. The definition of the semi-ring over the set of labels of the production graph in this case simply define the meaning of mathematical operations of addition and multiplication of the transition matrix so that the expression of the steady-state matrix can be formulated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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

Table 5.1: Semi-ring operations of sum and product.

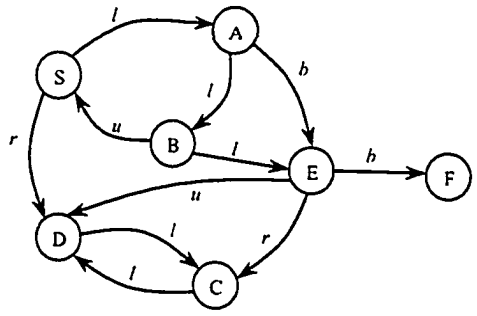

Figure 5.1: Production graph for the grammar in (5.2).

Anselmo, Giammarresi, and Varricchio (2003) have proven that if

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

does not contain labels 'b', the corresponding grammar G is Non-Self-Embedding. This demonstrates that the Non-Self-Embedding property of a Context-Free Grammar can be verified in polynomial time.

To illustrate the application of this algorithm, let us revisit the example CFG (3.9) (reproduced here for convenience):

$$\Gamma = \left\{ \begin{array}{l} S \rightarrow DA \\ A \rightarrow bEaB \\ B \rightarrow aE|S \\ C \rightarrow bD \\ D \rightarrow daC|a \\ E \rightarrow D|Cc|aF|Fc \\ F \rightarrow bd \end{array} \right\}. \tag{5.2}$$

The labeled production graph for this grammar is shown in Figure 5.1, and the transition matrix of this graph with respect to the vertex ordering

$$\{S, A, B, C, D, E, F\}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

has the following structure:

$$\mathbf{M}(G) = \begin{bmatrix} 0 & l & 0 & 0 & r & 0 & 0 \\ 0 & 0 & l & 0 & 0 & b & 0 \\ u & 0 & 0 & 0 & 0 & l & 0 \\ 0 & 0 & 0 & 0 & l & 0 & 0 \\ 0 & 0 & 0 & l & 0 & 0 & 0 \\ 0 & 0 & 0 & r & u & 0 & b \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.3}$$

$\mathbf{diag}\left[\mathbf{M}^{\leq N}(G)\right] = [l, l, l, l, l, 0, 0]$, therefore, CFG (5.2) is Non-Self-Embedding. In the following section, we will describe a 3 step procedure that accepts an NSE CFG and automatically synthesizes a Finite-State Automaton (FSA) that is equivalent to this grammar.

5.2 Grammatical decomposition

In this section, we show that given a Non-Self-Embedding Context-Free Grammar we may decompose this grammar into a set of Finite-State Grammars in polynomial time. We start by introducing the concept of the grammatical $\oplus$-*composition.*

Definition 5.2.1 *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 Context-Free Grammars 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* $S = S_1$.

Anselmo, Giammarresi, and Varricchio (2003) have demonstrated that for any Non-Self-Embedding Context-Free Grammar G there exist n Finite-State Grammars (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 $P_i(G)$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

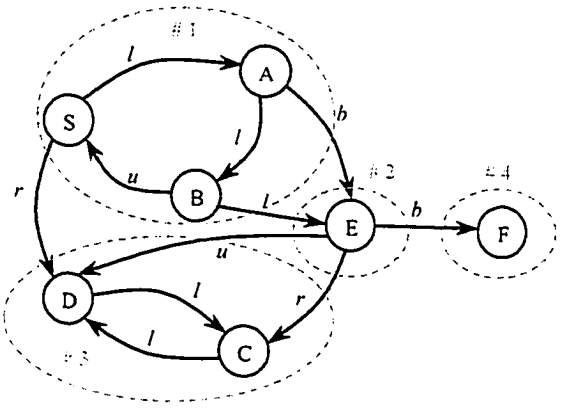

Figure 5.2: Strongly-connected components of the production graph shown in Figure 5.1.

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 $\Gamma_i = \{A \rightarrow \alpha | A \in \mathcal{E}_i\}$.

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 (Pothen and Fan 1990) 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.

Note that the Dulmage-Mendelsohn decomposition as well as the grammatical decomposition procedures described above have polynomial complexity. Therefore, the grammatical decomposition method presented in this chapter is an efficient and practical tool for characterization of Non-Self-Embedding Context-Free Grammars.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Now consider an example of the decomposition procedure applied to grammar (5.2). Its production graph includes four strongly-connected components shown in Figure 5.2. The four Finite-State Grammar components of this Context-Free Grammar are:

$$G_1 = \left(\begin{array}{rcl} \mathcal{A}_1 & = & \{a,b,c,d,C,D,E\}, \\ \mathcal{E}_1 & = & \{S,A,B\}, \\ \Gamma_1 & = & \left\{ \begin{array}{l} S \rightarrow DA, \\ A \rightarrow bEaB, \\ B \rightarrow aE|S \end{array} \right\}, \\ S_1 & = & S \end{array} \right), \tag{5.4a}$$

$$G_2 = \left(\begin{array}{rcl} \mathcal{A}_2 & = & \{a,b,c,d,C,D,F\}, \\ \mathcal{E}_2 & = & \{E\}, \\ \Gamma_2 & = & \{\ E \rightarrow D|Cc|aF|Fc\ \}, \\ S_2 & = & E \end{array} \right), \tag{5.4b}$$

$$G_3 = \left(\begin{array}{rcl} \mathcal{A}_3 & = & \{a,b,c,d\}, \\ \mathcal{E}_3 & = & \{C,D\}, \\ \Gamma_3 & = & \left\{ \begin{array}{l} C \rightarrow bD, \\ D \rightarrow daC|a \end{array} \right\}, \\ S_3 & = & \{X|X \in \mathcal{E}_3\} \end{array} \right), \tag{5.4c}$$

$$G_4 = \left(\begin{array}{rcl} \mathcal{A}_4 & = & \{a,b,c,d\}, \\ \mathcal{E}_4 & = & \{F\}, \\ \Gamma_4 & = & \{\ F \rightarrow bd\ \}, \\ S_4 & = & F \end{array} \right), \tag{5.4d}$$

where, in (5.4c), X may represent either of the nonterminals contained in $\mathcal{E}_3$.

5.3 Synthesis of finite-state components

The next step involves synthesis of individual Finite-State Automata (FSA) for each of the Finite-State Grammars (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 (Aho, Sethi, and Ullman 1986; Aho and Ullman 1972; Fu 1974; Fu 1982; Hopcroft, Motwani, and Ullman 2001; Hopcroft and Ullman 1979).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

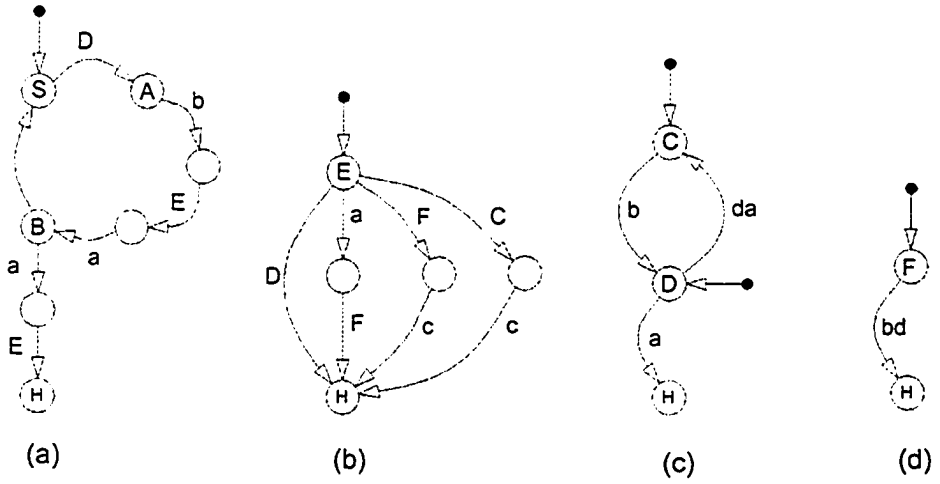

Figure 5.3: Components of the finite-state automaton for the grammar (5.2). States that are not labeled in these graphs are considered intermediate and have no direct representation in the set of nonterminals of the grammars. (a) corresponds to the grammar (5.4a), (b) corresponds to the grammar (5.4b), (c) corresponds to the grammar (5.4c), and (d) corresponds to the grammar (5.4d).

The FSA for the example grammars (5.4) are shown in Figure 5.3. They have the following structure:

$$\Lambda_1 = \begin{pmatrix} \Sigma_1 &= \mathcal{A}_1, \\ Q_1 &= \mathcal{E}_1 \cup Q_1', \\ \delta_1 &= \delta(\Gamma_1), \\ q_0 &= \{S\}, \\ F_1 &= H \in Q_1' \end{pmatrix}, \tag{5.5a}$$

$$\Lambda_2 = \begin{pmatrix} \Sigma_2 &= \mathcal{A}_2, \\ Q_2 &= \mathcal{E}_2 \cup Q_2', \\ \delta_2 &= \delta(\Gamma_2), \\ q_0 &= \{E\}, \\ F_2 &= H \in Q_2' \end{pmatrix}, \tag{5.5b}$$

$$\Lambda_3 = \begin{pmatrix} \Sigma_3 &= \mathcal{A}_3, \\ Q_3 &= \mathcal{E}_3 \cup Q_3', \\ \delta_3 &= \delta(\Gamma_3), \\ q_0 &= \{\{X\} | X \in \mathcal{E}_3\}, \\ F_3 &= H \in Q_3' \end{pmatrix}, \tag{5.5c}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\Lambda_4 = \begin{pmatrix} \Sigma_4 = \mathcal{A}_4, \\ Q_4 = \mathcal{E}_4 \cup Q_4', \\ \delta_4 = \delta(\Gamma_4), \\ q_0 = \{F\}, \\ F_4 = H \in Q_4' \end{pmatrix}, \tag{5.5d}$$

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 FSA highlighting only the important structural aspects. Specifically, in Figures 5.3 (c) and (d) transitions labeled with nonterminals '*da*' and '*bd*', when rigorously treated, require the insertion of an intermediate state. Also, before constructing a FSA, the grammar should have been converted to the unit production free form so that every edge in the graphs in Figure 5.3 corresponds to the generation of a single nonterminal. We will suppress these intermediate steps in the rest of this thesis, and, without loss of generality, will adopt a slightly simplified form of FSA representation.

5.4 Composition of the Finite-State Automaton

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 Nederhof (2000) and its modified version was presented by Dilkes and Visnevski (2004). However, here we present an alternative, "breadth-first" algorithm.

The objective of this chapter is to explain the basic principle of the radar model synthesis procedure. In this context, the breadth-first algorithms are typically more compact than their depth-first counterparts; however, the breadth-first algorithms have higher memory demands. The simplicity and compactness of the breadth-first algorithm that we are about to describe allows us to meet the main objective of the chapter.

The FSA composition procedure is formalized in terms of two algorithms presented on page 56. 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. 5.3.

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 (5.2). This FSA is shown in Fig. 5.4 (a).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Algorithm 1 Procedure "createFSA" creates a Finite-State Automaton.

```
1: procedure CREATEFSA(S_0, Λ_1, ..., Λ_n)        ▷ Creates an FSA from components Λ_1, ..., Λ_n
2:     Σ ← {S_0}                                   ▷ Initializing set of transitions
3:     Q ← {q', H}                                 ▷ Adding intermediate states
4:     δ ← {δ(q', S_0) = {H}}                      ▷ Adding initial transition
5:     q_0 ← q'                                    ▷ Setting initial state
6:     F ← {H}                                     ▷ Setting terminal states
7:     Λ ← (Σ, Q, δ, q_0, F)                       ▷ Initializing the FSA
8:     Λ ← expandFSA(Λ, Λ_1, ..., Λ_n)             ▷ Calling the expansion procedure
9: end procedure
```

Algorithm 2 Function "expandFSA" inserts FSA components into the Finite-State Automaton Λ.

```
1: function EXPANDFSA(Λ, Λ_1, ..., Λ_n)            ▷ Inserts FSA comps into Λ
2:     for all Λ_i do
3:         for all α ∈ Σ do
4:             if α ∈ Q_i then                     ▷ If transition matches a state of Λ_i
5:                 q_from ← arg_q(δ(q_α, α))       ▷ Saving from state
6:                 q_to ← δ(q_from, α)             ▷ Saving to state
7:                 Σ ← Σ \ α ∪ Σ_i                 ▷ Expanding set of transitions
8:                 Q ← Q ∪ Q_i                     ▷ Expanding set of states
9:                 δ ← δ ∪ δ_i                     ▷ Appending transition structure
10:                for all δ(q_j, β) do
11:                    if δ(q_j, β) = q_from then
12:                        δ(q_j, β) ← α           ▷ Rerouting q_from inputs
13:                    end if
14:                    if δ(q_j, β) ∈ F_i then
15:                        δ(q_j, β) ← q_to        ▷ Rerouting q_to inputs
16:                    end if
17:                end for
18:                Q ← Q \ F_i                     ▷ Removing term. states of Λ_i
19:                Q ← Q \ q_from                  ▷ Removing q_from state
20:            end if
21:        end for
22:    end for
23:    return Λ                                    ▷ Returning the complete FSA Λ
24: end function
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

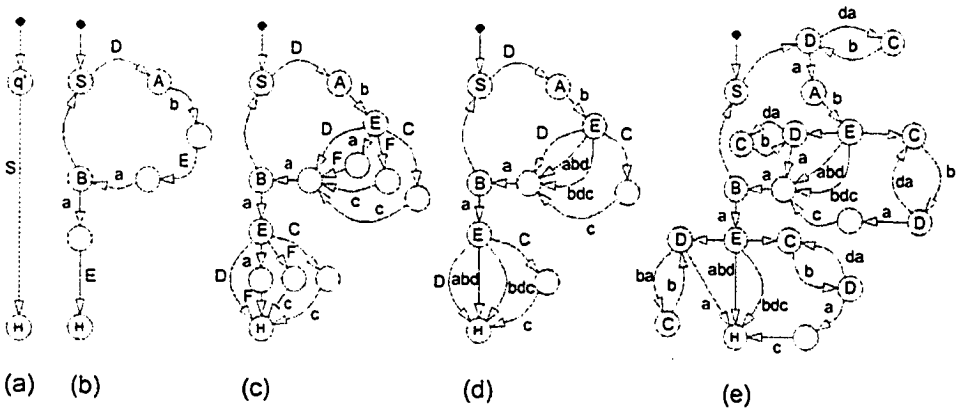

Figure 5.4: Synthesis procedure of the Finite-State Automaton for the example CFG (5.2).

The function "expandFSA" accepts the "dummy" FSA as well as four FSA components shown in Fig. 5.3. 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. 5.4 (b)–(e). First, the FSA shown in Fig. 5.3 (a) is inserted into the FSA in Fig. 5.4 (a) instead of the transition labeled S. The resulting intermediate FSA is shown in Fig. 5.4 (b). Next, all the E-transitions are replaced with the FSA in Fig. 5.3 (b). The resulting intermediate FSA is shown in Fig. 5.4 (c). Then, all the F-transitions are replaced with the FSA in Fig. 5.3 (d). The resulting intermediate FSA is shown in Fig. 5.4 (d). Finally, all the C- and D-transitions are replaced with the FSA in Fig. 5.3 (c). The final automaton equivalent to the grammar (5.2) is shown in Fig. 5.4 (e). Note that the labels of the states in Fig. 5.4 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. 5.3.

5.5 Summary

This chapter presented a novel word-level modeling technique for Multi-Function Radars. This approach offers two important benefits – the compactness of Context-Free Grammar models and the relatively low computational cost of the signal processing algorithms that utilize finite-state models.

Fig. 5.4 (e) clearly demonstrates the compactness property of the Non-Self-Embedding Context-Free Grammars. The original grammar (5.2) is made up of seven nonterminals. However, if we were to construct the Finite-State Grammar for the automaton in Fig. 5.4 (e), the number of non-terminals in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

this grammar will exceed thirty. As we will see in Chapter 8, this property of compactness offers serious benefits to the radar model designers enabling derivation of parsimonious syntactic radar models from textual descriptions of high-level radar functionality.

Another important benefit of the syntactic modeling approach presented in this chapter is the offer of homogeneous modeling basis for Multi-Function Radars. Regardless of distinctions in functionality of MFRs, as long as there is enough prior knowledge of its operation, a finite-state model of this radar can be synthesized in polynomial time. Until now, this mechanism has not been available in the field of Electronic Warfare.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 6

Pulse-level modeling of Multi-Function Radars

The word-level modeling approach described in the previous chapter relies on the sequences of radar words that have to be supplied by the pulse-level components of the hierarchical radar model introduced in Chapter 2. The novel, model-centric approach to pulse-level radar modeling is the main focus of this chapter.

Although pulse-level components of the hierarchical model presented in Chapter 2 are identified and described first, we view pulse-level modeling as a supporting task to the problem of word-level modeling. As mentioned in Chapter 1, the main focus of this thesis is modeling word-level functionality of the radars. Therefore, we chose to develop word-level modeling methodology first and deferred the treatment of the pulse-level modeling until this chapter.

The rest of this chapter is organized as follows. Section 6.1 introduces the problem of *pulse train analysis*. Pulse-level models are primarily used for the purpose of pulse train analysis when the sequences of radar words have to be extracted from noisy and corrupted observations of radar pulses. We also discuss some of the important issues related to pulse train analysis that affect pulse-level modeling.

In Section 6.2, we develop a syntactic pulse-level radar model using the tools introduced in Chapters 3 and 4. We view the problem of extraction of words from the radar pulse sequences as a problem of pattern matching. The standard technique of syntactic pattern matching involves application of *regular expressions* introduced in Chapter 3. We show how to build regular expressions for pulse templates of radar words, and how to convert these regular expressions into templates based of Hidden Markov Models (HMMs).

Finally, Section 6.3 offers a summary of the key results of this chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

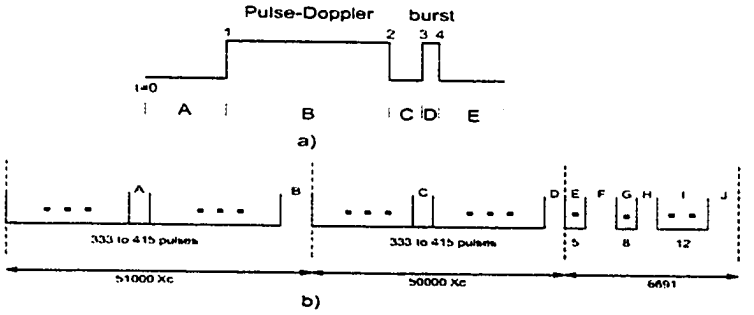

Figure 6.1: Sample Multi-Function Radar words. Words of the first emitter have a common pulse envelope shown in (a). It includes 5 distinct sections. Words of another radar are structured in pairs separated by a termination character, as shown in (b). For more details of the structure of these words see Appendices A and B.

6.1 The problem of pulse train analysis

A generic Electronic Warfare (EW) system includes three main components - a set of receivers, a processor, and an interface. EW systems often utilize both narrow- and wide-band receivers fed by multiple antennas positioned to provide a wide field-of-view. The wide-band receivers detect radar pulses over a very wide frequency range, whereas narrow-band receivers are tuned to detect continuous-wave and pulse-Doppler radar signals (Adamy 2002). Received signals are digitized, buffered, and then processed in software. The processor classifies each signal source and estimates its intent. The results are sent to an operator for situational awareness and action.

Due to signal density and parametric overlap, signals coming out of the receivers often need to be separated into "tracks" that correspond to unique emitters. When this separation is carried out on pulsed signals in the time domain, it is called *de-interleaving*. This problem has been studied in the past and is discussed in detail in (Clarkson, Perkins, and Mareels 1996; Moore and Krishnamurthy 1994).

After identification of the radar emitters and separation of their respective signals into individual tracks, the sequence of radar words must be extracted from the sequence of recorded pulses. The concept of radar words has been introduced and discussed in detail in Chapter 2. As a refresher, in Fig. 6.1 we illustrate sample word structures of two different MFRs discussed earlier (Fig. 2.1 and Fig. 2.3).

There are several challenges that must be overcome for proper radar word sequence recognition:

1. Radar pulses are observed in a stochastic environment, the characteristics of which may not be known.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2. De-interleaving algorithms may fail to correctly separate radar pulses of distinct emitters. De-interleaving errors occur if pulses originating from one emitter "leak" into tracks that are predominantly associated with another emitter.

3. The received sequence of pulses is subject to electromagnetic signal reception and quantization distortions.

4. Since the reception of the radar signal by the EW system is a non-cooperative process, there is no synchronization between the EW receiver and the radar emitter.

We address the first two challenges by using probabilistic models of electromagnetic pulse propagation channels. A simple binary or binary erasure channel model may be used to tackle the first challenge. More sophisticated models like Markov-modulated channels may be used to emulate the effect of imperfect de-interleaving. Both here, and in the rest of this thesis, we consider a binary channel in which the probability that nothing is received when the radar emitted a pulse is denoted by p_{miss}, and spurious pulses follow a Poisson process whose average density is ρ pulses per unit of time[1]. The choice of the Poisson process is dictated by the nature of the binary channel model. As Cassandras and Lafortune (1999) have demonstrated, Poisson processes are the key stochastic models that support Markovian dynamics such as that of the binary channel that we intend to utilize in our modeling approach.

The statistical model of the binary channel that we utilize in our approach is illustrated in Fig. 6.2. The left-hand side of the trellis structure shows the symbols actually emitted by the radar, and the right-hand side shows the symbols that were observed by the EW receiver. The channel introduces errors which are shown as arrows of the trellis. Symbol '0' can be erroneously received as '1' with the probability of p_{spur}, and correctly received as '0' with the probability $1 - p_{\text{spur}}$. By the same token, symbol '1' can be erroneously received as '0' with the probability of p_{miss}, and correctly received as '1' with the probability $1 - p_{\text{miss}}$. Here p_{spur} is the instantaneous sample probability of the Poisson process mentioned earlier. A formal definition of this probability presented in (6.9) later in this chapter.

Some radar signal processing literature refers to a similar set of probabilistic binary channel characteristics – the probability of pulse detection p_{d} and the

[1]Note that we consider both the pulses induced in the receiver due to thermal noise peaks and the pulses that leaked into the pulse train of the emitter of interest due to the de-interleaver errors as spurious pulses. Given that the main focus of this work is on word-level processing, we only consider the simplest possible pulse-train analysis scenario when the spurious pulse density is approximated by the simple Poisson process. More accurate thermal noise and correlated pulse leakage models as well as errors due to multi-path propagation effects are outside the scope of this thesis.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

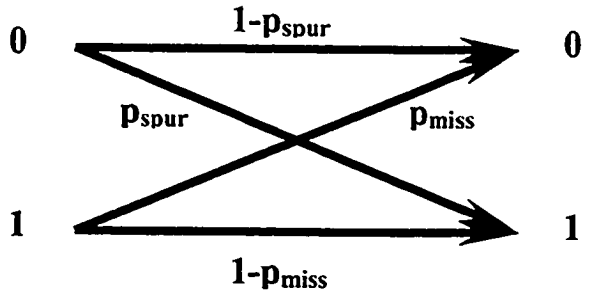

Figure 6.2: Probabilistic binary channel model for radar pulse reception process. Left-hand side of the trellis represents symbols actually emitted by the radar, and the right-hand side represents the detected symbols. Another way of viewing this model is in terms of probability of detection $p_{\mathrm{d}} \equiv 1 - p_{\mathrm{miss}}$ and the the probability of false alarm $p_{\mathrm{fa}} \equiv p_{\mathrm{spur}}$.

probability of false alarm p_{fa}. These probabilities are related to p_{miss} and p_{spur} in the following way:

$$p_{\mathrm{d}} \equiv 1 - p_{\mathrm{miss}}, \tag{6.1}$$

$$p_{\mathrm{fa}} \equiv p_{\mathrm{spur}}. \tag{6.2}$$

To address the issue of quantization distortions, we must examine the specific hardware implementation of the EW receivers. In most EW hardware, the arrival time of every pulse is quantized by a process driven by a master clock (Lavoie 2001). Denoting by '1' an observation period during which a leading edge is detected, and by '0' all other periods, the quantization process transforms a sequence of noisy leading-edge time of arrivals into a *binary observation sequence.*

We model the quantized and asynchronous nature of time stamping by defining the *quantization index* random variable

$$n_i(\varphi) \equiv \left\lfloor \frac{t_i + \varphi}{T_{\mathrm{obs}}} \right\rfloor, \tag{6.3}$$

where

$n = \lfloor x \rfloor$ denotes the largest integer n such that $n \leq x$,

t_i is the relative Time-Of-Arrival (TOA) of the i^{th} pulse,

φ is a uniformly distributed random variable corresponding to the phase offset between the emitter and the observer, $\varphi \in [0, T_{\mathrm{obs}})$, and

T_{obs} is the period of the master clock.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The probability distribution for this discrete random variable is given by

$$\begin{cases} P[n_i(\varphi) = \lfloor \frac{t_i}{T_{\text{obs}}} \rfloor] &= 1 - p_i \,, \\ P[n_i(\varphi) = \lceil \frac{t_i}{T_{\text{obs}}} \rceil] &= p_i \,, \end{cases} \tag{6.4}$$

where

$$p_i \equiv \text{frac}\left\{\frac{t_i}{T_{\text{obs}}}\right\} \equiv \frac{t_i}{T_{\text{obs}}} - \left\lfloor \frac{t_i}{T_{\text{obs}}} \right\rfloor \,, \tag{6.5}$$

is the probability of a positive quantization error (*pulse splitting*)for the i^{th} pulse, $n = \lfloor x \rfloor$ is the largest integer such that $n \leq x$, and $n = \lceil x \rceil$ is the smallest integer such that $n \geq x$.

6.2 Syntactic pulse-level model

In light of the discussion above, we may view radar words as simply sequences of arrival times which, after quantization, are represented by binary sequences.

The leading edge of an individual pulse followed by the pulse itself and then some "dead" time before the next leading edge is called a *pulse interval*. In the pulse sequence, the ith pulse interval is matched by the following regular expression[2]:

$$(0 + \epsilon)10^{l_{i,i+1}} \tag{6.6}$$

where $l_{i,i+1}$ is the number of zeros in the binary sequence that corresponds to the dead time before the next pulse. This number depends on the quantization precision (T_{obs}). The $(0 + \epsilon)$ element of the regular expression accommodates for the possibility of the pulse splitting as discussed in the previous section.

Each radar word is composed of several sequential pulse intervals. Therefore, the entire word can be matched by a sequence of regular expressions similar to (6.6), concatenated together. For example, given a simple pulse TOA sequence $[50\mu s, 70\mu s, 101\mu s, 117\mu s]$, the observer clock period $T_{obs} = 1.1\mu s$, and the quantization process defined by (6.3), the following regular expression

$$0^{44}(0 + \epsilon)10^{18}(0 + \epsilon)10^{28}(0 + \epsilon)10^{15}(0 + \epsilon)1$$

will match the quantized sequence of pulses regardless of the quantization phase φ.

Exploiting the relationship between regular expressions, Finite-State Automata, and Hidden Markov Models discussed in Chapter 3, we may represent a single pulse interval regular expression (6.6) by the Markov chain shown in

[2]Regular expressions were introduced and demonstrated to be representations of the Finite-State Languages in Chapter 3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

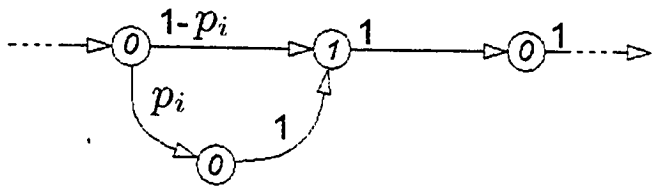

Figure 6.3: Pulse interval Markov chain representing i^{th} pulse.

Fig. 6.3. The transition matrix for the Markov chain of the i^{th} pulse interval within the k^{th} radar word is given by

$$\mathbf{A}'_{k,i} = \begin{pmatrix} 0 & p_i & 1-p_i & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{6.7}$$

where p_i is defined by (6.5).

This *pulse interval Markov chain* is a left-to-right Markov chain (Rabiner and Juang 1993), owing to the super-diagonal structure of the transition matrix. Almost all non-zero transition probabilities are equal to unity. The only exception is the transition from the first state, which accounts for the probability distribution (6.4).

Any sequence of pulses emitted by a radar can be represented by a concatenation of appropriate pulse interval Markov chains (6.7). In this way, one can model radar words of arbitrary complexity, including the ones shown in Fig. 6.1, and described in Appendices A and B. The transition probability matrix $\mathbf{A}_k$ of a Markov chain for the the k^{th} radar word has the following form

$$\mathbf{A}_k = \begin{pmatrix} \mathbf{A}'_{k,1} & \mathbf{A}''_{k,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{A}'_{k,2} & \mathbf{A}''_{k,2} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}'_{k,3} & \mathbf{A}''_{k,3} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}'_{k,4} & \mathbf{A}''_{k,4} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{6.8}$$

where

$$\mathbf{A}''_{k,i} = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline 1 & \mathbf{0} \end{array}\right).$$

As we have discussed in Section 6.1, the radar pulse sequences are observed in the stochastic environment modeled by the binary channel. Thus, a pulse-level model defined by (6.8) may be viewed as a Markov chain indirectly observed via the stochastic process that characterizes the binary channel. In

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4, we have identified such a case as a case of *hidden Markov modeling*. Therefore, combining the Markov chain (6.8) with the model of the stochastic binary EW channel, it is easy to develop a Hidden Markov Model (HMM) of a radar word.

Recalling that the probability of missing a pulse is p_{miss}, the probability of observing a spurious pulse is

$$p_{\text{spur}} \equiv \rho T_{\text{obs}}\,, \tag{6.9}$$

and following the notation of Chapter 4, the pulse-level HMM is defined as

$$\lambda_k = (\mathbf{A}_k, \mathbf{B}_k, \pi_k)\,, \tag{6.10}$$

where $\mathbf{A}_k$ is defined by (6.8), $\mathbf{B}_k$ denotes the observation probability matrix,

$$\mathbf{B}_k = \begin{pmatrix} 1-p_{\text{spur}} & p_{\text{spur}} \\ 1-p_{\text{spur}} & p_{\text{spur}} \\ p_{\text{miss}} & 1-p_{\text{miss}} \\ 1-p_{\text{spur}} & p_{\text{spur}} \\ \vdots & \vdots \end{pmatrix}, \tag{6.11}$$

and π_k denotes the initial state probability distribution, $\pi_k \equiv (1\,0\,0\,0\,\cdots)^T$.

It is worth pointing out that the size of the HMM model (6.10) for real-life radar word templates is very large. An even bigger disadvantage of this modeling approach is the dependency of the number of states of the model on the quantization resolution dictated by the clock period T_{obs}. The shorter the clock period, the greater is $l_{i,i+1}$ in (6.6), the greater is the size of the matrix (6.8). However, the structure of the matrix (6.8) offers significant computational relief to the radar signal processing algorithms that would utilize this pulse-level model.

The matrix (6.8) is extremely sparse. Also, the resulting pulse-level HMM belongs to the category of left-to-right HMMs that were introduced in Chapter 4. In Chapter 9, exploiting the sparsity of the transition probability matrix and the left-to-right nature of the model, we develop a very efficient pulse train analysis algorithm that scales almost linearly with respect to the size of the pulse-level model.

In addition, Appendix C presents an alternative pulse train analysis approach that completely eliminates the dependency of the size of the pulse-level model on the quantization resolution. Although this alternative approach is not as flexible as the one presented in Chapter 9, it, nevertheless, is applicable to modeling of a large class of Multi-Function Radars.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3 Summary

In this chapter, we have developed a modeling approach for the pulse-level component of the hierarchical Multi-Function Radar model introduced in Chapter 2. These pulse-level models may be used in the pulse train analysis application to extract radar words from the noisy and corrupted sequences of radar signal observations. These words are utilized by the word-level model described in Chapter 5 to capture the high-level behavioral aspects of the radars.

We have introduced the problem of pulse train analysis and discussed important modeling parameters such as signal distortion probability distributions dictated by this problem. We would like to emphasize a strong parallel between the problem of pulse train analysis, and that of the gene sequencing (Baldi and Brunak 2001; Durbin, Eddy, Krogh, and Mitchison 1998).

In both problems, sequential patterns have to be extracted from noisy observations. In fact, the only significant difference between the two problems is the nature of the data they are dealing with. In gene sequencing, the observable entities are molecules that build up genetic material. In the pulse train analysis, the observable data is the sequence of radar pulses.

As the literature indicates (Baldi and Brunak 2001; Durbin, Eddy, Krogh, and Mitchison 1998), the very same modeling techniques that we presented in this chapter have been successfully applied to the problem of gene sequencing. The field of bioinformatics has accumulated a significant amount of knowledge related to dealing with the problems of pattern extraction. As we have shown in this chapter, if thoroughly examined, this knowledge could be re-applied to the problems of radar signal processing.

Finally, as mentioned earlier, the major focus of this thesis is on word-level modeling. Therefore, we treat the pulse-level modeling as a problem supporting the needs of word-level modeling. However, this is a very interesting and challenging problem on its own. One of the simplifying assumptions that we made in this chapter was that pulse splitting and pulse jitter are the same concepts and result from quantization errors of the receiver. In general, this is not always the case. When rigorously treated, pulse jitter should be viewed as a stochastic process with a certain probability distribution. The interested reader is referred to the work of Dilkes (2004) and Dilkes (2005) that treats statistical distributions of pulse jitter in a more rigorous fashion.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 7

Model-centric Electronic Threat Libraries

In Chapters 5 and 6, we developed syntactic modeling methods for the word- and pulse-levels of the hierarchical Multi-Function Radar (MFR) model proposed in Chapter 2. In this chapter, we would like to discuss some practical aspects of these modeling approaches and show how these modeling techniques fit into the novel model-centric paradigm proposed in Chapter 1 and developed in this thesis.

This chapter is organized as follows. In Section 7.1, we discuss some important grammatical derivation constraints that have to be followed by the analyst while deriving word-level syntactic models for Multi-Function Radars. Following the guidelines suggested in this section ensures that the critical Non-Self-Embedding property of the resulting radar grammar is not violated.

In Section 7.2, we propose and describe in detail a design cycle for model-centric Electronic Threat Libraries based on the modeling approaches presented in Chapters 5 and 6. We describe a semiautomatic procedure that uses data-centric Electronic Intelligence records containing Multi-Function Radar parameters and specifications and, with the participation of an Electronic Warfare (EW) analyst, generates model-centric library entries. The role of the analyst is limited to deriving the grammars for the word- and pulse-levels, and supplying the synthesis procedure with some stochastic parameters. The rest of this procedure is completely automated thus minimizing costly human involvement.

Finally, Section 7.4 summarizes the material presented in this chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

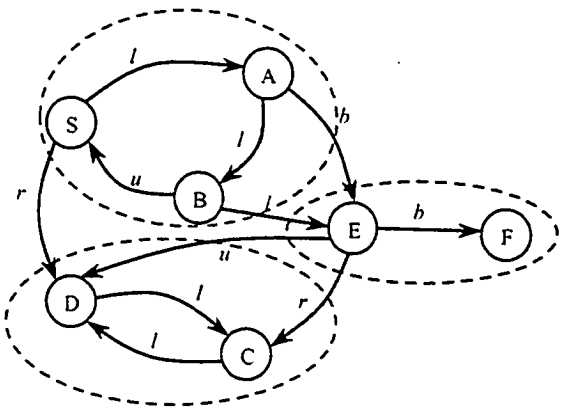

Figure 7.1: Cyclic and acyclic components of the production graph of the grammar (5.2). Red dashed circles highlight the cyclic components of this graph, while blue circle marks the acyclic component.

7.1 Derivation constraints for Multi-Function Radar grammars

In the word-level model synthesis procedure presented in Chapter 5, we have assumed that the behaviour of the complex radar signal is described by a grammar that is a context-free and Non-Self-Embedding (NSE). We have not yet addressed the issue of how to derive this grammar.

In Part II of this thesis, we illustrate the process of translation of the prior information from intelligence databases into Context-Free Grammars that describes Multi-Function Radars. This could be one of the functions of the Electronic Warfare (EW) analyst since processing and analyzing intelligence data cannot yet be completely automated. However, ensuring that the Non-Self-Embedding property of the MFR grammar is maintained requires attention since this property is the key to the application of the Finite-State Automata (FSA) synthesis procedure described in Chapter 5.

As we have discussed in Chapter 5, radar systems, as any digital finite-memory systems, are expected to behave in a causal or Markovian fashion. As a consequence, the language associated with MFRs is expected to be a Finite-State Language (FSL). Therefore, as proven by Anselmo, Giammarresi, and Varricchio (2003), Non-Self-Embedding Context-Free Grammars describing MFR languages can be found. However, if an EW analyst writes productions of the radar grammar in an arbitrary fashion, the Non-Self-Embedding property of the radar grammar may be violated. Consequently, the Finite-State Automata synthesis procedure described in Chapter 5 will not be applicable. In fact, by adhering to a set of simple constraints on the structure of the production rules, the EW analyst can avoid the problem of violating the Non-Self-Embedding property of the resulting MFR grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fig. 7.1 illustrates the production graph corresponding to example (5.2) analyzed in Chapter 5. As we can see from this figure, a production graph of a Context-Free Grammar contains *cyclic* and *acyclic* components. For example, $\{S, A, B\}$ forms a cycle, as does $\{C, D\}$. $\{E, F\}$, on the other hand, involves no cycle. When applied to the case of Multi-Function Radar grammars, cyclic components of the grammar will describe the radar state evolution dynamics, and acyclic components will describe the structure of radar words and phrases generated within each radar state (see Fig. 8.6 and Fig. 8.15).

While deriving cyclic components of the MFR grammar, the analyst has a choice of specifying productions in either right-linear, or left-linear form. A direct corollary of the Non-Self-Embedding property verification algorithm described in Chapter 5 entails that if every cyclic component in the production graph is in either left-linear or right-linear form, the resulting grammar will be Non-Self-Embedding. Therefore, when deriving MFR grammars, one form should be chosen and maintained throughout the derivation of all cyclic components. This will ensure that the resulting grammar passes the Non-Self-Embedding test.

7.2 Design cycle for the model-centric Electronic Threat Libraries

We now would like to discuss how the modeling techniques presented earlier in this part of the thesis fit into a paradigm of model-centric Electronic Intelligence (ELINT). As stated in Chapter 1, one of the most important goals of our study is to develop efficient forms of MFR representation in the model-centric Electronic Threat Libraries (ETL). Fig. 7.2 illustrates the proposed procedure for semi-automatic synthesis of MFR models that can be used to create entries for such ETL.

Fig. 7.2 represents the result of integration of four independent groups of components and actions:

1. Syntactic modeling.
2. Synthesis of the word-level radar model.
3. Synthesis of the pulse-level radar model.
4. Model-centric Electronic Threat Libraries and their applications.

We will now consider each of these groups in more detail.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

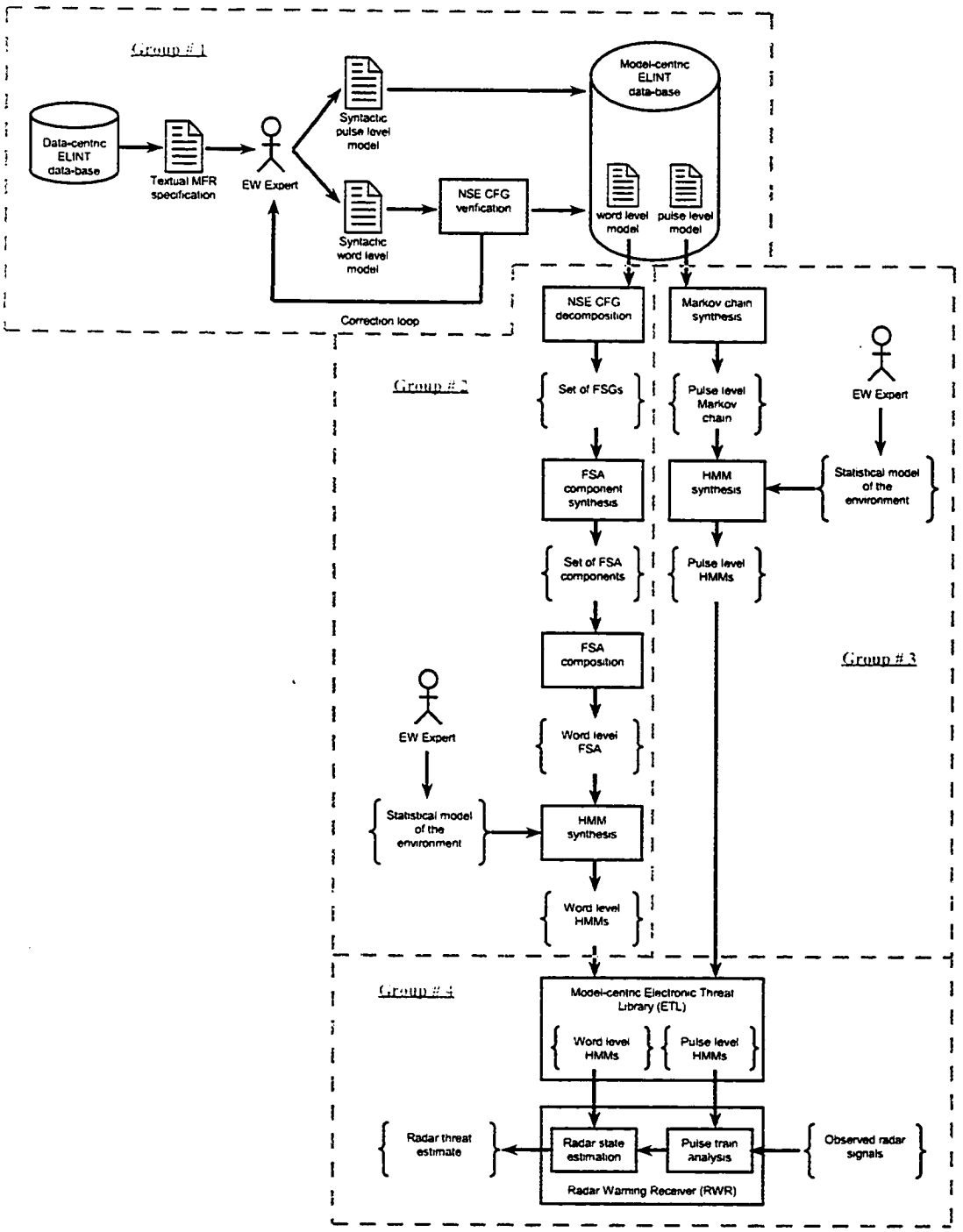

Figure 7.2: Design cycle for the model-centric Electronic Threat Libraries.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Syntactic modeling

Syntactic modeling is marked in Fig. 7.2 as *"Group # 1"*. The goal at this stage is to translate data-centric Electronic Intelligence records into a model-centric representation.

Due to the non-homogeneous nature of the data-centric Electronic Intelligence, participation of an analyst is critical at this stage. The analyst may receive a textual description of the Multi-Function Radar functionality similar to those given in Appendices A and B. The task of the analyst is to derive syntactic pulse- and word-level models based on this textual description.

The pulse-level model of the radar consists of a set of regular expressions that describe all radar words. The word-level model formulation involves derivation of a Context-Free Grammar following the guidelines described in Section 7.1. This is the only heuristic step in the procedure, and we will discuss it in more detail in Chapter 8.

Since the Non-Self-Embedding property of the word-level Context-Free Grammar is so critical, the grammar produced by the analyst should be validated by the verification algorithm described in Section 5.1. This should detect possible problems in the model and allow for the opportunity to correct the problems through use of the feedback provided by the verification step.

Finally, the word- and the pulse-level models of the radar can be stored in the model-centric Electronic Intelligence data base. The power of model-centricity is manifested through the homogeneous nature of the data records in this data base. The format of records that represent every Multi-Function Radar in such a data base is exactly the same. This makes it possible to automate the process of synthesis of elements of model-centric Electronic Threat Librariess.

Synthesis of the word-level radar model

The word-level model synthesis is marked in Fig. 7.2 as *"Group # 2"*. The procedure follows the same steps as described in Chapter 5. First, the syntactic word-level model has to be decomposed into a set of Finite-State Grammars as described in Section 5.2. Then, Finite-State Automata for all individual Finite-State Grammars obtained at the previous step are synthesized (Section 5.3). Then, the complete Finite-State Automaton of the radar is created following the composition procedure introduced in Section 5.4. Finally, with the input from the analyst, the model of uncertainty in the operation environment is used to convert the Finite-State Automaton of the radar into a Hidden Markov Model (see Chapter 4). This step is emitter-dependent and is discussed in more detail in Chapter 8.

The result of this synthesis procedure is a statistical word-level model of the radar based on a Hidden Markov Model. It may now be included into

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

model-centric Electronic Threat Libraries.

Synthesis of the pulse-level radar model

The pulse-level model synthesis is marked in Fig. 7.2 as *"Group # 3"*. The procedure follows modeling steps presented in Chapter 6. First, regular expressions that make up the syntactic pulse-level model are converted into the pulse-level Markov chains. Then, the binary channel model with parameters specified by the analyst is used to extend the pulse-level Markov chain into a pulse-level Hidden Markov Model. This Hidden Markov Model can now be included into model-centric Electronic Threat Libraries.

Model-centric ETL and its applications

"Group # 4" in Fig. 7.2 illustrates a potential application of the model-centric ETL. Each Multi-Function Radar is represented in this ETL by a pair of Hidden Markov Models that model word- and pulse-level aspects of the radar. An important client for such a library is a Radar Warning Receiver (RWR) of an aircraft or a ship.

Observed radar signals are fed through a pulse train analysis block (see Chapters 6 and 9 as well as Appendix C) which extracts radar word sequences using pulse-level HMMs. Then, word-level HMMs can be applied to the problem of radar state estimation and threat assessment utilizing the sequences of extracted radar words.

The elements of such model-centric Electronic Threat Libraries can also be used for other applications such as radar classification and predictive jamming.

7.3 Model accuracy considerations

From the stand point of theory of system analysis the detailed design process illustrated in Fig. 7.2 can be viewed at high level as a structured process of formalizing radar system requirements outlined in the Electronic Intelligence (ELINT) report. This process is illustrated in Fig. 7.3.

System analysis assumes four levels of formalization in the process of system design:

1. system requirements,
2. system specification,
3. system design, and
4. system implementation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

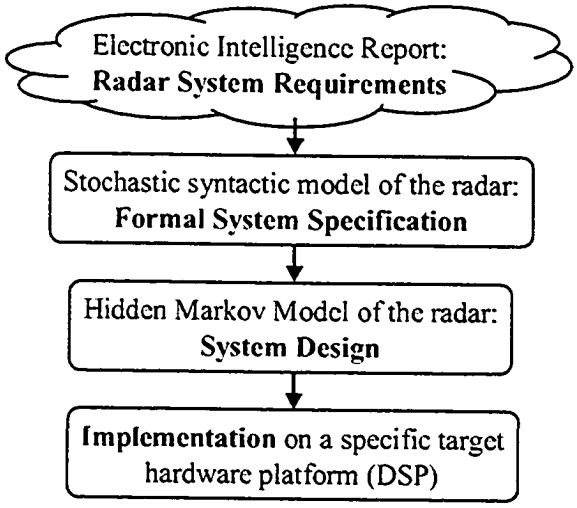

Figure 7.3: System analysis considerations. The Electronic Intelligence report could be viewed as a set of radar system requirements. Stochastic Context-Free Grammar of the radar constitutes a formal mathematical specification from which a precise system design in the form of a Hidden Markov Model can be synthesized. This model can then be used in a specific radar signal processing system implementation on a particular target platform.

At the high level, an informal, customer-specific set of *requirements* is collected. In case of MFR modeling, these requirements are available to the designers in the form of data-centric ELINT database records and intelligence reports.

The next step a system designer is required to undertake involves translation of informal (often simply textual) requirements into a formal *system specification*. This involves translating requirements into rules defined by a certain mathematical formalism. In our case, Stochastic Context-Free Grammars play a role of such formal system specifications.

The next step involves translation of the formal specification of the system into a solid *system design* (Hidden Markov Models in the case of MFRs) that could subsequently be *implemented* on a particular hardware platform.

With the accuracy of the MFR models in mind, one may pose the following legitimate question – at which point in the design process illustrated in Fig. 7.3 are approximations or simplifying assumptions made, and how can these sources of approximation affect the quality of model synthesis? The presence of these approximations creates an information gap between the model and the real radar system. The unavoidable presence of this gap complicates the tasks associated with radar signal processing such as radar emitter recognition and radar state estimation. In this section, we will address this issue from the conceptual standpoint.

System analysis tells us that approximations in the model design occur

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

at the transitions between the blocks in Fig. 7.3 when one form of system description is translated into another, more formal one. Therefore, there are four stages in the design process that potentially result in approximations:

1. formulation of system requirements,
2. translation of system requirements into a formal specification,
3. translation of the specification into a formal system design, and
4. synthesis of system implementation.

In case of finite-state models, efficient real-time code generation algorithms are available for direct synthesis of Hidden Markov Model implementations (Rabiner and Juang 1993). Therefore, no approximations or simplifications of the system's design occurs at the implementation stage of the diagram in Fig. 7.3.

The Hidden Markov Model synthesis procedure presented in Chapter 5 and summarized in Section 7.2 is also an exact procedure and involves no loss of information from the system specification level. Therefore, accuracy of the MFR model is maintained at this stage of the design process.

The approximations and simplifying design assumptions come into play at the top two stages of the procedure in Fig. 7.3. The first issue is the accuracy and completeness of the radar system requirements. The second issue is related to simplifying assumptions made at the stage of formalizing the requirements into a set of radar system specifications in the form of a Context-Free Grammar. Here, we will discuss both of these issues.

The first issue is the deficiency of intelligence reports resulting in the incomplete or inconsistent set of system requirements. In that case, the stochastic nature of the syntactic radar models can be exploited to absorb some uncertainties in the intelligence data (Fu 1974; Fu 1982). However, there is no substitute for the accurate system requirements. Therefore, Electronic Intelligence community continues to modernize and improve the methods of intelligence gathering and report generation.

The second issue involves approximating assumptions made at the stage of formalizing the requirements into a set of syntactic rules that constitute a formal radar system specification. Naturally, one set of requirements can be mapped into many sets of specifications that support different degrees of accuracy and compactness of the models.

One of the major assumptions that we are going to make (see Chapter 8) is that syntactic models of interest to us should support a high degree of computational efficiency of radar signal processing applications such as radar emitter recognition and radar state estimation. This typically yields syntactic models that describe supersets of original languages of the radars (Fu 1982;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Hopcroft and Ullman 1979; Nederhof 2000). In other words, all sequences of words generated by the radar will be in the underlying language described by the syntactic model. However, some of the strings that can be generated by this syntactic model will not represent valid radar signals.

The immediate implication of this design consideration is that the syntactic modeling methodology presented in this thesis cannot directly be applied to radar simulation and signal generation. In Chapter 10, we will discuss the issue of radar simulation once again and will provide some recommendations as to how to modify this syntactic modeling approach to accommodate for the task of radar system simulation.

7.4 Summary

This chapter addressed some practical aspects of the syntactic modeling methodology proposed in the thesis. In Chapter 5, we have demonstrated that syntactic models based on Non-Self-Embedding Context-Free Grammars offer a compact and powerful modeling framework for Multi-Function Radars. In this chapter, we have discussed some guidelines that have to be followed during the heuristic phase of radar grammar derivation in order to ensure the Non-Self-Embedding property of the resulting radar grammar.

We have also put the material presented in Chapters 5 and 6 into a global perspective by describing a complete design cycle for the model-centric Electronic Threat Libraries. This design cycle is almost entirely automated. The input of the analyst is required at three different points in the cycle (see Fig. 7.2). This includes translation of existing data-centric Electronic Intelligence records into syntactic radar models, and specification of statistical parameters of the environment in which the radars may be encountered.

Finally, we have discussed some issues concerning approximations and simplifying assumptions that affected the model synthesis procedure. These approximations result in a potential information gap, the unavoidable presence of which complicates the tasks of radar signal processing. We have looked at the general radar model design process and identified at which points in this process these approximations are most likely to occur.

Part III presents some practical modeling experiments and discusses some useful applications of the modeling methodology presented in the previous chapters.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Part III

Experiments and Applications

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 8

Word-level modeling experiments

In this chapter, we consider two case studies of word-level radar modeling. We analyze two real-life Anti-aircraft Defence radars described in Appendices A and B, derive word-level grammars for these emitters, consider statistical parameters for their syntactic models, and describe the steps involved in the Finite-State Automata synthesis procedure presented in Chapter 5 as applied to grammars of these emitters.

The rest of this chapter is organized as follows. In Section 8.1 we present a case study for the radar named *"Mercury"* described in Appendix A. Section 8.1.1 deals with derivation of deterministic, characteristic, and stochastic grammars for this radar. Section 8.1.2 describes steps of the Finite-State Automata synthesis algorithm applied to the grammar of this radar.

In Section 8.2, we repeat the same analysis for the *"Pluto"* emitter described in Appendix B. The grammars for this radar are derived in Section 8.2.1, and the state machine is synthesized in Section 8.2.2.

Finally, Section 8.3 presents some final remarks related to the experiments described in this chapter.

8.1 Case study: Mercury emitter

8.1.1 Mercury phrase structure grammar

In this section, we develop deterministic, characteristic, and stochastic phrase structure grammars of the Mercury emitter described in Appendix A. The grammar is derived as a word-level syntactic model of the emitter. We consider how the dynamics of radar words that make up one of the individual vertical slots in Fig. A.2 captures internal processes occurring within the radar emitter.

Table 8.1 provides an exhaustive list of all possible Mercury phrases, and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

associates them with the functional states of the radar. This table was obtained from specifications in Appendix A and is central to grammatical derivations that follow.

Deterministic grammar

According to Def. 3.2.1, 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:

$$\mathcal{A} = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9\} \tag{8.1}$$

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

To determine the set of nonterminals and the structure of the production rules of the Mercury radar grammar, we must first analyze its functionality at a global level. 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, Non-Adaptive Track, three stages of Range Resolution, and Track Maintenance. The transitions between these functional states can be captured by the state machine illustrated in Fig. 8.1.

If the radar is in Search, it can remain in this functional state (shown by the self-loop of the Search state in Fig. 8.1), or move to the Acquisition state if the target is detected. The target acquisition cycle involves transitions from Acquisition state to Non-Adaptive Track, to Range Resolutions states, and finally to Track Maintenance. The radar can remain in any of these states an unspecified length of time. Finally, the target acquisition or track can be abandoned at any point and the radar can move back to Search.

At every functional state, the radar emits a phrase consisting of four words and drawn from the Table 8.1. These phrases form strings in the radar language that we are interested in modeling syntactically.

Based on the background material presented in Chapter 3, it is easy to formulate the following set of right-linear grammatical rules for the state machine of Fig. 8.1:

$$\begin{aligned}
<State> &\rightarrow <Search> | <ACQ> | <NAT> | <RR> | <TM> \\
<RR> &\rightarrow <RR_1> | <RR_2> | <RR_3> \\
<Search> &\rightarrow <SearchPhrase><Search> | <SearchPhrase><ACQ> \\
<ACQ> &\rightarrow <AcqPhrase><ACQ> | <AcqPhrase><NAT> | <AcqPhrase><Search> \\
<NAT> &\rightarrow <NATPhrase><NAT> | <NATPhrase><RR_1> | <NATPhrase><Search> \\
<RR_1> &\rightarrow <RR_1Phrase><RR_1> | <RR_1Phrase><RR_2> | <RR_1Phrase><Search> \\
<RR_2> &\rightarrow <RR_2Phrase><RR_2> | <RR_2Phrase><RR_3> | <RR_2Phrase><Search> \\
<RR_3> &\rightarrow <RR_3Phrase><RR_3> | <RR_3Phrase><TM> | <RR_3Phrase><Search> \\
<TM> &\rightarrow <TMPhrase><TM> | <TMPhrase><RR_1> | <TMPhrase><Search>
\end{aligned}$$

The grammatical rules above form the syntactic skeleton of the Mercury grammar. To complete the derivation of the grammar, we must define the rules

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Functional State	Phrase Content	Functional State	Phrase Content
Four-Word Search	$[w_1 w_2 w_4 w_5]$	Track Maintenance (TM)	$[w_1 w_7 w_7 w_7]$
	$[w_2 w_4 w_5 w_1]$		$[w_2 w_7 w_7 w_7]$
	$[w_4 w_5 w_1 w_2]$		$[w_3 w_7 w_7 w_7]$
	$[w_5 w_1 w_2 w_4]$		$[w_4 w_7 w_7 w_7]$
Three-Word Search	$[w_1 w_3 w_5 w_1]$		$[w_5 w_7 w_7 w_7]$
	$[w_3 w_5 w_1 w_3]$		$[w_6 w_7 w_7 w_7]$
	$[w_5 w_1 w_3 w_5]$		$[w_1 w_8 w_8 w_8]$
Acquisition	$[w_1 w_1 w_1 w_1]$		$[w_2 w_8 w_8 w_8]$
	$[w_2 w_2 w_2 w_2]$		$[w_3 w_8 w_8 w_8]$
	$[w_3 w_3 w_3 w_3]$		$[w_4 w_8 w_8 w_8]$
	$[w_4 w_4 w_4 w_4]$		$[w_5 w_8 w_8 w_8]$
	$[w_5 w_5 w_5 w_5]$		$[w_6 w_8 w_8 w_8]$
Non-Adaptive Track (NAT) or Track Maintenance (TM)	$[w_1 w_6 w_6 w_6]$		$[w_1 w_9 w_9 w_9]$
	$[w_2 w_6 w_6 w_6]$		$[w_2 w_9 w_9 w_9]$
	$[w_3 w_6 w_6 w_6]$		$[w_3 w_9 w_9 w_9]$
	$[w_4 w_6 w_6 w_6]$		$[w_4 w_9 w_9 w_9]$
	$[w_5 w_6 w_6 w_6]$		$[w_5 w_9 w_9 w_9]$
Range Resolution	$[w_7 w_6 w_6 w_6]$		$[w_6 w_9 w_9 w_9]$
	$[w_8 w_6 w_6 w_6]$		$[w_7 w_7 w_7 w_7]$
	$[w_9 w_6 w_6 w_6]$		$[w_8 w_8 w_8 w_8]$
Acq., NAT, or TM	$[w_6 w_6 w_6 w_6]$		$[w_9 w_9 w_9 w_9]$

Table 8.1: List of all Mercury emitter phrase combinations according to the functional state of the radar. Some phrases are unique and directly identify the functional state of the radar (i.e. $[w_1 w_2 w_4 w_5]$ can only me encountered during search operations). Other phrases are characteristic to several radar states (i.e. $[w_6 w_6 w_6 w_6]$ can be utilized in Acquisition, Non-Adaptive Track, and Track Maintenance).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

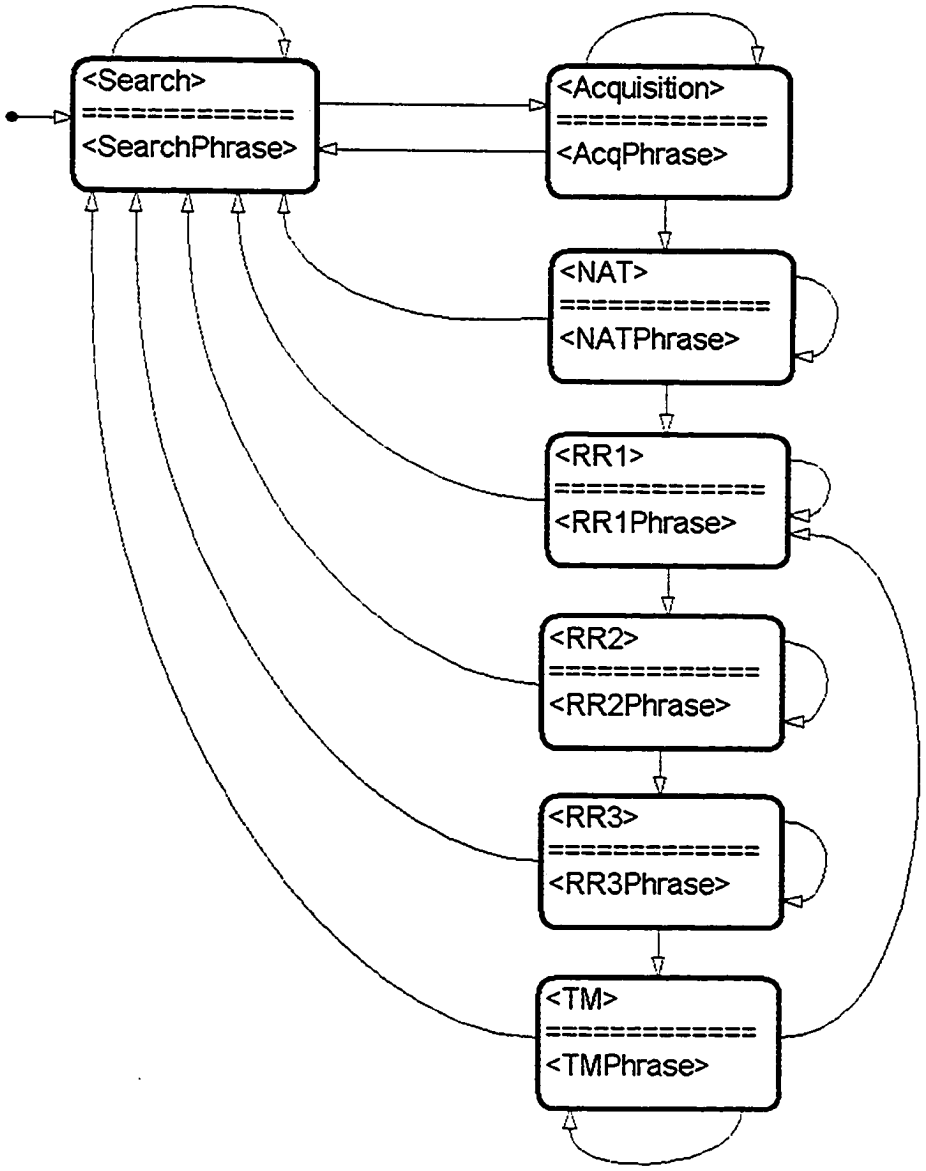

Figure 8.1: Mercury emitter functionality at the high level. There are seven functional states of the emitter (Search, Acquisition, Non-Adaptive Track, three stages of Range Resolution, and Track Maintenance). The transitions between the states are defined according to the specification of Appendix A. The state machine is shown as the Moor automaton with outputs defined by the states. A corresponding phrase from Table 8.1 is generated in every state of the automaton.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for the $< XPhrase >$ nonterminals where X stands for the corresponding name of the emitter state in which this phrase is emitted.

Using data from Table 8.1, 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:

$$Q_1 \rightarrow w_1w_1w_1w_1 \quad Q_4 \rightarrow w_4w_4w_4w_4 \quad Q_7 \rightarrow w_7w_7w_7w_7$$
$$Q_2 \rightarrow w_2w_2w_2w_2 \quad Q_5 \rightarrow w_5w_5w_5w_5 \quad Q_8 \rightarrow w_8w_8w_8w_8$$
$$Q_3 \rightarrow w_3w_3w_3w_3 \quad Q_6 \rightarrow w_6w_6w_6w_6 \quad Q_9 \rightarrow w_9w_9w_9w_9$$

The $< SearchPhrase >$ rules are:

$$\begin{aligned} < SearchPhrase > &\rightarrow < FourWSearch > | < ThreeWSearch > \\ < FourWSearch > &\rightarrow w_1w_2w_4w_5|w_2w_4w_5w_1|w_4w_5w_1w_2|w_5w_1w_2w_4 \\ < ThreeWSearch > &\rightarrow w_1w_3w_5w_1|w_3w_5w_1w_3|w_5w_1w_3w_5 \end{aligned}$$

The $< AcqPhrase >$ rules are:

$$< AcqPhrase > \rightarrow Q_1|Q_2|Q_3|Q_4|Q_5|Q_6$$

The $< NATPhrase >$ rules are:

$$\begin{aligned} < NATPhrase > &\rightarrow S_1T_6|Q_6 \\ S_1 &\rightarrow w_1|w_2|w_3|w_4|w_5 \end{aligned}$$

The Range Resolution rules are:

$$\begin{aligned} < RR_1Phrase > &\rightarrow w_7T_6 \\ < RR_2Phrase > &\rightarrow w_8T_6 \\ < RR_3Phrase > &\rightarrow w_9T_6 \end{aligned}$$

Finally, the Track Maintenance rules are:

$$\begin{aligned} < TMPhrase > &\rightarrow < FourWTrack > | < ThreeWTrack > \\ < FourWTrack > &\rightarrow Q_6|Q_7|Q_8|Q_9 \\ < ThreeWTrack > &\rightarrow S_1T_6|S_2T_7|S_2T_8|S_2T_9 \\ S_2 &\rightarrow S_1|w_6 \\ S_1 &\rightarrow w_1|w_2|w_3|w_4|w_5 \end{aligned}$$

The resulting deterministic phrase structure grammar for the Mercury emitter G is shown in Fig. 8.2. According to the Chomsky hierarchy discussed in Chapter 3, this grammar is a Context-Free Grammar (CFG).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\rightarrow < Search > | < ACQ > | < NAT > | < RR > | < TM > \\
< RR > &\rightarrow < RR_1 > | < RR_2 > | < RR_3 > \\
< Search > &\rightarrow < SearchPhrase >< Search > | < SearchPhrase >< ACQ > \\
< ACQ > &\rightarrow < AcqPhrase >< ACQ > | < AcqPhrase >< NAT > | \\
&\quad < AcqPhrase >< Search > \\
< NAT > &\rightarrow < NATPhrase >< NAT > | < NATPhrase >< RR_1 > | \\
&\quad < NATPhrase >< Search > \\
< RR_1 > &\rightarrow < RR_1Phrase >< RR_1 > | < RR_1Phrase >< RR_2 > | \\
&\quad < RR_1Phrase >< Search > \\
< RR_2 > &\rightarrow < RR_2Phrase >< RR_2 > | < RR_2Phrase >< RR_3 > | \\
&\quad < RR_2Phrase >< Search > \\
< RR_3 > &\rightarrow < RR_3Phrase >< RR_3 > | < RR_3Phrase >< TM > | \\
&\quad < RR_3Phrase >< Search > \\
< TM > &\rightarrow < TMPhrase >< TM > | < TMPhrase >< RR_1 > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\rightarrow < FourWSearch > | < ThreeWSearch > \\
< AcqPhrase > &\rightarrow Q_1|Q_2|Q_3|Q_4|Q_5|Q_6 \\
< NATPhrase > &\rightarrow S_1T_6|Q_6 \\
< RR_1Phrase > &\rightarrow w_7T_6 \\
< RR_2Phrase > &\rightarrow w_8T_6 \\
< RR_3Phrase > &\rightarrow w_9T_6 \\
< TMPhrase > &\rightarrow < FourWTrack > | < ThreeWTrack > \\
< FourWSearch > &\rightarrow w_1w_2w_4w_5|w_2w_4w_5w_1|w_4w_5w_1w_2|w_5w_1w_2w_4 \\
< ThreeWSearch > &\rightarrow w_1w_3w_5w_1|w_3w_5w_1w_3|w_5w_1w_3w_5 \\
< FourWTrack > &\rightarrow Q_6|Q_7|Q_8|Q_9 \\
< ThreeWTrack > &\rightarrow S_1T_6|S_2T_7|S_2T_8|S_2T_9 \\
S_2 &\rightarrow S_1|w_6 \\
S_1 &\rightarrow w_1|w_2|w_3|w_4|w_5 \\
T_6 &\rightarrow w_6w_6w_6 \\
T_7 &\rightarrow w_7w_7w_7 \\
T_8 &\rightarrow w_8w_8w_8 \\
T_9 &\rightarrow w_9w_9w_9 \\
Q_1 &\rightarrow w_1w_1w_1w_1 \\
Q_2 &\rightarrow w_2w_2w_2w_2 \\
Q_3 &\rightarrow w_3w_3w_3w_3 \\
Q_4 &\rightarrow w_4w_4w_4w_4 \\
Q_5 &\rightarrow w_5w_5w_5w_5 \\
Q_6 &\rightarrow w_6w_6w_6w_6 \\
Q_7 &\rightarrow w_7w_7w_7w_7 \\
Q_8 &\rightarrow w_8w_8w_8w_8 \\
Q_9 &\rightarrow w_9w_9w_9w_9
\end{aligned}
$$

Figure 8.2: Deterministic grammar of the Mercury emitter. The structure of the grammar suggests that it is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

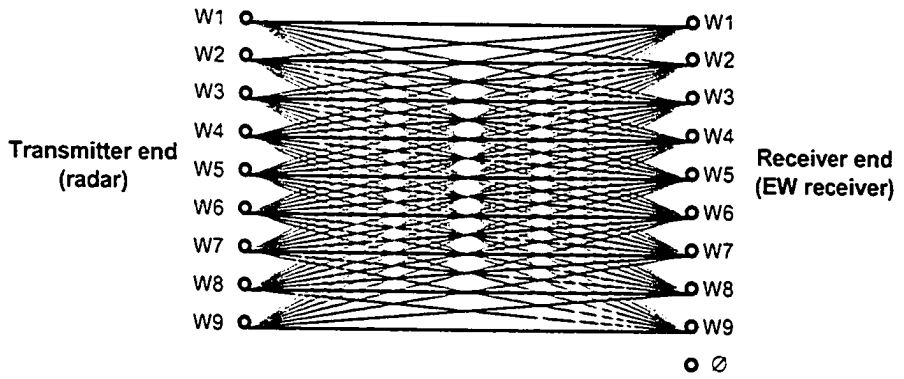

Figure 8.3: Stochastic channel impairment model. Due to signal propagation effects as well as de-interleaving and pulse train analysis errors, the words received by the EW receiver can differ from the ones that were actually transmitted. The most extreme case is when a certain word could not be identified by the pulse train analysis layer at all. In that case, the $\varnothing$ word is declared. A certain probability measure is associated with every edge of this graph. The color intensity of edges of this graph provides an example of a possible probability distribution (the darker the edge, the more likely the transition).

Characteristic grammar

The characteristic grammar of the Mercury emitter must extend the deterministic grammar of Fig. 8.2 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 as depicted in Fig. 8.3.

To accommodate for the channel impairment model of Fig. 8.3, we have to make two modifications to the deterministic grammar of the Mercury emitter. First of all, the alphabet (8.1) has to be expanded to include the $\varnothing$ character – the character indicating that no reliable radar signal detection was possible:

$$\mathcal{A}_c = \{\varnothing\} \cup \mathcal{A} = \{\varnothing, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9\}. \tag{8.2}$$

Finally, we introduce an additional level of indirection into the grammar of Fig. 8.2 by adding nine new nonterminals $W_1, \ldots, W_9$ and nine new production rules:

$$W_i \rightarrow \varnothing | w_1 | w_2 | w_3 | w_4 | w_5 | w_6 | w_7 | w_8 | w_9 , \tag{8.3}$$

where $i = 1, \ldots, 9$. The reason for this modification will become apparent in the next section where we associate probabilities with the production rules of the grammar. The resulting characteristic grammar of the Mercury emitter is depicted in Fig. 8.4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\rightarrow < Search > | < ACQ > | < NAT > | < RR > | < TM > \\
< RR > &\rightarrow < RR_1 > | < RR_2 > | < RR_3 > \\
< Search > &\rightarrow < SearchPhrase >< Search > | < SearchPhrase >< ACQ > \\
< ACQ > &\rightarrow < AcqPhrase >< ACQ > | < AcqPhrase >< NAT > | \\
&\quad < AcqPhrase >< Search > \\
< NAT > &\rightarrow < NATPhrase >< NAT > | < NATPhrase >< RR_1 > | \\
&\quad < NATPhrase >< Search > \\
< RR_1 > &\rightarrow < RR_1Phrase >< RR_1 > | < RR_1Phrase >< RR_2 > | \\
&\quad < RR_1Phrase >< Search > \\
< RR_2 > &\rightarrow < RR_2Phrase >< RR_2 > | < RR_2Phrase >< RR_3 > | \\
&\quad < RR_2Phrase >< Search > \\
< RR_3 > &\rightarrow < RR_3Phrase >< RR_3 > | < RR_3Phrase >< TM > | \\
&\quad < RR_3Phrase >< Search > \\
< TM > &\rightarrow < TMPhrase >< TM > | < TMPhrase >< RR_1 > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\rightarrow < FourWSearch > | < ThreeWSearch > \\
< AcqPhrase > &\rightarrow Q_1|Q_2|Q_3|Q_4|Q_5|Q_6 \\
< NATPhrase > &\rightarrow S_1T_6|Q_6 \\
< RR_1Phrase > &\rightarrow W_7T_6 \\
< RR_2Phrase > &\rightarrow W_8T_6 \\
< RR_3Phrase > &\rightarrow W_9T_6 \\
< TMPhrase > &\rightarrow < FourWTrack > | < ThreeWTrack > \\
< FourWSearch > &\rightarrow W_1W_2W_4W_5|W_2W_4W_5W_1|W_4W_5W_1W_2|W_5W_1W_2W_4 \\
< ThreeWSearch > &\rightarrow W_1W_3W_5W_1|W_3W_5W_1W_3|W_5W_1W_3W_5 \\
< FourWTrack > &\rightarrow Q_6|Q_7|Q_8|Q_9 \\
< ThreeWTrack > &\rightarrow S_1T_6|S_2T_7|S_2T_8|S_2T_9 \\
S_2 &\rightarrow S_1|W_6 \\
S_1 &\rightarrow W_1|W_2|W_3|W_4|W_5 \\
T_6 &\rightarrow W_6W_6W_6 \\
T_7 &\rightarrow W_7W_7W_7 \\
T_8 &\rightarrow W_8W_8W_8 \\
T_9 &\rightarrow W_9W_9W_9 \\
Q_i &\rightarrow W_iW_iW_iW_i \\
W_i &\rightarrow \varnothing|w_1|w_2|w_3|w_4|w_5|w_6|w_7|w_8|w_9 \\
&\quad i = 1,\ldots,9
\end{aligned}
$$

Figure 8.4: Characteristic grammar of the Mercury emitter. This grammar, like its deterministic counterpart, is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Stochastic grammar

The channel impairment model of the Fig. 8.3 has the following transition probability matrix:

$$\mathbf{P}_o = \left[\begin{array}{ccccccccc} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 & \mathbf{p}_5 & \mathbf{p}_6 & \mathbf{p}_7 & \mathbf{p}_8 & \mathbf{p}_9 \end{array} \right]^T =$$

$$\left(\begin{array}{ccccccccc} p_{1,\varnothing} & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} & p_{1,6} & p_{1,7} & p_{1,8} & p_{1,9} \\ p_{2,\varnothing} & p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} & p_{2,5} & p_{2,6} & p_{2,7} & p_{2,8} & p_{2,9} \\ p_{3,\varnothing} & p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} & p_{3,5} & p_{3,6} & p_{3,7} & p_{3,8} & p_{3,9} \\ p_{4,\varnothing} & p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} & p_{4,5} & p_{4,6} & p_{4,7} & p_{4,8} & p_{4,9} \\ p_{5,\varnothing} & p_{5,1} & p_{5,2} & p_{5,3} & p_{5,4} & p_{5,5} & p_{5,6} & p_{5,7} & p_{5,8} & p_{5,9} \\ p_{6,\varnothing} & p_{6,1} & p_{6,2} & p_{6,3} & p_{6,4} & p_{6,5} & p_{6,6} & p_{6,7} & p_{6,8} & p_{6,9} \\ p_{7,\varnothing} & p_{7,1} & p_{7,2} & p_{7,3} & p_{7,4} & p_{7,5} & p_{7,6} & p_{7,7} & p_{7,8} & p_{7,9} \\ p_{8,\varnothing} & p_{8,1} & p_{8,2} & p_{8,3} & p_{8,4} & p_{8,5} & p_{8,6} & p_{8,7} & p_{8,8} & p_{8,9} \\ p_{9,\varnothing} & p_{9,1} & p_{9,2} & p_{9,3} & p_{9,4} & p_{9,5} & p_{9,6} & p_{9,7} & p_{9,8} & p_{9,9} \end{array} \right), \tag{8.4}$$

where the rows represent the transmitted radar words (the left hand side of Fig. 8.3), and the columns represent the words inferred by the pulse train analysis layer of the EW receiver from the noisy and corrupted observations (the right hand side of Fig. 8.3).

The stochastic radar grammar can be obtained from the characteristic grammar of Fig. 8.4 by associating probability vectors of (8.4) with the corresponding productions of (8.3):

$$W_i \xrightarrow{\mathbf{p}_i} \varnothing | w_1 | w_2 | w_3 | w_4 | w_5 | w_6 | w_7 | w_8 | w_9 \,, \tag{8.5}$$

where $i = 1, \ldots, 9$. Thus, the complete stochastic grammar of the Mercury emitter is shown in Fig. 8.5.

Strictly speaking, this grammar should be called weighted grammar rather than stochastic. As shown by Fu (1982), a stochastic CFG must satisfy the limiting stochastic consistency criterion. However, Fu (1982) 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.

8.1.2 Mercury state machine synthesis

Non-Self-Embedding Context-Free Grammar test

As stated by Fu (1982), the analysis of stochastic and weighted grammars must be performed using their characteristic counterparts. However, since the characteristic grammar of Fig. 8.4 is so close to the original deterministic grammar of Fig. 8.2, we can perform the Non-Self-Embedding property test directly on the deterministic CFG.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\xrightarrow{1} < Search > | < ACQ > | < NAT > | < RR > | < TM > \\
< RR > &\xrightarrow{1} < RR_1 > | < RR_2 > | < RR_3 > \\
< Search > &\xrightarrow{1} < SearchPhrase >< Search > | < SearchPhrase >< ACQ > \\
< ACQ > &\xrightarrow{1} < AcqPhrase >< ACQ > | < AcqPhrase >< NAT > | \\
&\quad < AcqPhrase >< Search > \\
< NAT > &\xrightarrow{1} < NATPhrase >< NAT > | < NATPhrase >< RR_1 > | \\
&\quad < NATPhrase >< Search > \\
< RR_1 > &\xrightarrow{1} < RR_1Phrase >< RR_1 > | < RR_1Phrase >< RR_2 > | \\
&\quad < RR_1Phrase >< Search > \\
< RR_2 > &\xrightarrow{1} < RR_2Phrase >< RR_2 > | < RR_2Phrase >< RR_3 > | \\
&\quad < RR_2Phrase >< Search > \\
< RR_3 > &\xrightarrow{1} < RR_3Phrase >< RR_3 > | < RR_3Phrase >< TM > | \\
&\quad < RR_3Phrase >< Search > \\
< TM > &\xrightarrow{1} < TMPhrase >< TM > | < TMPhrase >< RR_1 > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\xrightarrow{1} < FourWSearch > | < ThreeWSearch > \\
< AcqPhrase > &\xrightarrow{1} Q_1|Q_2|Q_3|Q_4|Q_5|Q_6 \\
< NATPhrase > &\xrightarrow{1} S_1T_6|Q_6 \\
< RR_1Phrase > &\xrightarrow{1} W_7T_6 \\
< RR_2Phrase > &\xrightarrow{1} W_8T_6 \\
< RR_3Phrase > &\xrightarrow{1} W_9T_6 \\
< TMPhrase > &\xrightarrow{1} < FourWTrack > | < ThreeWTrack > \\
< FourWSearch > &\xrightarrow{1} W_1W_2W_4W_5|W_2W_4W_5W_1|W_4W_5W_1W_2|W_5W_1W_2W_4 \\
< ThreeWSearch > &\xrightarrow{1} W_1W_3W_5W_1|W_3W_5W_1W_3|W_5W_1W_3W_5 \\
< FourWTrack > &\xrightarrow{1} Q_6|Q_7|Q_8|Q_9 \\
< ThreeWTrack > &\xrightarrow{1} S_1T_6|S_2T_7|S_2T_8|S_2T_9 \\
S_2 &\xrightarrow{1} S_1|W_6 \\
S_1 &\xrightarrow{1} W_1|W_2|W_3|W_4|W_5 \\
T_6 &\xrightarrow{1} W_6W_6W_6 \\
T_7 &\xrightarrow{1} W_7W_7W_7 \\
T_8 &\xrightarrow{1} W_8W_8W_8 \\
T_9 &\xrightarrow{1} W_9W_9W_9 \\
Q_i &\xrightarrow{1} W_iW_iW_iW_i \\
W_i &\xrightarrow{p_i} \varnothing|w_1|w_2|w_3|w_4|w_5|w_6|w_7|w_8|w_9 \\
&\quad i = 1,\ldots,9
\end{aligned}
$$

Figure 8.5: Weighted grammar of the Mercury emitter. This grammar, like its deterministic counterpart, is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fig. 8.7 shows the transition matrix M of the underlying production graph (Fig. 8.6) of the CFG of Fig. 8.2. The matrix is very sparse and has only a few nonempty elements. The steady-state analysis matrix

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

is shown in Fig. 8.8, and the test yields the following result:

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

This confirms that the Mercury grammar is a Non-Self-Embedding Context-Free Grammar. Therefore, a Finite-State Automaton of the Mercury emitter can be synthesized from the grammar of Fig. 8.5.

Mercury state machine

Using the Dulmage-Mendelsohn decomposition of the transition matrix M of Fig. 8.7, as described in Chapter 5, we obtain 29 strongly-connected components of the production graph of the CFG of the Mercury emitter. They are listed in Table 8.2.

As shown in Chapter 5, each strongly connected component of the production graph corresponds to a Finite-State Grammar (FSG). Finite-State Automata (FSA) for each of these FSGs are shown in Fig. 8.9 and Fig. 8.10. The complete state machine of the Mercury emitter can be obtained by applying the FSA composition operation[1].

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. 8.5. 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 (8.4)) are brought into the structure of the radar model. This conversion procedure is illustrated in Chapter 4 in (4.2) – (4.5). 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 (4.3), and the probabilities of observations as demonstrated in (4.5).

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

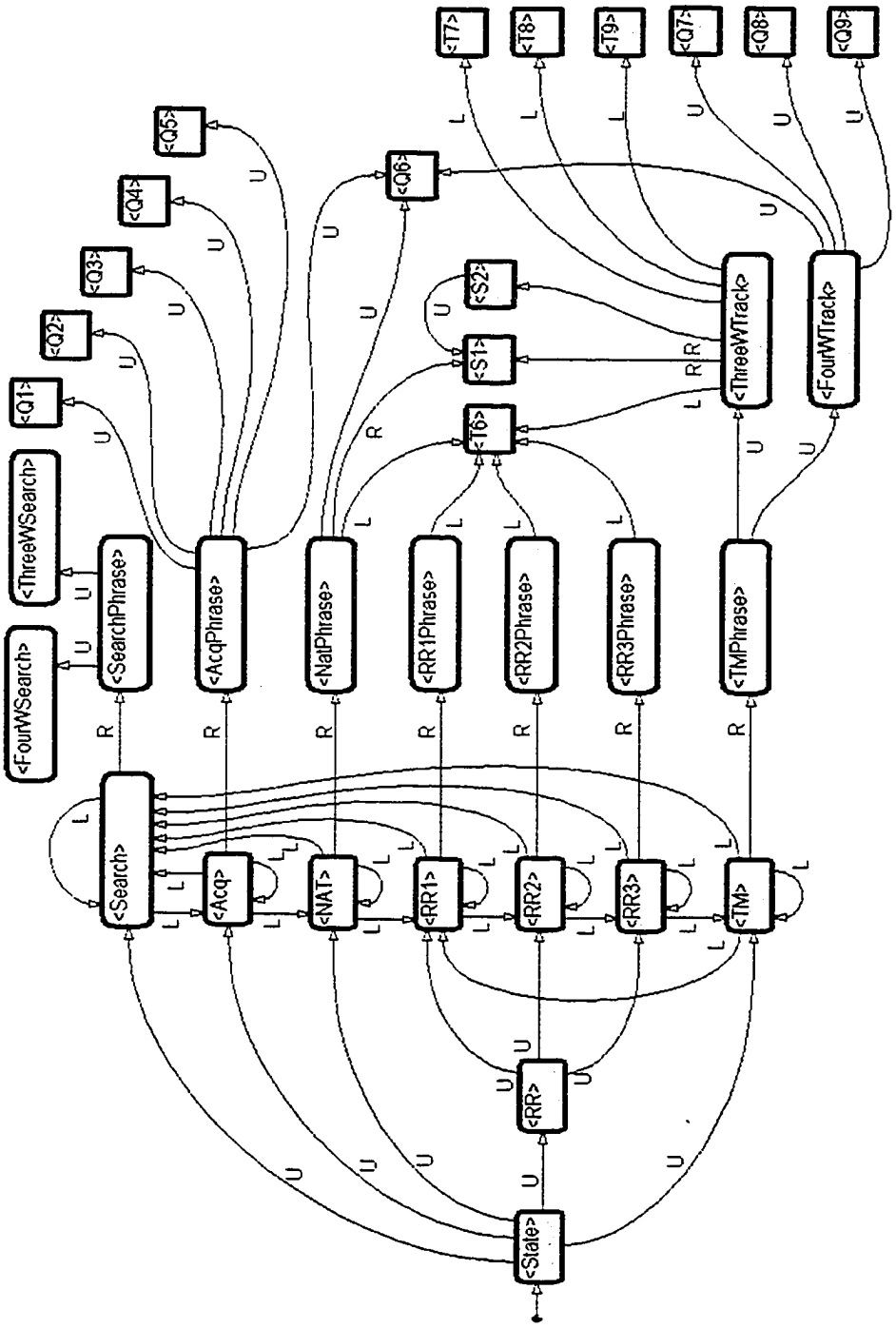

Figure 8.6: Production graph of the Mercury grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

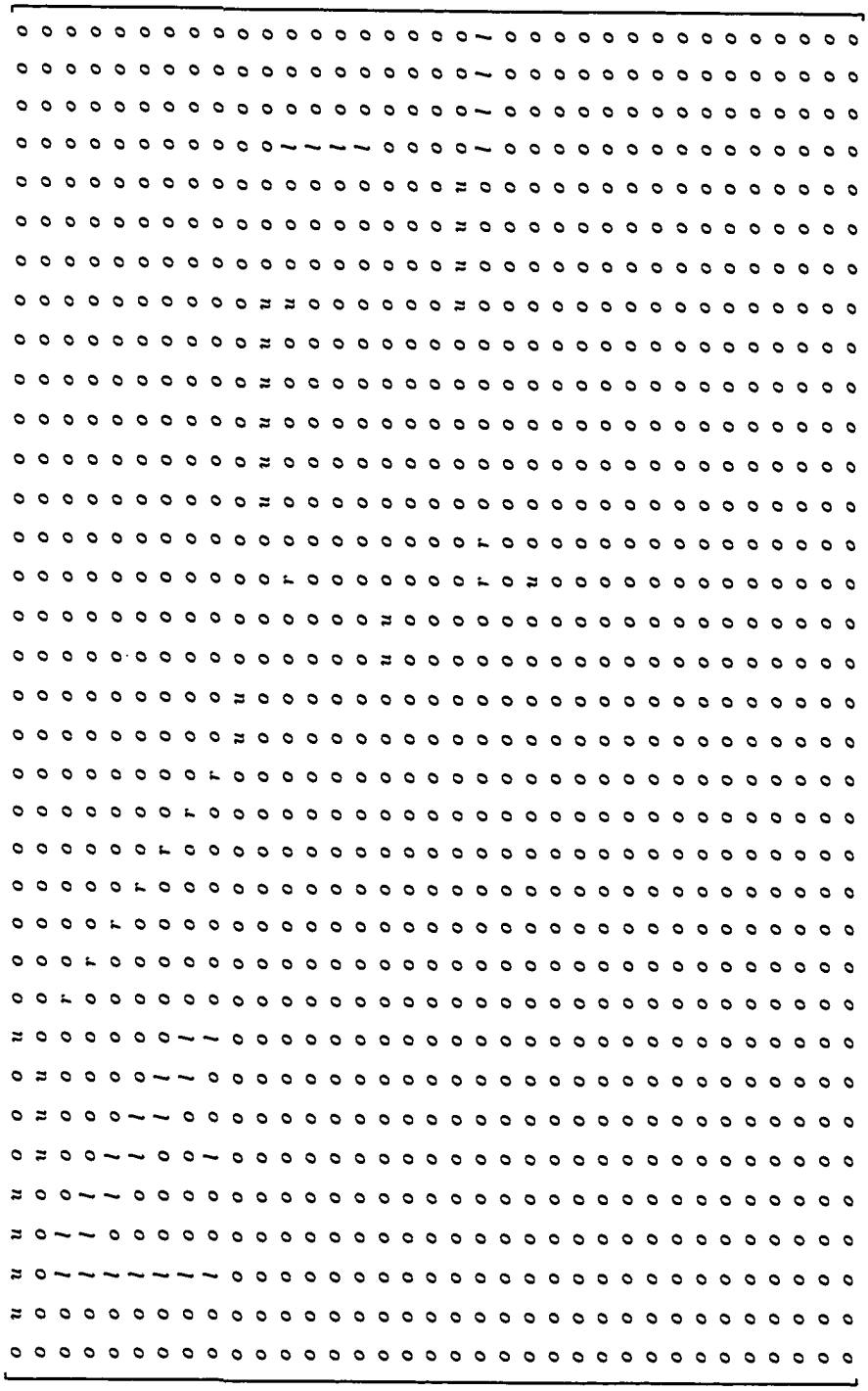

Figure 8.7: Transition matrix M(G) for the Mercury emitter grammar based on the grammar in Fig. 8.2.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Figure 8.8: Production graph analysis matrix $\mathbf{M}^{\leq N}(G) = \sum_{i=1}^{N} \mathbf{M}^{i}(G)$ for the Mercury emitter grammar. The diagonal elements of this matrix are $\mathbf{diag}(\mathbf{M}^{\leq N}(G)) = \left[\begin{array}{cccccccccc} o & o & l & l & l & l & l & l & o & \ldots & o \end{array}\right]^{T}$. This indicates that the Mercury grammar is a Non-Self-Embedding Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

8.2 Case study: Pluto emitter

8.2.1 Pluto phrase structure grammar

In this section, we develop deterministic, characteristic, and stochastic phrase structure grammars of the Pluto emitter described in Appendix B.

Before we proceed with grammatical derivation, we have to make the two following remarks:

1. As described in Appendix B, word w_2 is present in radar signals in two varieties differentiated by the distinguishable beam pattern. Therefore, we will consider these varieties of word w_2 as two distinct words – w_2 and w_n.

2. From Fig. B.1, we observe that the slot for the first word in the phrase is slightly longer than that of the second word. We can accommodate for this fact by introducing a special "blank" pad character w_p. Thus, the phrase structure of this radar can be defined as $[w_i w_p w_j w_t]$ where $i, j \in \{0, 1, 2, 3, 4, n\}$.

This effectively means that each phrase contains 4 words. Therefore, structural signal flow of the Pluto emitter is similar to the flow of the Mercury emitter shown in Fig. A.2.

Table 8.3 provides an exhaustive list of all possible Pluto phrases, and associates them with the functional states of the radar. This table is central to the grammatical derivations that follow.

Deterministic grammar

We define the alphabet of the Pluto emitter as the following set of radar words:

$$\mathcal{A} = \{w_0, w_1, w_2, w_3, w_4, w_n, w_p, w_t\} \tag{8.6}$$

High-level functionality of the Pluto emitter is illustrated in Fig. 8.11. Note that this radar is rather primitive compared to the Mercury emitter whose functionality is shown in Fig. 8.1. Applying the same steps as described in Section 8.1.1 for derivation of the Mercury emitter grammar, we derive the grammar of the Pluto emitter shown in Fig. 8.12.

As it is evident from Table 8.3, Pluto signals contain a high degree of ambiguity. For instance, in the Search state, this emitter employs phrases that are also used in almost all other states of the functional cycle. We have to accommodate for this ambiguity in the grammar by sharing many nonterminals between the phrase composition rules in Fig. 8.12.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

#	Nonterminals in the strongly connected component
1	$< State >$
2	$< RR >$
3	$< Search >$ $< Acq >$ $< NAT >$ $< RR_1 >$ $< RR_2 >$ $< RR_3 >$ $< TM >$
4	$< SearchPhrase >$
5	$< AcqPhrase >$
6	$< NATPhrase >$
7	$< RR_1Phrase >$
8	$< RR_2Phrase >$
9	$< RR_3Phrase >$
10	$< TMPhrase >$
11	$< FourWSearch >$
12	$< ThreeWSearch >$
13	$< FourWTrack >$
14	$< ThreeWTrack >$
15	$< S_1 >$
16	$< S_2 >$
17	$< Q_1 >$
18	$< Q_2 >$
19	$< Q_3 >$
20	$< Q_4 >$
21	$< Q_5 >$
22	$< Q_6 >$
23	$< Q_7 >$
24	$< Q_8 >$
25	$< Q_9 >$
26	$< T_6 >$
27	$< T_7 >$
28	$< T_8 >$
29	$< T_9 >$

Table 8.2: Strongly connected components of the Mercury production graph of Fig. 8.6.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

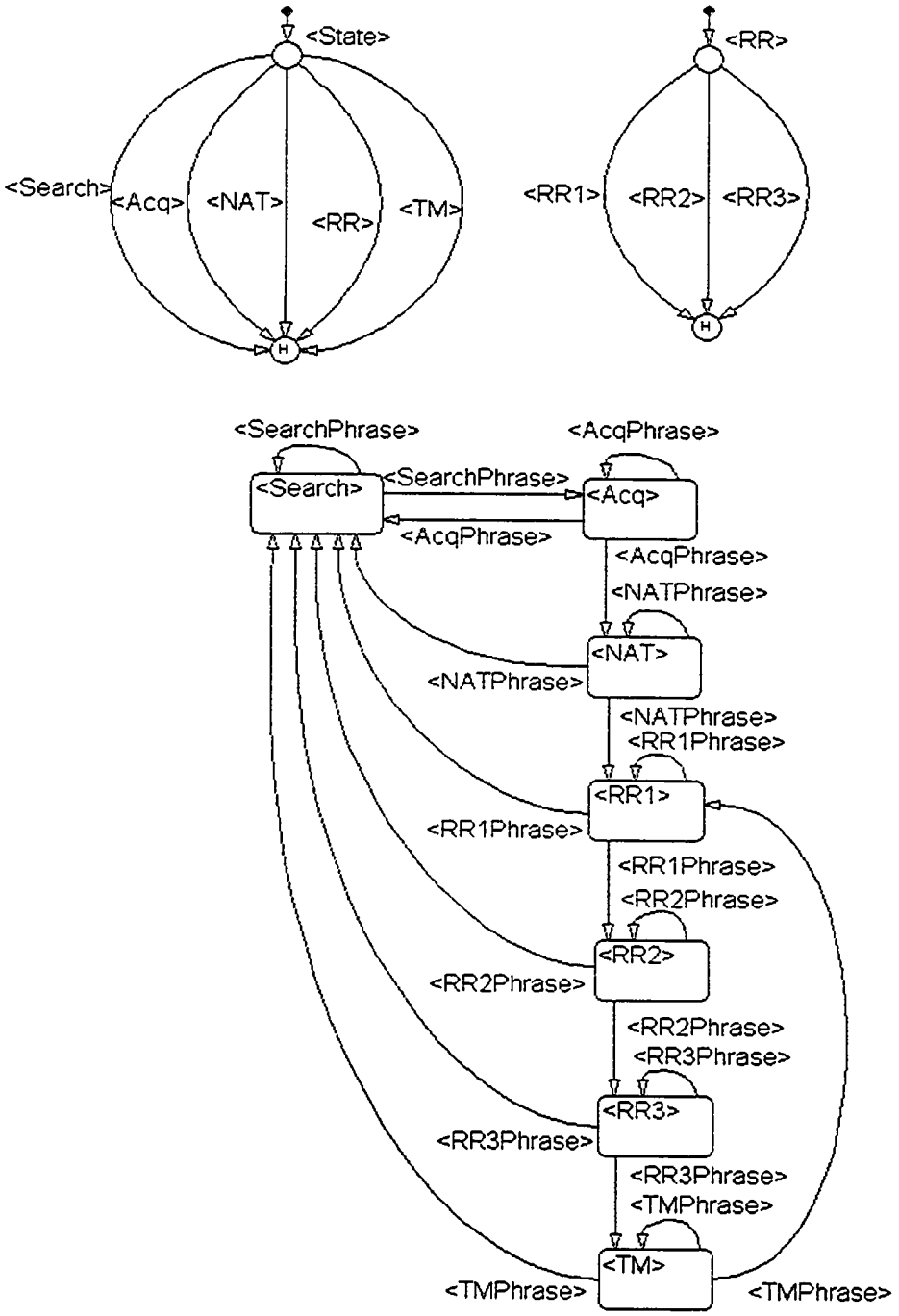

Figure 8.9: Mercury state machine components.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

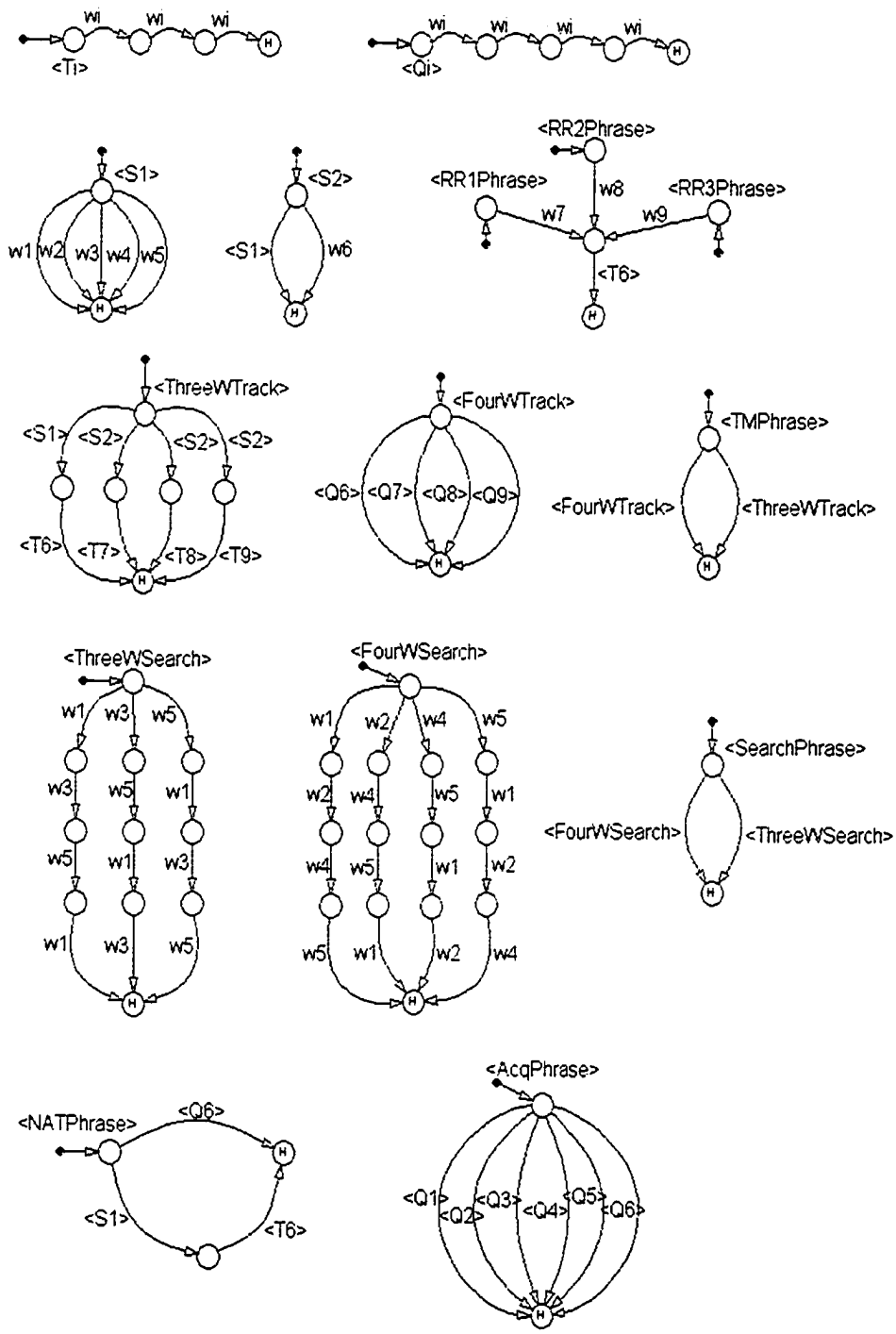

Figure 8.10: Mercury state machine components.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Functional State	Phrase Content	Functional State	Phrase Content
Search	$[w_0 w_p w_2 w_t]$	Track Maintenance (TM)	$[w_0 w_p w_1 w_t]$
	$[w_2 w_p w_2 w_t]$		$[w_0 w_p w_2 w_t]$
Non-Adaptive Track (NAT)	$[w_0 w_p w_2 w_t]$		$[w_0 w_p w_3 w_t]$
	$[w_2 w_p w_2 w_t]$		$[w_0 w_p w_4 w_t]$
	$[w_0 w_p w_n w_t]$		$[w_2 w_p w_1 w_t]$
	$[w_2 w_p w_n w_t]$		$[w_2 w_p w_2 w_t]$
Range Resolution	$[w_0 w_p w_1 w_t]$		$[w_2 w_p w_3 w_t]$
	$[w_0 w_p w_2 w_t]$		$[w_2 w_p w_4 w_t]$
	$[w_0 w_p w_3 w_t]$		
	$[w_0 w_p w_4 w_t]$		
	$[w_0 w_p w_n w_t]$		

Phrase Content	Functional States Phrase Using This Phrase
$[w_0 w_p w_2 w_t]$	Search, NAT, RR, TM
$[w_2 w_p w_2 w_t]$	Search, NAT, TM
$[w_0 w_p w_n w_t]$	NAT, RR, TM
$[w_2 w_p w_n w_t]$	NAT
$[w_0 w_p w_1 w_t]$	RR, TM
$[w_0 w_p w_3 w_t]$	
$[w_0 w_p w_4 w_t]$	
$[w_2 w_p w_1 w_t]$	TM
$[w_2 w_p w_3 w_t]$	
$[w_2 w_p w_4 w_t]$	

Table 8.3: List of all Pluto emitter phrase combinations according to the functional state of the radar. Pluto has only four distinct functional states – Search, Non-Adaptive Track, Range Resolution, and Track Maintenance. The first table lists phrases that are generated by the emitter in each functional state. The second table defines the one-to-many relationship of each phrase and the functional states of the radar in which these phrase can be encountered.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

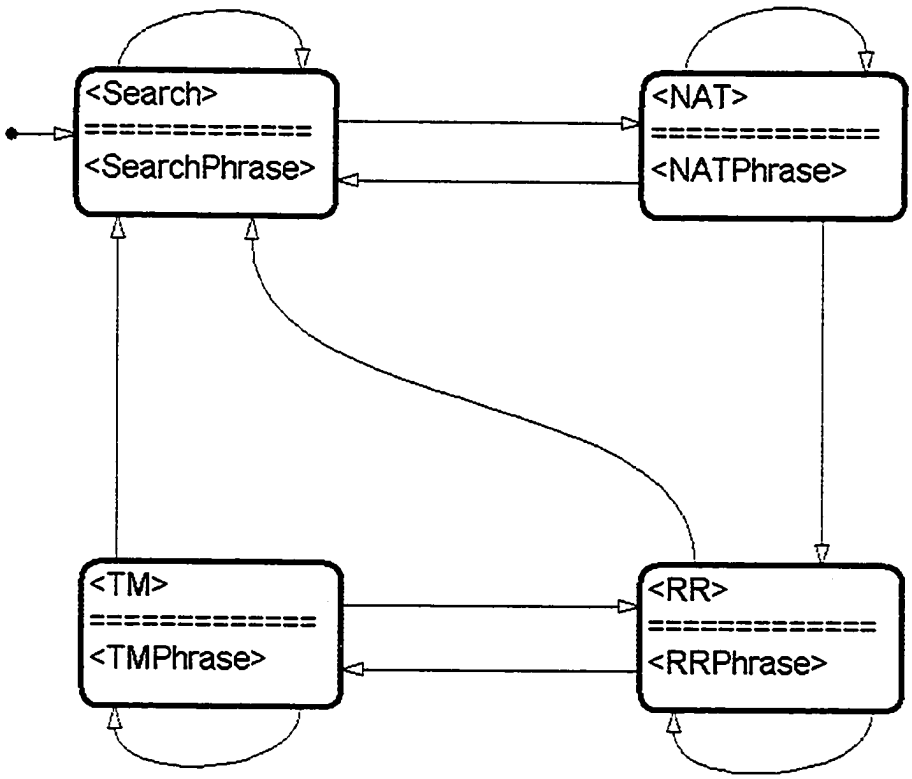

Figure 8.11: Pluto emitter functionality at the high level. There are four functional states of the emitter (Search, Non-Adaptive Track, Range Resolution, and Track Maintenance). The transitions between the states are defined according to the specifications in Appendix B. The state machine is shown as the Moor automaton with outputs defined by the states. A corresponding phrase from Table 8.3 is generated in every state of the automaton.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\rightarrow < Search > | < NAT > | < RR > | < TM > \\
< Search > &\rightarrow < SearchPhrase >< Search > | < SearchPhrase >< NAT > \\
< NAT > &\rightarrow < NATPhrase >< NAT > | < NATPhrase >< RR > | \\
&\quad < NATPhrase >< Search > \\
< RR > &\rightarrow < RRPhrase >< RR > | < RRPhrase >< TM > | \\
&\quad < RRPhrase >< Search > \\
< TM > &\rightarrow < TMPhrase >< TM > | < TMPhrase >< RR > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > \\
< NATPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_n > | < SectionA_2 >< SectionB_n > \\
< RRPhrase > &\rightarrow < AmbiguousPhrase_0 > | < SharedPhrase_n > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | < SharedPhrase_4 > \\
< TMPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | \\
&\quad < SharedPhrase_4 > | < SectionA_2 >< SectionB_1 > \\
&\quad < SectionA_2 >< SectionB_3 > | < SectionA_2 >< SectionB_4 > \\
< AmbiguousPhrase_0 > &\rightarrow < SectionA_0 >< SectionB_2 > \\
< AmbiguousPhrase_2 > &\rightarrow < SectionA_2 >< SectionB_2 > \\
< SharedPhrase_1 > &\rightarrow < SectionA_0 >< SectionB_1 > \\
< SharedPhrase_3 > &\rightarrow < SectionA_0 >< SectionB_3 > \\
< SharedPhrase_4 > &\rightarrow < SectionA_0 >< SectionB_4 > \\
< SharedPhrase_n > &\rightarrow < SectionA_0 >< SectionB_n > \\
< SectionA_0 > &\rightarrow w_0 w_p \\
< SectionA_2 > &\rightarrow w_2 w_p \\
< SectionB_1 > &\rightarrow w_1 w_t \\
< SectionB_2 > &\rightarrow w_2 w_t \\
< SectionB_3 > &\rightarrow w_3 w_t \\
< SectionB_4 > &\rightarrow w_4 w_t \\
< SectionB_n > &\rightarrow w_n w_t
\end{aligned}
$$

Figure 8.12: Deterministic grammar of the Pluto emitter. The structure of the grammar suggests that it is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Characteristic grammar

The characteristic grammar of the Pluto emitter can be derived employing the same approach as the characteristic grammar of the Mercury emitter described in Section 8.1.1. The distinctive feature of the Pluto emitter is that the words w_p, w_t and w_n cannot be confused with other words. The words w_p and w_t are of shorter length and occupy a predetermined position in the phrase. The word w_n, on the other hand, has a characteristic beam pattern that is distinct from other words. With these points taken into account, we show the characteristic grammar of the Pluto emitter in Fig. 8.13.

Stochastic grammar

Similarly to the case of the Mercury emitter (8.4), we can define the channel impairment transition probability matrix for the Pluto emitter as:

$$\mathbf{P}_o = \left[\begin{array}{ccccc} \mathrm{p}_0 & \mathrm{p}_1 & \mathrm{p}_2 & \mathrm{p}_3 & \mathrm{p}_4 \end{array} \right]^T = \left(\begin{array}{cccccc} p_{0,\varnothing} & p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} & p_{0,4} \\ p_{1,\varnothing} & p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,\varnothing} & p_{2,0} & p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,\varnothing} & p_{3,0} & p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \\ p_{4,\varnothing} & p_{4,0} & p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} \end{array} \right), \tag{8.7}$$

where the rows represent the transmitted radar words, and the columns represent the words inferred by the pulse train analysis layer of the EW receiver from the noisy and corrupted observations.

Using the observation impairment model of (8.7), we present the stochastic grammar of the Pluto emitter in Fig. 8.14.

8.2.2 Pluto state machine synthesis

Non-Self-Embedding Context-Free Grammar test

The production graph of the grammar of the Pluto emitter is shown in Fig. 8.15. The transition matrix of this graph is listed in Fig. 8.16, and the steady-state analysis matrix is shown in Fig. 8.17. The diagonal of this matrix

$$\mathbf{diag}\left(\mathrm{M}^{\leq N}(G)\right) = \left[\begin{array}{ccccccc} o & l & l & l & l & o & \ldots \quad o \end{array} \right]^T,$$

indicates that Pluto grammar passes the NSE Context-Free Grammar test.

Pluto state machine

Applying the Dulmage-Mendelsohn decomposition to the transition matrix M shown in Fig. 8.16 we obtain 19 strongly-connected components of the production graph of the Pluto emitter. They are listed in Table 8.4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\rightarrow < Search > | < NAT > | < RR > | < TM > \\
< Search > &\rightarrow < SearchPhrase >< Search > | < SearchPhrase >< NAT > \\
< NAT > &\rightarrow < NATPhrase >< NAT > | < NATPhrase >< RR > | \\
&\quad < NATPhrase >< Search > \\
< RR > &\rightarrow < RRPhrase >< RR > | < RRPhrase >< TM > | \\
&\quad < RRPhrase >< Search > \\
< TM > &\rightarrow < TMPhrase >< TM > | < TMPhrase >< RR > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > \\
< NATPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_n > | < SectionA_2 >< SectionB_n > \\
< RRPhrase > &\rightarrow < AmbiguousPhrase_0 > | < SharedPhrase_n > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | < SharedPhrase_4 > \\
< TMPhrase > &\rightarrow < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | \\
&\quad < SharedPhrase_4 > | < SectionA_2 >< SectionB_1 > \\
&\quad < SectionA_2 >< SectionB_3 > | < SectionA_2 >< SectionB_4 > \\
< AmbiguousPhrase_0 > &\rightarrow < SectionA_0 >< SectionB_2 > \\
< AmbiguousPhrase_2 > &\rightarrow < SectionA_2 >< SectionB_2 > \\
< SharedPhrase_1 > &\rightarrow < SectionA_0 >< SectionB_1 > \\
< SharedPhrase_3 > &\rightarrow < SectionA_0 >< SectionB_3 > \\
< SharedPhrase_4 > &\rightarrow < SectionA_0 >< SectionB_4 > \\
< SharedPhrase_n > &\rightarrow < SectionA_0 >< SectionB_n > \\
< SectionA_0 > &\rightarrow W_0 w_p \\
< SectionA_2 > &\rightarrow W_2 w_p \\
< SectionB_1 > &\rightarrow W_1 w_t \\
< SectionB_2 > &\rightarrow W_2 w_t \\
< SectionB_3 > &\rightarrow W_3 w_t \\
< SectionB_4 > &\rightarrow W_4 w_t \\
< SectionB_n > &\rightarrow w_n w_t \\
W_i &\rightarrow \varnothing | w_0 | w_1 | w_2 | w_3 | w_4 \\
&\quad i = 0, \ldots, 4
\end{aligned}
$$

Figure 8.13: Characteristic grammar of the Pluto emitter. This grammar, like its deterministic counterpart, is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$
\begin{aligned}
< State > &\xrightarrow{1} < Search > | < NAT > | < RR > | < TM > \\
< Search > &\xrightarrow{1} < SearchPhrase >< Search > | < SearchPhrase >< NAT > \\
< NAT > &\xrightarrow{1} < NATPhrase >< NAT > | < NATPhrase >< RR > | \\
&\quad < NATPhrase >< Search > \\
< RR > &\xrightarrow{1} < RRPhrase >< RR > | < RRPhrase >< TM > | \\
&\quad < RRPhrase >< Search > \\
< TM > &\xrightarrow{1} < TMPhrase >< TM > | < TMPhrase >< RR > | \\
&\quad < TMPhrase >< Search > \\
< SearchPhrase > &\xrightarrow{1} < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > \\
< NATPhrase > &\xrightarrow{1} < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_n > | < SectionA_2 >< SectionB_n > \\
< RRPhrase > &\xrightarrow{1} < AmbiguousPhrase_0 > | < SharedPhrase_n > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | < SharedPhrase_4 > \\
< TMPhrase > &\xrightarrow{1} < AmbiguousPhrase_0 > | < AmbiguousPhrase_2 > | \\
&\quad < SharedPhrase_1 > | < SharedPhrase_3 > | \\
&\quad < SharedPhrase_4 > | < SectionA_2 >< SectionB_1 > \\
&\quad < SectionA_2 >< SectionB_3 > | < SectionA_2 >< SectionB_4 > \\
< AmbiguousPhrase_0 > &\xrightarrow{1} < SectionA_0 >< SectionB_2 > \\
< AmbiguousPhrase_2 > &\xrightarrow{1} < SectionA_2 >< SectionB_2 > \\
< SharedPhrase_1 > &\xrightarrow{1} < SectionA_0 >< SectionB_1 > \\
< SharedPhrase_3 > &\xrightarrow{1} < SectionA_0 >< SectionB_3 > \\
< SharedPhrase_4 > &\xrightarrow{1} < SectionA_0 >< SectionB_4 > \\
< SharedPhrase_n > &\xrightarrow{1} < SectionA_0 >< SectionB_n > \\
< SectionA_0 > &\xrightarrow{1} W_0 w_p \\
< SectionA_2 > &\xrightarrow{1} W_2 w_p \\
< SectionB_1 > &\xrightarrow{1} W_1 w_t \\
< SectionB_2 > &\xrightarrow{1} W_2 w_t \\
< SectionB_3 > &\xrightarrow{1} W_3 w_t \\
< SectionB_4 > &\xrightarrow{1} W_4 w_t \\
< SectionB_n > &\xrightarrow{1} w_n w_t \\
W_i &\xrightarrow{p_i} \varnothing | w_0 | w_1 | w_2 | w_3 | w_4 \\
&\quad i = 0, \ldots, 4
\end{aligned}
$$

Figure 8.14: Weighted grammar of the Pluto emitter. This grammar, like its deterministic counterpart, is a Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

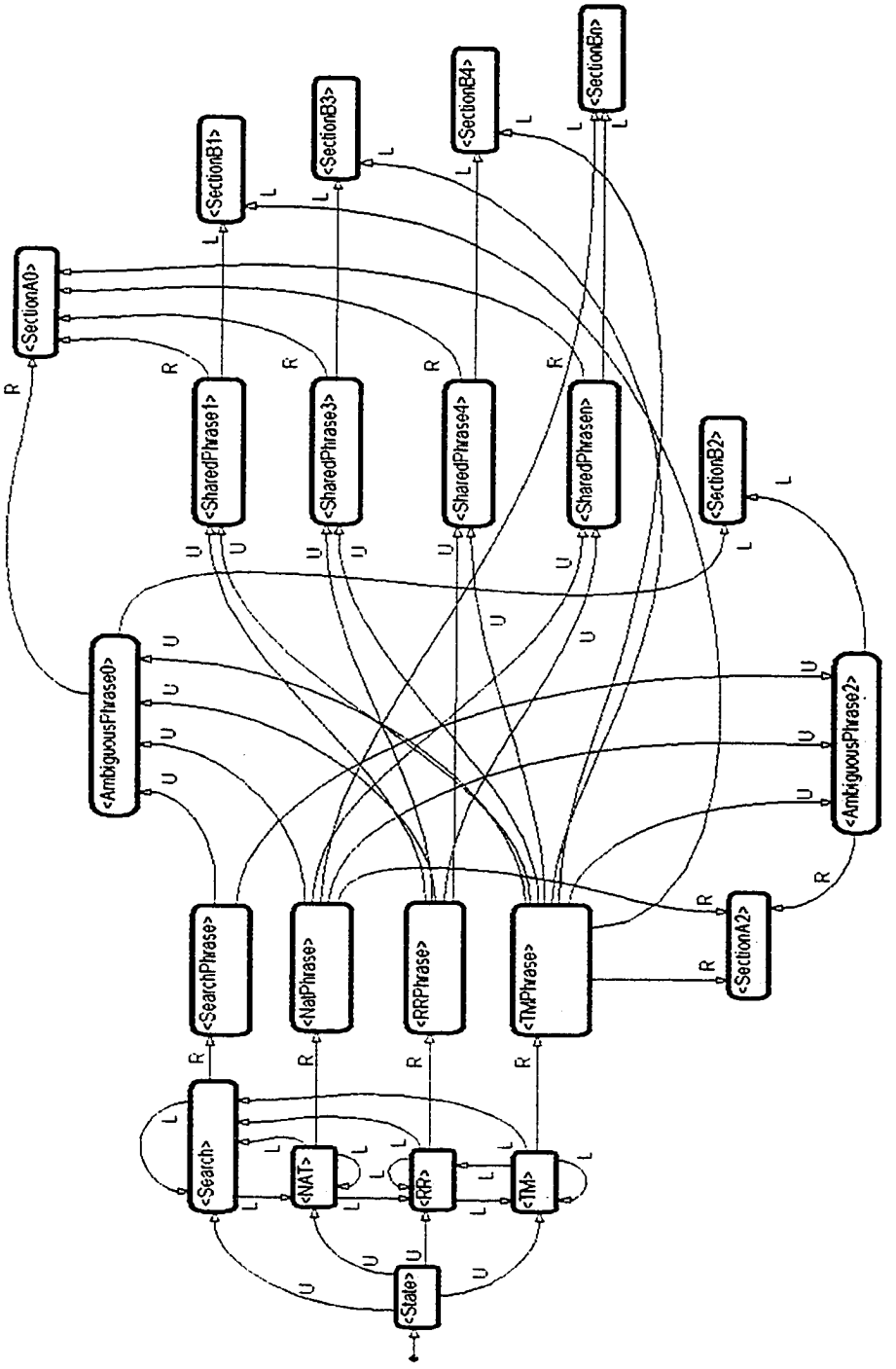

Figure 8.15: Production graph of the Pluto grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

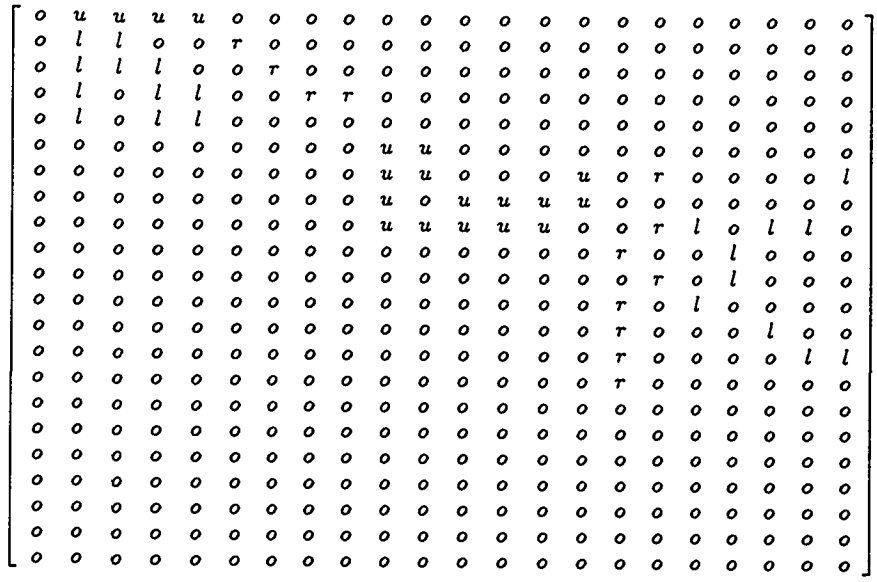

Figure 8.16: Transition matrix $\mathbf{M}(G)$ for the Pluto emitter grammar based on the grammar in Fig. 8.12.

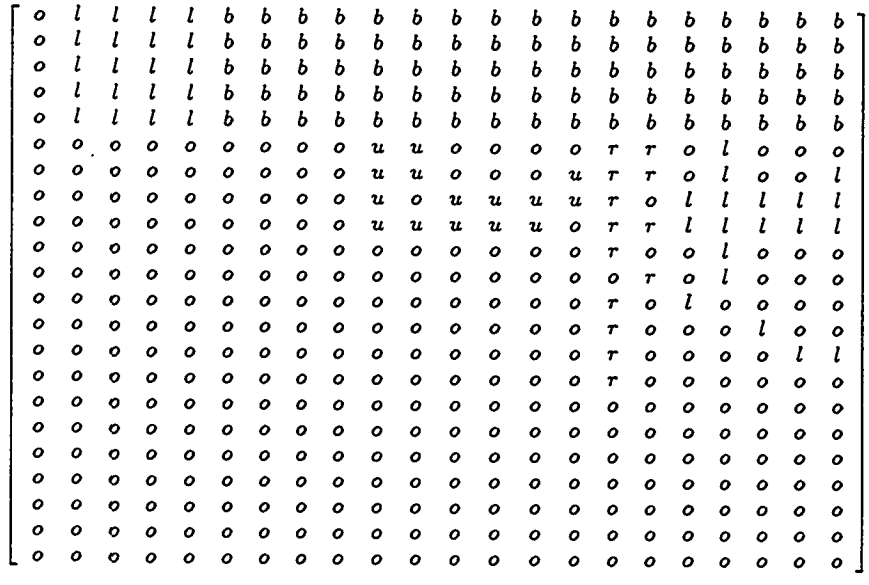

Figure 8.17: Production graph analysis matrix $\mathbf{M}^{\leq N}(G) = \sum_{i=1}^{n} \mathbf{M}^{i}(G)$ for the Pluto emitter grammar. The diagonal elements of this matrix are $\mathbf{diag}(\mathbf{M}^{\leq N}(G)) = \left[\begin{array}{ccccccc} o & l & l & l & l & o & \ldots & o \end{array}\right]^{T}$. This indicates that the Pluto grammar is a Non-Self-Embedding Context-Free Grammar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

As shown in Chapter 5, each strongly connected component of the production graph corresponds to a Finite-State Grammar (FSG). Finite-State Automata for each of these FSGs are shown on Fig. 8.18. The complete state machine of the Mercury emitter can be obtained by applying the FSA composition operation[2]. By combining the resulting state machine with the channel impairment probability distribution (8.7), we obtain a Hidden Markov Model of the Pluto emitter at the radar word-level.

8.3 Summary

In this chapter, we presented two case studies describing the application of syntactic modeling to real-life Anti-aircraft Defence radars. We demonstrated possible approaches to deriving deterministic, characteristic, and stochastic Context-Free Grammars modeling word-level behavior of Multi-Function Radars. We also demonstrated how the Finite-State Automata synthesis procedure presented in Chapter 5 can be applied to verify the Non-Self-Embedding property of the radar grammars and to generate Finite-State Automata models of the radars.

Approaches to grammatical derivation presented here are by no means unique. The interested reader may derive a better grammatical structure for both case studies relying on the emitter specifications provided in Appendices A and B. More functional details can also be included in the production rules of the grammar, yielding potentially more accurate emitter models.

The basic derivation principles and techniques presented in this chapter can be applied to the modeling of a variety of Multi-Function Radars and, possibly, other Discrete Event Systems.

[2]Due to very large size of the final Pluto state machine, we do not include it in this chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

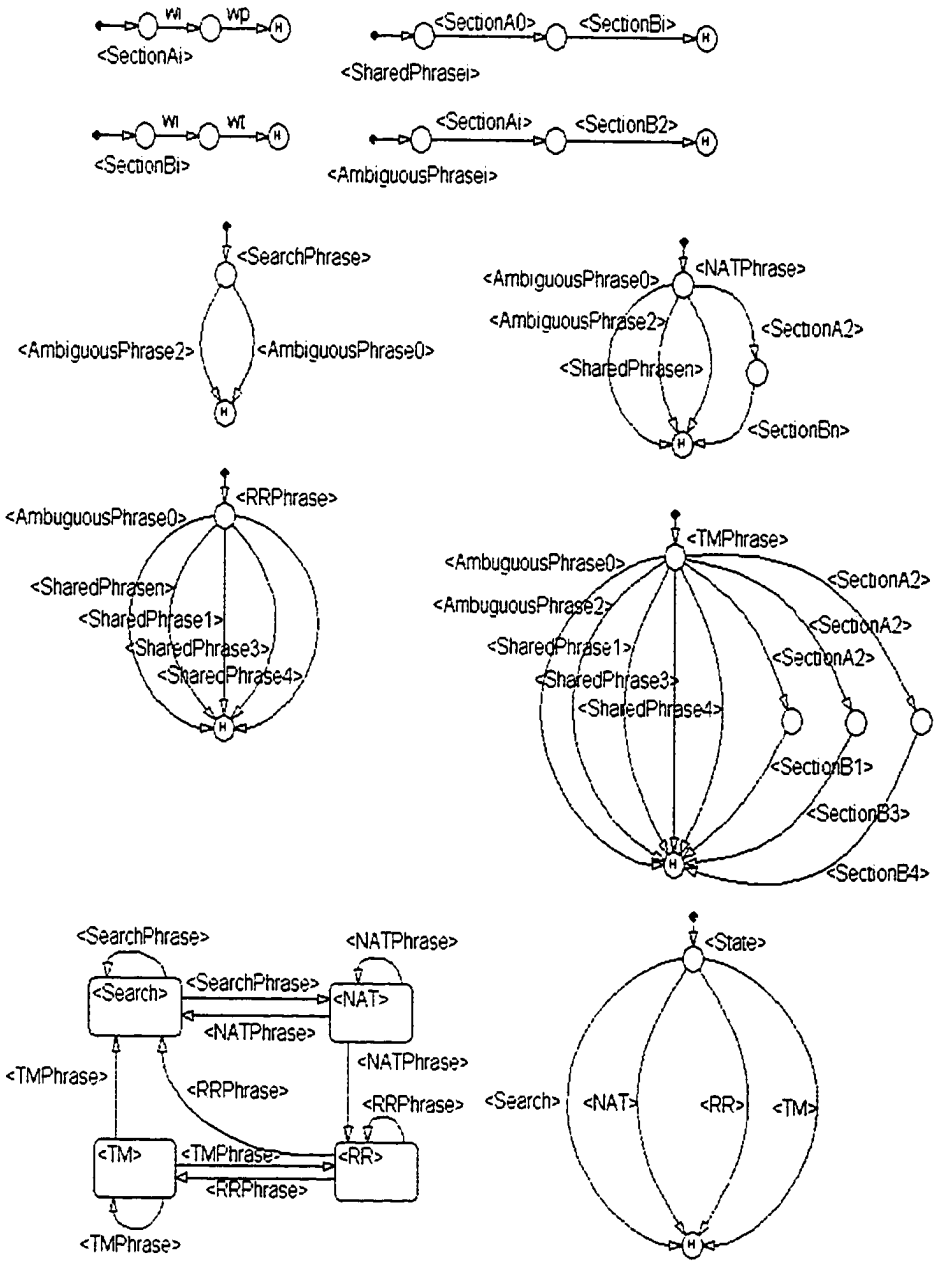

Figure 8.18: Pluto state machine components.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

#	Nonterminals in the strongly connected component
1	$< State >$
2	$< Search >$ $< NAT >$ $< RR >$ $< TM >$
3	$< SearchPhrase >$
4	$< NATPhrase >$
5	$< RRPhrase >$
6	$< TMPhrase >$
7	$< AmbiguousPhrase_0 >$
8	$< AmbiguousPhrase_2 >$
9	$< SharedPhrase_1 >$
10	$< SharedPhrase_3 >$
11	$< SharedPhrase_4 >$
12	$< SharedPhrase_n >$
13	$< SectionA_0 >$
14	$< SectionA_2 >$
15	$< SectionB_1 >$
16	$< SectionB_2 >$
17	$< SectionB_3 >$
18	$< SectionB_4 >$
19	$< SectionB_n >$

Table 8.4: Strongly connected components of the Pluto production graph of Fig. 8.15.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 9

Pulse train analysis application

In this chapter, we would like to demonstrate the use of the pulse-level model developed in Chapter 6 in the *pulse train analysis* application. As we described in Section 6.1, pulse train analysis is concerned with extracting patterns from radar signals observed in the noisy environment.

The most common patterns of interest in radar signal processing applications are structured groups of pulses that we call radar *words*. Extracting the words from radar pulse sequences enables the application of word-level modeling discussed in Chapters 5 and 8 to solutions of such problems as radar classification, radar state estimation, analysis of instantaneous threat posed by the radar system to the target, and prediction of radar behavior for the purpose of applying active countermeasures such as predictive jamming.

There are three main challenges to the problem of radar pulse train analysis (see Section 6.1):

1. The observed radar signals are corrupted by noise due to propagation effects of electromagnetic waves through the stochastic wireless environment as well as the thermal noise.

2. The uncooperative nature of interaction between the radar and the tactical EW equipment that records the observations of radar signals results in absence of synchronization between the two. This introduces further distortions into observed signals.

3. Recorded pulse sequences are subject to further corruption by imperfect de-interleaving.

These issues result in observed radar pulse trains containing *spurious pulses* due to thermal noise peaks and de-interleaver leakage. Also, observed sequences of pulses can be sparse. Pulses that were emitted by the radar, but were not recorded by the receiver or ended up being misclassified as originating from another emitter are called the *missing pulses*.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Therefore, the problem of pulse train analysis involves finding the patterns in the noisy and corrupted observation sequence that match the predefined templates or models of words with high probability. In this chapter, we use models of radar words based on Hidden Markov Models (HMMs) as described in Chapter 6. We will develop a customized version of the Viterbi algorithm described in Chapter 4 that finds this maximum-likelihood pattern match.

The rest of this chapter is organized as follows. Section 9.1 develops the customized version of the Viterbi algorithm to address the pattern matching needs of the pulse train analysis application. We utilize the structure of the pulse-level model developed in Chapter 6 to reduce the computational cost of the generally expensive Viterbi algorithm. In the end, we obtain a very efficient variation of the algorithm with a computational complexity being close to linear.

Section 9.2 presents some pulse train analysis simulation results to illustrate the application of the Viterbi algorithm to pulse train analysis of real-life Anti-aircraft Defence radars.

Finally, Section 9.3 presents the summary of this chapter.

9.1 Application of the Viterbi Algorithm

Given the stochastic model (6.10), the problem of pulse train analysis can be viewed as a problem of detection and recognition of radar words in a binary observation sequence. One possible solution rests in the use of the Viterbi algorithm, described in Section 4.2, as follows:

1. For each radar word, $k = 1, 2, \cdots, K$, a HMM $\{\lambda_k\}$ is constructed, as per (6.10).

2. For each HMM, the sequence of Viterbi log-scores is calculated for the binary observation sequence, as described in Section 4.2.

3. The relative peaks in the sequence of Viterbi log-scores correspond to likely starting times for radar words in the binary observation sequence.

4. The Viterbi backtracking procedure described in Section 4.2 yields the likely distribution of individual pulse labels, *i.e.*, it identifies which pulses are explained by the radar model, which ones are likely to be spurious, and which ones are likely to be missing.

It is well known (Rabiner and Juang 1993) that the asymptotic computational complexity of the Viterbi algorithm is $\mathcal{O}[LM_k^2]$, where L denotes the length of the binary observation sequence and M_k the number of states in λ_k. Since the number of states in the HMM is directly proportional to the quantization resolution (inversely proportional to the observer clock period T_{obs}), the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

computational complexity of the general-purpose Viterbi algorithm described in Chapter 4 is prohibitively high. Fortunately, a specialized implementation of the algorithm is possible by virtue of the sparseness of the transition matrices (6.8). The computational complexity of this specialized implementation is $\mathcal{O}[(N_p \times D)(M_k \times W_k)]$, where $N_p \ll L$ is the number of non-zero elements in the binary observation sequence, and D and W_k are constants discussed later in this section. Most importantly, this specialized implementation scales linearly as a function of the number of states M_k.

The first observation that leads to a reduction in computational complexity is that the scores need only be evaluated at the positions in the binary observation sequence where pulse leading edges are present. The observation sequence has the total length L, and contains N_p nonzero elements. Whereas both M_k and L increase with decreasing T_{obs}, the number of observed pulses N_p is independent of T_{obs}.

One difficulty with using this approach is that if the first several pulses of a radar word are missing in the binary observation sequence, then the whole radar word may not be detected. One possible solution is to introduce backtracking of depth-D in the scoring algorithm. This involves the calculation of D scores for each non-zero element in the binary observation sequence. For the binary channel observation model introduced in Section 6.1, the probability of missing several pulses in a row decreases exponentially with the number of pulses. Therefore, D can be chosen so that this probability is below a user-specified threshold. The resulting complexity bound is $\mathcal{O}[(N_p \times D)M_k^2]$.

The second and most significant observation that leads to the reduction in computational complexity is that the transition probability matrix $\mathbf{A}_k$ is very sparse. Fig. 9.1 (a) and (b) illustrate how this can be exploited to reduce the number of paths explored by the Viterbi algorithm. For example, consider the HMM shown in Fig. 9.1 (a). It is often convenient to analyze the performance of the Viterbi algorithm using a lattice-like graphical structure called a *trellis*. Fig. 9.1 (b) shows the trellis corresponding to all possible transitions of the HMM of Fig. 9.1 (a). The rows on the diagram represent the sequential state numbers in the HMM (1-16), and the columns, the quantized time steps of the Viterbi algorithm (0-15). The shortest path through the trellis (State 16 at Time 11) corresponds to the case in which no positive quantization error has occurred. The longest path (State 16 at Time 15) corresponds to the case in which a positive quantization error has occurred for every pulse. The maximum width of the diagonal belt in the Viterbi trellis W_k is equal to $15 - 11 = 4$. In general, $W_k = W(\lambda_k)$ is an explicit function of the radar word structure, with its upper bound being the total number of pulses in the word. Hence, clearly, the resulting computational complexity is

$$\mathcal{O}[(N_p \times D)(M_k \times W_k)] \ll \mathcal{O}[LM_k^2]. \tag{9.1}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

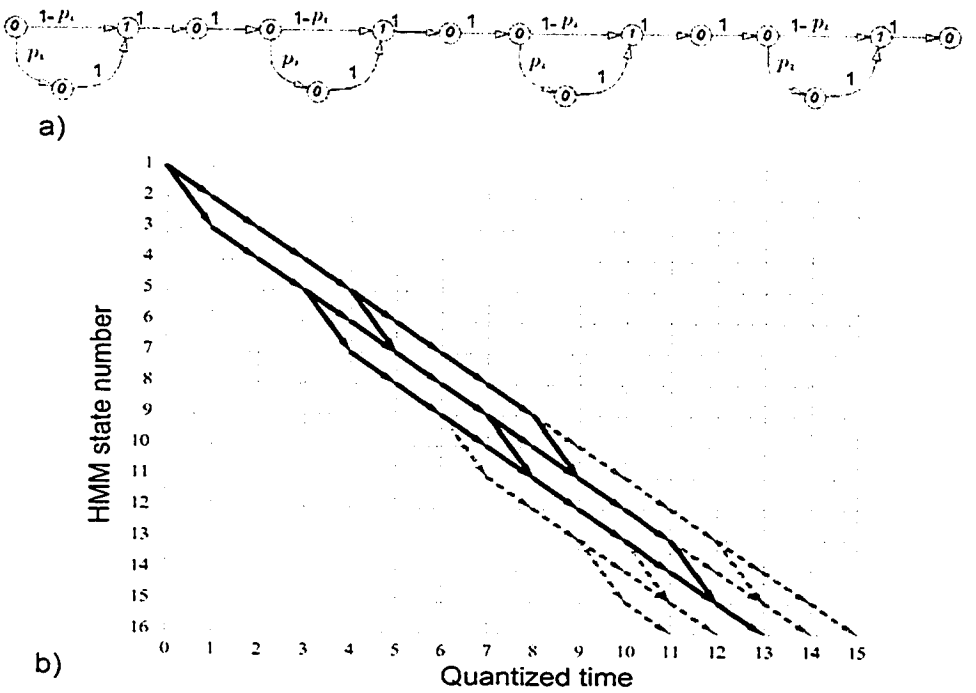

Figure 9.1: Example of the trellis path of the Viterbi algorithm. (a) shows a specific Markov chain for which the trellis paths are depicted in (b).

It is much lower than the complexity of the conventional Viterbi algorithm and is linear with respect to the number of states of the HMM radar word template.

Note that in practice the trellis can be pruned even further. Indeed, each radar word has a certain duration (for example, see Fig. 6.1 (b)), and this duration in the quantized domain can be evaluated using (6.3). Due to this duration, valid paths are word-specific and are normally located between the shortest and the longest paths. Suppose that in Fig. 9.1 the quantized word duration is 13. Then all the paths shown by grey dashed lines in Fig. 9.1 (b) would be invalid, and could be eliminated. For very large radar word HMMs, this can yield up to 50% of additional reduction in the amount of computations involved by eliminating about a half of paths in the trellis, along which computation otherwise would have to be performed.

9.2 Simulation Results

In this section, we present pulse train analysis simulation results for radar emitters having the word structures shown in Fig. 6.1. We have performed a number of experiments where we generated synthetic pulse sequences for real-life Anti-aircraft Defence radars which are described in Appendices A and B.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

We then corrupted these sequences with quantization distortions and noise as described in Chapter 6, and simulated de-interleaving errors by randomly removing some pulses from the generated sequence.

Specifically, the experiments were set up using the following procedure:

1. An emitter was picked and synthetic data were generated[1]. The experiment generated between 35 and 75 seconds of undistorted radar signals emitted by the radar in response to several targets (up to 5 targets at any single point in time).

2. Subsequently, the data were quantized according to the process defined in Chapter 6 in (6.3).

3. Next, every quantization bin containing a pulse was considered. The outcome of the simple random trail with the probability p_{miss} established whether this pulse was kept in the bin, or erased from it. This simulated the missed pulses discussed in Section 6.1.

4. Finally, a sample of the Poisson process of appropriate length (between 35 and 75 seconds long, depending on the length of the synthetic radar simulation data) was generated. This process simulated the effect of spurious pulses. The sequence of spurious pulses so generated was then combined with the synthetic radar data from the previous step.

Different experiments involved different sets of simulated radar data as well as different values parameters p_{miss} and Poisson rate ρ.

We performed scoring and word extraction on this data using the Viterbi algorithm presented in Section 9.1. Fig. 9.2 (a) shows results for approximately 35ms of data from the radar emitter having the word structure shown in Fig. 6.1 (a). Here, $T_{\text{obs}} = 1.1\mu$s, and about 20% of the radar pulses were missed. The spur rate $\rho = \frac{p_{\text{spur}}}{T_{\text{obs}}}$ was set to 6000pulses/s. The top graph of Fig. 9.2 (a) shows the Viterbi scores, and the bottom graph plots the sequence of words extracted from the scores of the top graph. The emitter words and their respective scores are color-coded so that the words w_1, w_3, w_5, w_6, and w_7 are represented, respectively, by red, blue, pink, black, and green (see legend in Fig. 9.2).

The scores for each word peak sharply in the location of the start of the word in the sequence. To extract words, we used linear thresholds on Viterbi scores. Each word was assigned an independent score threshold value chosen to maximize the probability of detection of the word while simultaneously

[1]The simulator design was a joined effort of the author of this thesis and Mr. Brian Currie at McMaster University. In this simulator various targets were introduced at random, and the response of the radars to these targets was emulated according to the radar specification defined in Appendices A and B.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Percentage of missed pulses	20%			30%		50%
Spur rate ρ, *pulses/sec*	6000	10000	50000	6000	10000	6000
Mercury emitter word detection error	less then 2%	less then 5%	$\approx$ 8%	$\approx$ 12%	$\approx$ 17%	over 30%
Pluto emitter word detection error	less then 4%	less then 9%	$\approx$ 13%	$\approx$ 20%	$\approx$ 30%	over 40%

Table 9.1: Summary of the pulse train analysis experiments.

minimizing the probability of a false alarm (Dilkes 2004; Dilkes 2005). The word was considered detected if the Viterbi score exceeded the value of the threshold.

The extracted word sequence exactly matched the actual sequence emitted by the radar. This particular emitter is capable of searching for and tracking multiple targets in a time-multiplexed fashion. The transmission is arranged into blocks of four sequential words, each corresponding to either a tracking sequence or a searching pattern. In Fig. 9.2 (a), the emitter was tracking one target using a block consisting of four repetitions of the seventh word, $w_7 - w_7 - w_7 - w_7$. The subsequent block was allocated to tracking a second target using a $w_6 - w_6 - w_6 - w_6$ sequence. The next three blocks (or 12 words) follow a different pattern used to search for additional targets. The radar then resumes tracking the first target using the $w_7 - w_7 - w_7 - w_7$ block, and the entire five-block pattern repeats, with minor variations. The evolution of the radar signals in time is the same as in Fig. 2.2.

Fig. 9.2 (b) shows results of a similar experiment performed with $17ms$ of data generated by the emitter having the word structure shown in Fig. 6.1 (b). Although this particular radar emitter has only 5 words, we have considered the termination character and the dead time between segments in Fig. 6.1 (b), as well as an occasional period of emitter silence (absence of any pulse radiation for a known period of time), as separate words. Therefore, the total number of processor words is equal to 8. In this example, the radar was performing target acquisition. The extracted word sequence exactly matched the expectations.

The results of other experiments are summarized in Table 9.1. This table lists average results obtained from a large number of independent simulation experiments with various percentages of missed pulses and various spur rates. The table lists radar-word detection errors in average percents of unrecognized radar words. From the table, we can make the following important observations:

1. On average, the performance of the pulse train analysis algorithm for the Mercury emitter word recognition is better than the performance of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

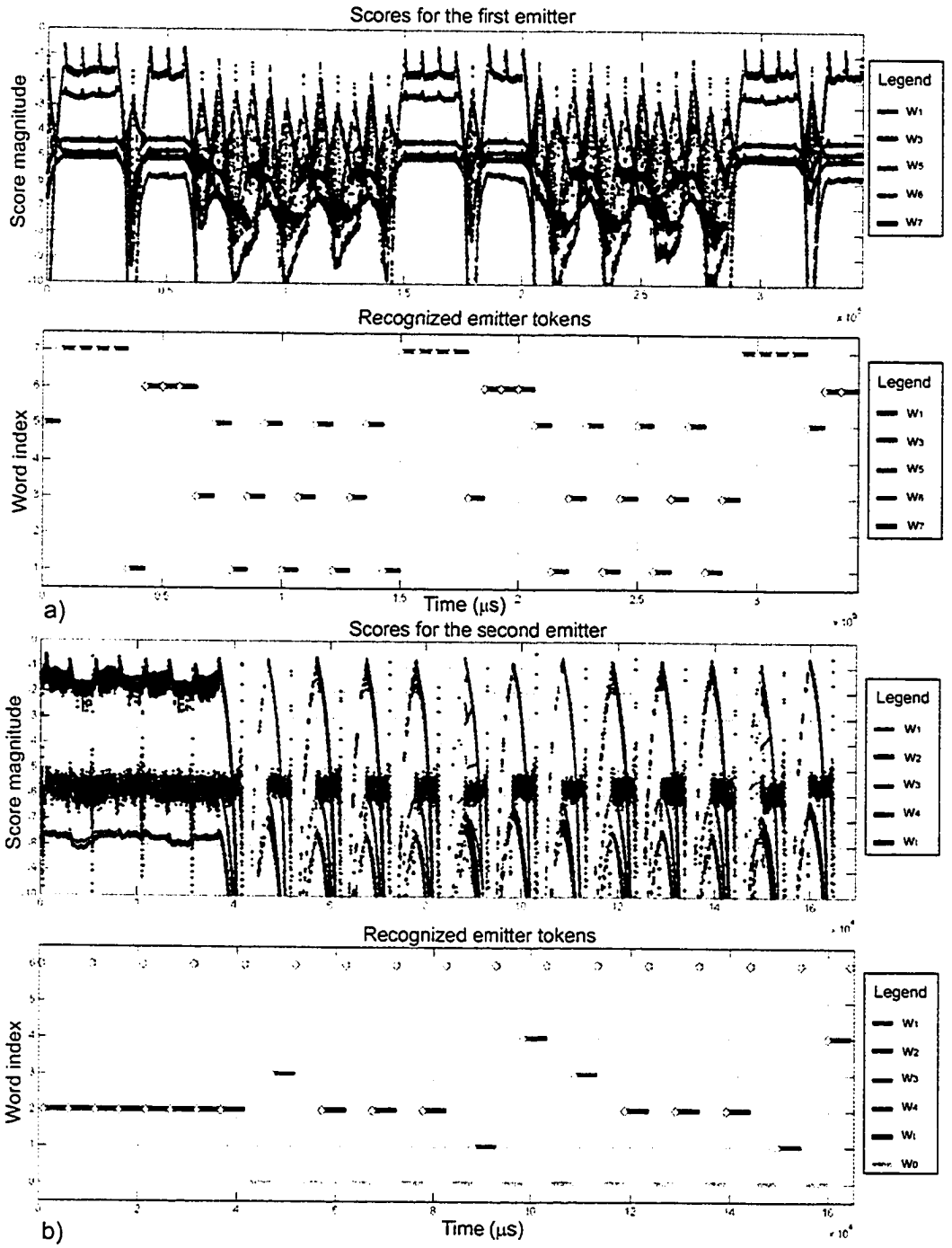

Figure 9.2: Simulation results showing Viterbi scores and extracted word sequence for (a) the radar words in Fig. 6.1 (a), and (b) the radar words in Fig. 6.1 (b).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the Pluto emitter word recognition. This can be explained by the fact that there is more similarity between the words of Pluto than those of Mercury. Mercury words were therefore easier to recognize even with the high level of spurious pulse rates.

2. The increase in the miss rate (decrease in the probability of detection) has a more detrimental effect on the performance of the algorithm than the increase in the spur rate. This can be explained by the fact that the algorithm is fine-tuned to recognize predetermined pulse sequences (radar words) even with the large amount of spurs. As the probability of detection decreases, more and more pulses from the radar word structure are lost. This increases the level of ambiguity in the decision process of the extraction algorithm.

9.3 Summary

In this chapter, we discussed radar pulse train analysis as an application to the word-level modeling technique discussed in Chapter 6.

Radar pulse train analysis is concerned with the recognition of structured pulse sequences characteristic for a given Multi-Function Radar. These sequences called radar words have to be identified and extracted from the noisy and corrupted observations of radar signals. Thus, the problem of pulse train analysis is a problem of pattern recognition.

To solve this problem, we utilized the pulse-level radar model presented in Section 6.2, and developed a modification of the Viterbi algorithm applied to scoring of pulse sequences against the HMM word templates. We showed the computational complexity of this algorithm to be lower than polynomial. We also presented simulated computer experiments in which sequences of radar words were extracted from synthetic radar pulse trains of two realistic radar emitters described in Appendices A and B.

We demonstrated encouraging simulation results that can be explained by the fact that HMM templates are well-suited for capturing the structure of radar words, and by the fact that Viterbi algorithm is sensitive to the Hidden Markov Model structure.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 10

Discussion

This thesis has undertaken a detailed examination of the problem of syntactic modeling of Multi-Function Radars with a focus on potential applications of radar emitter recognition, radar state estimation, and radar pulse train analysis. The aim of this chapter is to conclude the thesis by summarizing the main results of our investigation and highlighting the most important contributions of the work.

We first examine in detail the major contributions that this thesis offers to the scientific literature. Finally, we present quick summary of the results of this thesis as well as some important concluding remarks.

10.1 Contributions to the literature

This thesis presents the following contributions to the field of radar modeling and radar signal processing:

Model-centric principle. To resolve the issue of non-homogeneous intelligence data, and to address the growing complexity of the radar systems, we propose a *model-centric* approach to Electronic Intelligence (ELINT). This is a novel approach that promises significant advantages over the traditional data-centric or parametric ELINT organization.

We demonstrate how mathematical models for radar emitters can be derived using data-centric ELINT records. These mathematical models form a homogeneous basis for the model-centric ELINT and open new avenues for radar signal processing that traditional, data-centric approaches did not support.

Layered radar signal architecture. Our approach of radar model complexity management based on the *"divide and conquer"* principle applied to the hierarchical radar signal architecture provides means of clear separation of low-level elements of radar signals such as individual pulses from

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the higher-level elements such as radar words and elements of radar state dynamics. This allows to keep the radar model complexity manageable and to exercise a modular approach to building radar signal processing systems. Traditional radar signal processing techniques do not go beyond the tasks that we associate with the pulse level of our hierarchical model. To the best of our knowledge, this is the first time radar signal processing is taken beyond the pulse train analysis into a direction of more efficient utilization of information encoded in the radar pulse sequences. The proposed layered hierarchical model directly supports this efficiency improvement and therefore is viewed by the author as another important contribution.

Syntactic radar modeling. The novelty of this work rests in the view of complex radar emitters as abstract Discrete Event Systems that broadcast messages using some stochastic formal language. The most widely known and powerful model for formal languages is a grammar. Therefore, we can model radar languages with stochastic grammars, and process radar messages using methods from the theory of syntax analysis, which we also view as a strong and novel contribution.

Syntactic modeling has proven to be a powerful technique of solving data modeling and analysis problems in the areas of speech processing, computer languages, and bioinformatics. However, to the best of our knowledge, this thesis uses syntactic modeling in the area of radar modeling and radar signal processing for the first time.

Specific contributions to the open literature are summarized in the list on page (xiv) in the beginning of this thesis and include the following publications:

1. The initial indication of possible applicability of Hidden Markov Models (HMMs) to Multi-Function Radar (MFR) modeling was published at the IEEE International Conference on Decision and Control (CDC'2003) (see item [6] on page (xiv) or item (Visnevski, Krishnamurthy, Haykin, Currie, Dilkes, and Lavoie 2003) in bibliography).

2. The pulse-level modeling technique presented in Chapter 6, and the approach to radar pulse train analysis using HMM pulse-level models described in Chapter 9 were introduced in the papers at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'2005) (see item [5] on page (xiv) or item (Visnevski, Haykin, Krishnamurthy, Dilkes, and Lavoie 2005) in bibliography) and in the IEEE Transactions on Aerospace and Electronic Systems (see item [2] on page (xiv) or item (Visnevski, Dilkes, Lavoie, Krishnamurthy, and Haykin 2004) in bibliography). In addition, the special case of radar pulse train analysis using static radar word templates introduced in Appendix C is presented in IEEE Transactions on Aerospace and Electronic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Systems (see item [3] on page (xiv) or item (Visnevski, Dilkes, Krishnamurthy, Haykin, and Lavoie tion) in bibliography).

3. The core of the syntactic word-level modeling of MFRs described in Chapters 5 and 8 is presented in the special technical memorandum of Defence Research & Development Canada (see item [9] on page (xiv) or item (Dilkes and Visnevski 2004) in bibliography), at the IEEE International Radar Conference (IRC'2005) (see item [4] on page (xiv) or item (Visnevski, Dilkes, Haykin, and Krishnamurthy 2005) in bibliography), in the special issue of the IEEE Signal Processing Magazine focused on Knowledge Based Systems for Adaptive Radar Detection, Tracking and Classification (see item [1] on page (xiv) or item (Visnevski, Haykin, Dilkes, Krishnamurthy, and Lavoie 2006) in bibliography), in the Proceedings of the IEEE (see item [7] on page (xiv) or item (Haykin, Visnevski, Dilkes, Krishnamurthy, and Lavoie tion) in bibliography), and in Proceedings of the IEEE workshop on Digital Applications of Signal Processing (DASP'2005) (see item [8] on page (xiv) or item (Haykin, Visnevski, Dilkes, Krishnamurthy, and Lavoie 2005) in bibliography).

10.2 Conclusion

In this thesis, we have presented a new approach to radar modeling based on the principles of syntactic pattern recognition. We have developed a hierarchical Multi-Function Radar model that helps mitigate the effects of radar model complexity. We have also shown that for two critical levels of this hierarchy, syntactic models based on either Finite-State Grammars or Non-Self-Embedding Context-Free Grammars can be developed. These syntactic models provide compactness of Multi-Function Radar representation in Electronic Threat Libraries – a feature of critical importance for Electronic Warfare.

We have also presented methods of synthesizing Finite-State Automata and Hidden Markov Models from the syntactic models of the radars. This important step provides significant benefits to radar signal processing. Thus, efficient and simple finite-state system analysis algorithms can potentially be used for radar signal processing applications such as pulse train analysis, radar emitter recognition, and radar state estimation.

As we mentioned earlier, the syntactic radar modeling techniques presented in this thesis are centered around the idea of radar signal processing. The word-level models presented in Chapters 5 and 8 included only the minimal necessary information that allows to explain and analyze the observations of radar signals. This approach ensures low cost and complexity of radar signal processing algorithms. This feature is of critical importance to the real-time software of the tactical Electronic Warfare equipment, since the decisions regarding the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

electronic threats have to be made very quickly.

The same modeling principles allow to keep the entries of Electronic Threat Libraries compact and the overall size of the libraries small. However, this modeling approach comes with a cost. By focusing on modeling that explains the observations of radar signals, the modeling approach yields a syntactic model that describes a language that is a superset of the original language of the radar. In other words, the language that the syntactic model of the radar represents contains all possible sequences of radar signals as well as some sequences that the actual radar cannot generate. Consequently, syntactic models developed in this thesis cannot be used in the dual problem of radar signal generation or simulation since they may generate invalid data.

The fact that radar models described in this thesis cannot be used as radar signal generators does not exclude syntactic modeling from the set of useful tools for radar simulation. Syntactic models optimized for simulation tasks can be developed using similar syntactic principles, as described in this thesis. In fact, special syntactic models called *indexed* and *programmed grammars* (Fu 1982; Hopcroft and Ullman 1979) use a system of checks and balances to restrict the strings generated by Context-Free Grammars to a much more narrow set with specified properties. Therefore, we suggest that the valid radar simulation model can be obtained from the same Non-Self-Embedding Context-Free Grammars described in this thesis by defining the set of restrictions over the strings that these grammars can generate. These restrictions can be defined as either an index set or a set of programmed rules as described in (Fu 1982; Hopcroft and Ullman 1979).

Syntactic modeling has been used extensively in the past by computational linguists in representing artificial programming languages. More recently, the syntactic formalism has been successfully applied to natural language processing and to the problem of gene sequencing and bioinformatics. However, application of syntactic modeling to problem domains outside the ones mentioned above is very rare. In fact, we are aware of only two publications that dealt with Context-Free Grammars outside of the context of problems mentioned above – Zhu and Garcia-Frias (2002) as well as Zhu and Garcia-Frias (2004) applied Stochastic Context-Free Grammars in the context of bursty wireless channel modeling. As far as we know, this thesis presents the first application of syntactic modeling to the area of radar signal processing. The novelty of this work is viewed by the author as an important contribution.

The syntactic modeling formalism can easily be demonstrated to be applicable to the areas outside radar modeling and Electronic Warfare. In fact, we argue that this approach is very useful in modeling and signal processing problems related to any uncooperative discrete event system observed in the stochastic environment. This makes the contributions of this thesis even more important.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A

Functional specification of the "Mercury" emitter

This appendix contains a sanitized version of a textual intelligence report describing the functionality of an emitter called *"Mercury"*[1]. This description presents an example of a data-centric record within the Electronic Intelligence database that describes the functionality of this Multi-Function Radar (MFR).

General remarks

Radar transmit frequency is an important parameter of the radar functionality that may serve a purpose of uniquely identifying the radar system. However, as we mentioned in Chapter 1, the task of radar identification is outside the scope of this research effort. Therefore, we do not include radar transmit frequency characterization in this report.

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.

Radar words

The timing of this emitter is dictated by a sub-structure 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

[1]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 Defence radars, but has been altered and declassified before the release.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

has the same length (on the order of several milliseconds), and is associated with a fixed integer number of crystal counts. All words have the same generic structure shown in Fig. A.1. They are distinguished by particular values of the Pulse to Pulse Interval (PPI) in Section B as described below.

The word structure for this emitter consists of the following components:

Initial dead time (Section A): The word may start with some dead time coinciding with some integer number of crystal counts.

Pulse-Doppler sequence (Section B): The emitter radiates a certain number of pulses with a particular constant PPI. Each word has its own number of pulses and a characteristic PPI (which remains constant within that word).

Time to switch (Section C): Some additional dead time between the leading edge of the last pulse in the Pulse-Doppler sequence and the beginning of the burst sequence.

Burst sequence (Section D): A burst of pulses is radiated. The burst sequence is an emission of some fixed number of pulses. The parameters of the burst are the same in all words and are unrelated to the PPI of the Pulse-Doppler sequence. The PPI is constant except for the interval between the last two pulses, which is slightly different. The clock governs the beginning of this sequence; however, the clock does not dictate the timing within the sequence.

Final dead time (Section E): The end of the word is padded with some additional dead time.

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. A.1 and Fig. A.2).

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.

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 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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

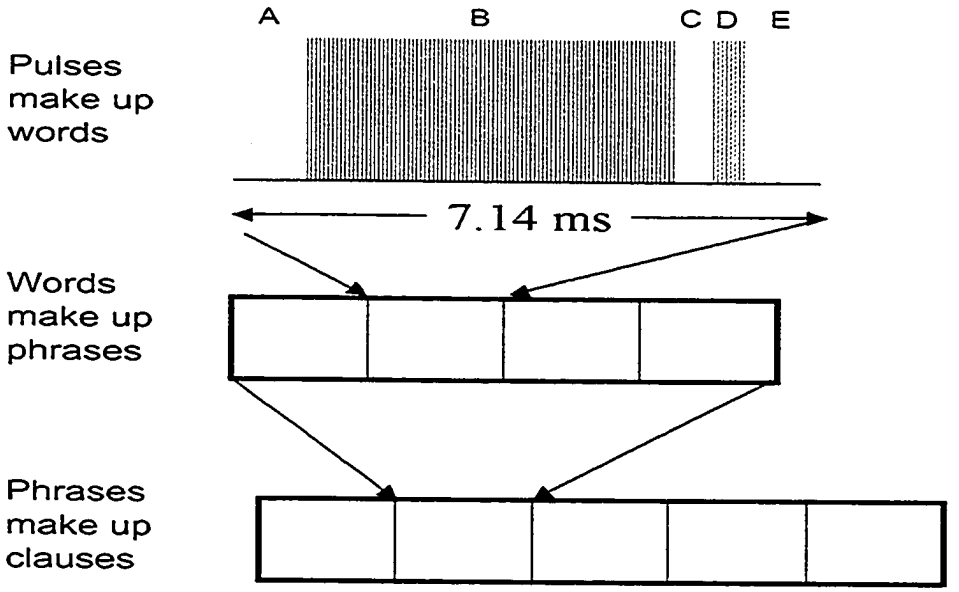

Figure A.1: A three-layer hierarchical radar signal structure of the Mercury emitter (reproduced from Fig. 2.1 for convenience).
Pulse sequences are arranged into groups according to specific patterns. These groups are called *words*. For this particular emitter, words are all of equal length of $7.14ms$, but in general they can be of variable length. Words are made up of five sections (A – E). Sections A, C, and E are dead times of known duration. Section B is a fixed PPI pulse-Doppler sequence, and section D is a scheduled PPI synchronization burst. *Phrases* are made up of several words grouped together (four in the case of this emitter). One phrase is commonly associated with a single task like search or track. *Clauses* are made up of phrases, and the number of phrases in one clause determines the number of tasks the radar can perform simultaneously. A clause is a product of Time Division Multiplexing in MFR operation. This particular radar has five phrases within one clause which means that it can perform five different operations simultaneously.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

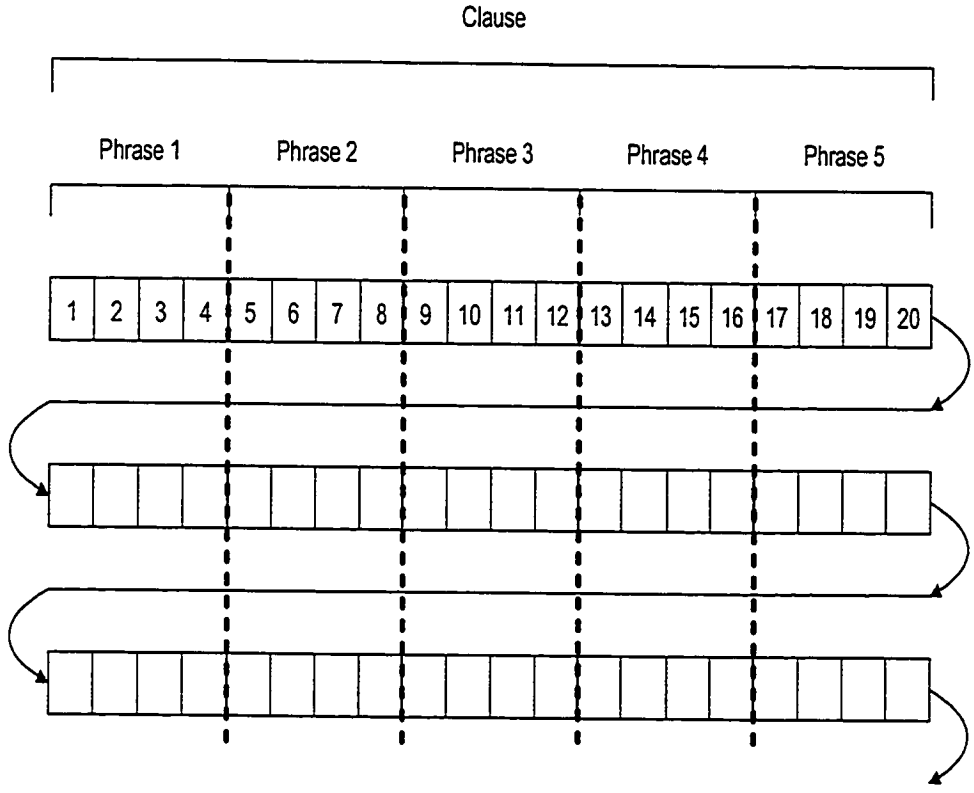

Figure A.2: The output sequence of the Mercury emitter with the signal structure of Fig. A.1 (reproduced from Fig. 2.2 for convenience).
The output sequence of this radar is formed so that the clauses follow each other sequentially. As soon as the last word of the last phrase of a clause is emitted, 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.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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.

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:

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.

Non-adaptive 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 non-adaptive track without range resolution). The remaining available words continue to radiate in the variety appropriate to the target Doppler.

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,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

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.

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.

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 (PRF) 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.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix B

Functional specification of the "Pluto" emitter

This appendix contains a sanitized version of a textual intelligence report describing the functionality of an emitter called *"Pluto"*[1]. This description presents an example of a data-centric record within the Electronic Intelligence database that describes the functionality of this Multi-Function Radar (MFR).

General remarks

Radar transmit frequency is an important parameter of the radar functionality that may serve a purpose of uniquely identifying the radar system. However, as we mentioned in Chapter 1, the task of radar identification is outside the scope of this research effort. Therefore, we do not include radar transmit frequency characterization in this report.

The emitter timing 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.

Radar words

One of the basic sub-structures of this emitter signal is called a *word*. There are five varieties of word, denoted by w_0, w_1, w_2, w_3, and w_4. In addition, the radar employs a special termination character w_t.

[1]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 Defence radars, but has been altered and declassified before the release.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Words are arranged into phrases according to illustration in Fig. B.1. There are two kinds of words:

1. A radiated word (w_1, w_2, w_3, or w_4) consists of a fixed number of pulses with a constant Pulse Repetition Interval (PRI) pattern, followed by whatever gap is required to pad the word before the beginning of the next word or the termination character.

2. A blank word (w_0) is a period during which no pulses are radiated at all. In what follows, the "first pulse" of such a word should be treated as an imaginary time tag.

Termination character

A termination character is a non-trivial structure consisting of:

1. five (5) evenly spaced pulses with a PRI of approximately $3.5\mu s$,

2. a gap of approximately $7\mu s$,

3. eight (8) evenly spaced pulses with a PRI of approximately $3.5\mu s$,

4. a gap of approximately $10\mu s$,

5. twelve (12) evenly spaced pulses with a PRI of approximately $52\mu s$, and

6. any gap required to pad the termination character before the next word.

In all cases, a "gap" is measured from the leading edge of the last pulse in the preceding structure to the leading edge of the first pulse in the subsequent structure (See Fig. B.1).

Time Division Multiplexing – phrases and clauses

This system uses an electronically steered radar so that it can perform multiple tasks at once. These functions are multiplexed in the time domain within the structures called clauses and phrases. A *phrase* is a sequence of two consecutive words followed by a termination character (See Fig. B.1). The spacing between words is not uniform. The structure is broken down as follows:

1. The first word of the phrase appears with its first pulse on the first clock cycle after the preceding phrase.

2. The second word of the phrase appears with its first leading edge occurring a fixed number of cycles from the first leading edge of the first word.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

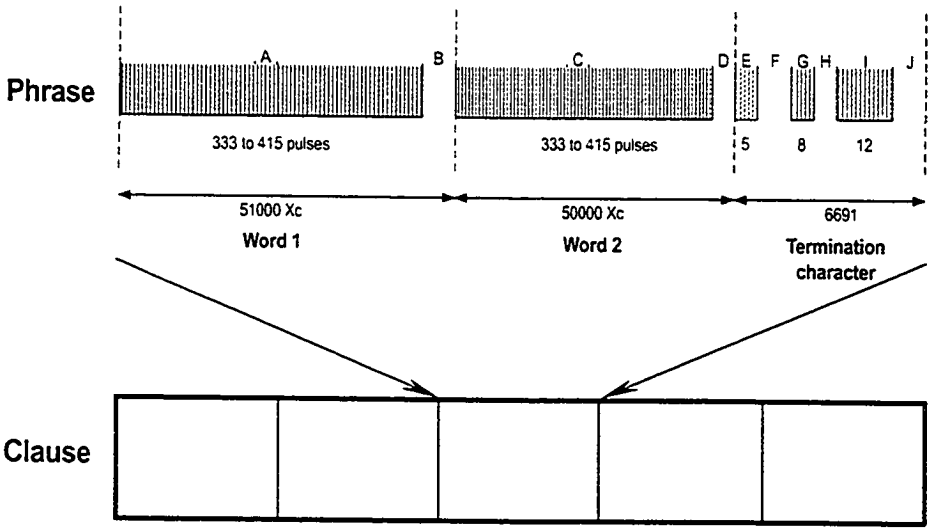

Figure B.1: Hierarchical, layered signal structure of the Pluto emitter (reproduced from Fig. 2.3 for convenience).
This emitter employs words, phrases, and clauses in its signal organization. A phrase is constructed using 2 words followed by a single termination character. Words within the phrase can be different, but the termination character is always the same. The words of this emitter are of different length, but the duration of the phrase is fixed (107.691 crystal clock counts (Xc), which is $\approx 10.23ms$). To accommodate for the differences in word length, dead time pads (regions B and D) of variable duration are employed. A termination character is a sequence of scheduled PPIs and consists of five distinct regions (E – J). Regions E, G, and I contain 5, 8, and 12 fixed PPI pulses, respectively. Regions F, H, and J are dead times of known length. This particular radar has five phrases within one clause which means that it can perform five different operations simultaneously. The output sequence of this radar is formed in a similar fashion to the sequence of the radar emitter considered earlier.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. A termination character appears with its first leading edge occurring a fixed number of clock cycles after the first leading edge of the second word.

A *clause* is a sequence of five consecutive phrases (See Fig. B.1).

Search-while-Track Scan

One of the generic operating states of the radar is a search scan denoted by $< Search >$. In the $< Search >$ scan mode, all of the words will be of type w_2.

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, $< Search >$ can also be multiplexed with target tracking functions. In the latter case, some of the words within each clause are occupied by adaptive target tracking and will not engage in search functions. In this case, all available words (those that are not occupied) are radiated as w_2. 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.

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 search to one of the acquisition patterns. All of the acquisition scans follow these steps in a sequence:

Non-adaptive track: The acquisition begins with a non-adaptive track phase. In this phase, all words in each phrase radiate w_2. The second word of one phrase will be designated for non-adaptive track of the target. This also radiates word W2 although it is associated with a distinguishable beam pattern.

Range resolution: Eventually the radar acquires the angular track resolution but still suffers from range ambiguities. The first word of each available phrase changes to w_0 (blank), the non-adaptive track word remains engaged as in the previous step, but all other available words cycle amongst words $w_1 - w_2 - w_3 - w_4$ (or any permutation thereof), in sequence on a word-by-word basis.

Return from Acquisition to Search: Finally the radar has established a track and is ready to terminate the acquisition scan. The non-adaptive tracking word becomes an adaptive tracking word that is used to track the target until the track is dropped. (Unlike the Mercury radar, only

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

one word is used for adaptive tracking). This word will not be available for search functions or further acquisitions. The radar then returns to the search-while-track functions. All occupied words maintain their tracks and all available words (possibly including the first word of the designated phrase) execute the appropriate scan pattern.

Only one acquisition can be performed at any given time.

Track maintenance

Each track is maintained only by the adaptive tracking word designated in the last step of acquisition. Those words are considered occupied and cannot participate in search or acquisition functions until the target track is dropped. The radar performs range tracking by adaptively changing amongst any of the four radiated words (w_1, w_2, w_3, or w_4) in order to avoid eclipsing and maintain their range gates.

Occasionally, the system may perform a range verification function on the track. This is the same behavior as the range resolution step described above.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix C

Event-driven pulse train analysis

In Chapter 9, we have introduced a Viterbi-like pulse train analysis algorithm that utilizes Hidden Markov Models (HMMs) as representations of Multi-Function Radar (MFR) words. The strength of the Viterbi algorithm for radar pulse train analysis described in Section 9.1 is in its ability to process highly structured pulse patterns. The algorithm does not even require a one-to-one mapping between the radar words and their pulse representations. Indeed, admissible HMM representations of the radar words may be much more complex than the simple topology shown in Figure 9.1 (a). As a result, this technique may accommodate the possibility that one radar word may have multiple realizations in the pulse domain.

For example, radar words shown in Fig. 6.1 (a) could have variable length of section B that could depend on the current Doppler-shift of the target. Likewise, lengths of sections A and C of the words in Fig. 6.1 (b) do not have to be fixed. In addition, the structure of the radar words can be dynamically varying as well. In these cases, the structure of the HMM word models presented in Section 6.2 will become more complicated, but the pulse train analysis algorithm will remain unchanged.

In the case when radar words are rigid entities whose pulse representations are unique and independent of the context in which they appear, an alternative probabilistic score can be utilized in the pulse train analysis procedure. This probabilistic score can be based on counting certain events that either have occurred or should have occurred. The radar word templates (6.10) can then be replaced by simpler, static templates, and the overall complexity of the pulse train analysis algorithm can be substantially reduced.

The weakness of the pulse train analysis algorithm presented in Section 9.1 is in the dependence of the number of states of the HMM radar word template M_k on the quantization resolution. In this chapter, we present an algorithm that is conceptually similar to Viterbi, but is much more computationally

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

efficient and does not depend on the value of T_{obs}. Unlike the Viterbi approach, this algorithm will require a rigid, one-to-one mapping between the radar words and their pulse representations, and does not apply in the case of dynamically varying word structure. For the large class of modern MFRs, this limitation is not really an issue, since most of them (including the ones discussed in Appendices A and B) employ static word-to-pulse mappings.

Static radar word templates

In Chapter 4, we defined the score of the observed sequence with respect to the HMM model as

$$P(\mathbf{O}|\lambda_k) = \sum_{i=1}^{M_k} \alpha_T(i)\,, \tag{C.1}$$

where $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \ldots\}$ is the given quantized sequence of observations, λ_k is the HMM model for the word template k defined by (6.10), and $\alpha_T(i)$ is the forward variable defined by (4.6).

For the case of static word-to-pulse mapping, we may actually find an alternative to the HMM λ_k model that is easier to use in the score evaluation procedure. We define the model ξ_k simply as a vector of pulse Time-Of-Arrivals (TOAs) of the word k

$$\xi_k = [t_1, t_2, \ldots, t_{N_k}]^T \tag{C.2}$$

where t_i is the TOA of the i^{th} pulse, and N_k is the total number of pulses in the k^{th} radar word. We refer to ξ_k as the *static radar word template* or simply *word template.*

Event-driven pulse train analysis algorithm

Given the new model (C.2), the new pulse sequence score can now be expressed as the total probability of a certain number of "events"

$$P(\mathbf{O}|\xi_k) = \prod_{i=1}^{N_e} p_{e_i} \tag{C.3}$$

where e_i is the i-th event with the probability p_{e_i}, and N_e is the total number of events contributing to the score.

There are four different types of events that contribute to the score (C.3). If there is a pulse in the quantized sequence that matches the expected pulse in the word template ξ_k, we declare this event an e_{match}. If the pulse in the sequence is "slightly" ahead of the position expected (we will define the concept "slightly" more precisely below), we declare it an e_{split}. If the sequence contains no pulse where it was expected, we call it an e_{miss}. Finally, if a pulse

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

was detected, but was not expected given the word template ξ_k, we declare it an e_{spur}. These are the probability measures associated with each of the events

$$\begin{cases} e_{match} & : \text{ with } p_{e_i} = (1-p_{spur})(1-p_{miss})(1-p_i) \\ e_{split} & : \text{ with } p_{e_i} = (1-p_{spur})(1-p_{miss})p_i \\ e_{miss} & : \text{ with } p_{e_i} = (1-p_{spur})p_{miss} \\ e_{spur} & : \text{ with } p_{e_i} = p_{spur} \end{cases} \tag{C.4}$$

where p_i is defined by (6.5). It is due to the fact that the computations of the score (C.4) happen at the points when certain events have either been recorded or expected, that we call this algorithm an *event-driven pulse train analysis.*

Final implementation issues can best be addressed by referring to Fig. C.1. Unlike the case of the HMM-based Viterbi algorithm of Section 9.1, we find that it is best to implement the event-driven scoring not in the quantized domain, but in the reconstructed domain. To reconstruct the quantized pulse sequence, we simply multiply every index $n_i(\varphi)$ from (6.3) by the T_{obs}. In general, due to the quantization distortion, the resulting quantity is not equal to the pulse TOA t_i defined in the template ξ_k

$$t_i' = n_i(\varphi)T_{obs} \neq t_i \tag{C.5}$$

According to (6.3), depending on the phase shift φ, any pulse can quantize to either n_i, or $n_i + 1$. Therefore, in the reconstructed domain, any pulse t_i will take one of the two possible values

$$t_i \Rightarrow \begin{cases} t'_{match} &= n_i T_{obs} \\ t'_{split} &= (n_i+1)\,T_{obs} \end{cases} \tag{C.6}$$

Fig. C.1 (a) provides an intuitive geometric interpretation of (C.6). For any pulse t_i, if the pulse in the reconstructed domain has the value $t' = t_i - \Delta\tau_i$, this corresponds to the event e_{match}. If, on the other hand, the reconstructed value $t' = t_i - \Delta\tau_i + T_{obs}$, this corresponds to the event e_{split}. Here

$$\Delta\tau_i = p_i T_{obs} \tag{C.7}$$

and p_i is defined by (6.5).

Fig. C.1 (b) shows an example of the event-driven scoring procedure. Clearly, the computational complexity of this algorithm is much lower than that of the Viterbi algorithm described in Section 9.1. It is proportional to $\mathcal{O}[N_e]$, where $N_e \geq N_k$ is the total number of registered and expected events.

We have performed pulse train analysis experiments similar to those described in Section 9.2. Results of these experiments are illustrated in Fig. C.2. The format of these results is analogous to Fig. 9.2. The results in both Fig. 9.2 and Fig. C.2 are qualitatively similar. The score curves behave differently due to the fact that they are computed differently, but the extracted radar word sequences match the expected sequences.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

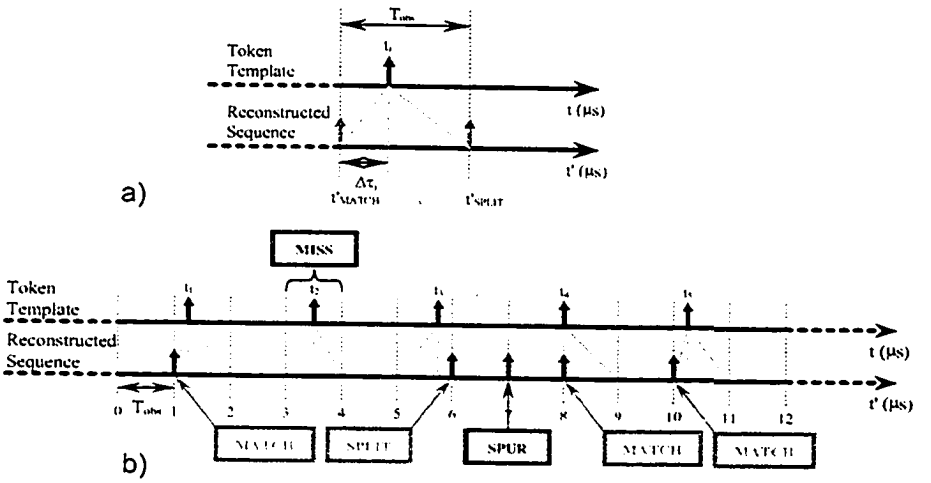

Figure C.1: Event-driven pulse train analysis principle. (a) provides geometric interpretation of the event-driven analysis algorithm. Reconstructed pulses are expected to appear in either the lower left or right vertices of the triangle. Otherwise the pulse should be declared missing. (b) illustrates the operation of the algorithm using the specific example of a reconstructed pulse sequence. The template is a constant PRI= $2.25\mu s$ pulse sequence of five pulses $\xi_k = [t_1, \ldots, t_5]^T$ with the first pulse $t_1 = 1.25\mu s$. $T_{obs} = 1\mu s$, and the reconstructed observed sequence also contains five pulses $O = [1, 6, 7, 8, 10]^T$. Six events should be considered in this example. The first pulse was found where it was expected, therefore it matched the template. The second pulse in the template could not be matched to any of the pulses in the reconstructed sequence, thus, a miss was declared. Pulse-splitting occurred for the third pulse. Then, the pulse in the sequence could not be matched with any of the pulses from ξ_k, so it was declared a spur. Two final pulses in the sequence matched the template as expected. The total score of this sequence can be calculated using (C.3) with event probabilities defined by (C.4).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

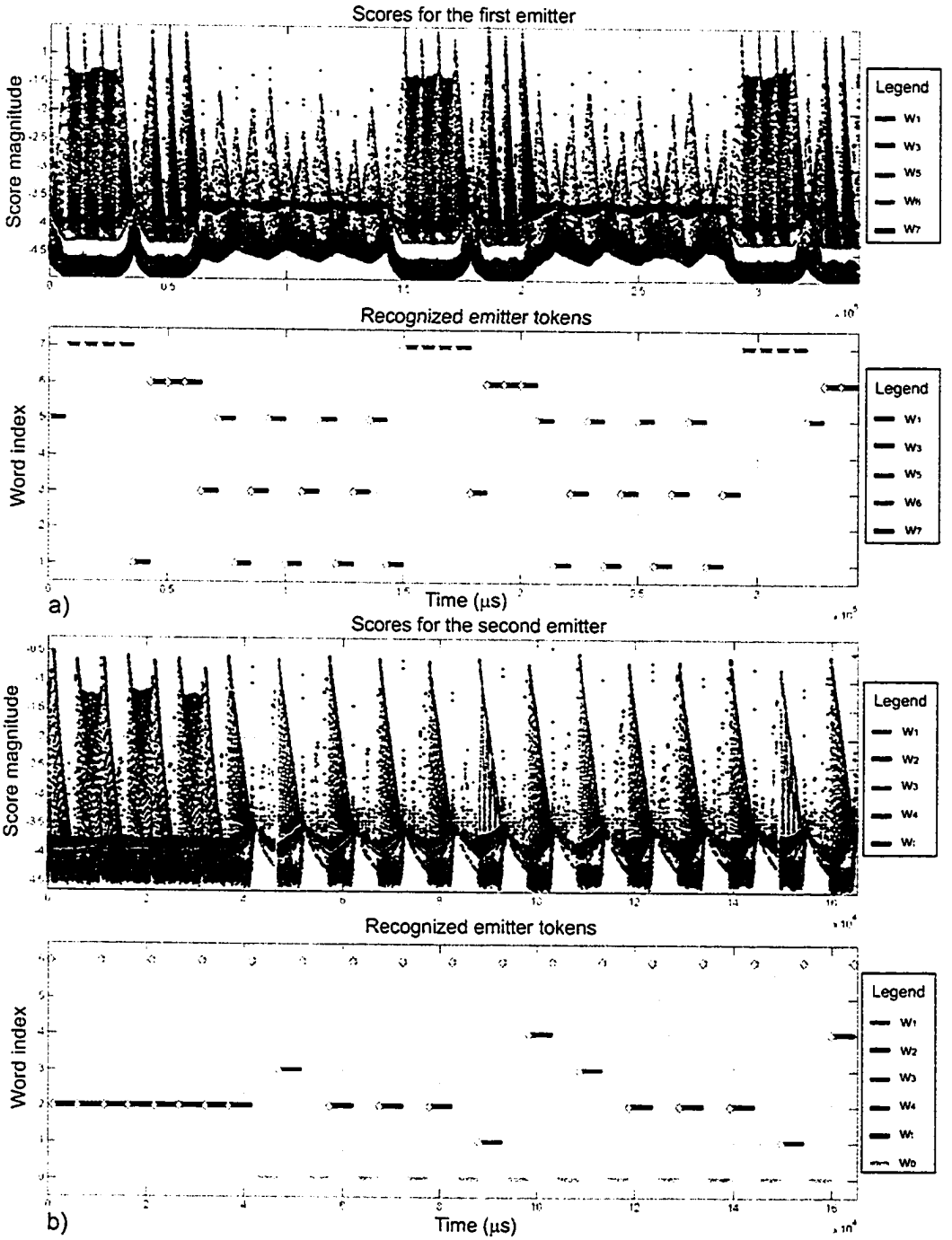

Figure C.2: Simulation results showing event-based scores and extracted word sequence for (a) the radar words in Fig. 6.1 (a), and (b) the radar words in Fig. 6.1 (b).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

Adamy, D. L. (2001). *EW 101: A First Course in Electronic Warfare.* Norwood, MA: Artech House.

Adamy, D. L. (2002). *Introduction to Electronic Warfare Modeling and Simulation.* Norwood, MA: Artech House.

Aho, A. V., R. Sethi, and J. D. Ullman (1986). *Compilers: principles, techniques, tools.* Reading, MA: Addison-Wesley.

Aho, A. V. and J. D. Ullman (1972). *The Theory of Parsing, Translation and Compiling*, Volume I: Parsing. Englewood Cliffs, NJ: Prentice-Hall.

Aho, A. V. and J. D. Ullman (1973). *The Theory of Parsing, Translation and Compiling*, Volume II: Compiling. Englewood Cliffs, NJ: Prentice-Hall.

Anselmo, M., D. Giammarresi, and S. Varricchio (2003, July). Finite automata and non-self-embedding grammars. In *7th International Conference on Implementation and Application of Automata (CIAA'02)*, Volume 2608 of *Lecture notes in Computer Science*, Tours, France, pp. 47–56.

Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp (2002, February). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans. Signal Processing 50*(2), 174–188.

Baldi, P. and S. Brunak (2001). *Bioinformatics: The Machine Learning Approach* (Second ed.). Cambridge, MA: MIT Press.

Bernardo, J. M. and A. F. M. Smith (2000). *Bayesian Theory.* New York, NY: John Wiley & Sons.

Bertsekas, D. P. (1996). *Neuro-Dynamic Programming* (First ed.). Cambridge, MA: Athena Scientific.

Bertsekas, D. P. (2001). *Dynamic Programming and Optimal Control* (Second ed.). Cambridge, MA: Athena Scientific.

Cassandras, C. G. and S. Lafortune (1999). *Introduction to Discrete Event Systems.* Boston, MA: Kluwer Academic.

Chomsky, N. (1956, September). Three models for the description of language. *IRE Trans. Inform. Theory 2*(3), 113–124.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chomsky, N. (1959a, December). A note on phrase structure grammars. *Inform. and Control 2*(4), 393–395.

Chomsky, N. (1959b, June). On certain formal properties of grammars. *Inform. and Control 2*(2), 137–167.

Chomsky, N. and G. A. Miller (1958, May). Finite state languages. *Inform. and Control 1*(2), 91–112.

Clarkson, I. V. L., J. E. Perkins, and I. M. Y. Mareels (1996, May). Number/theoretic solutions to intercept time problems. *IEEE Trans. Inform. Theory 42*(3), 959–971.

Cormen, T. H., C. E. Leiserson, and R. L. Rivest (1998). *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Crisan, D. and A. Doucet (2002, March). A survey of convergence results on particle filtering methods for practitioners. *IEEE Trans. Signal Processing 50*(3), 736–746.

Deller, J. R., J. H. L. Hansen, and J. G. Proakis (1999). *Discrete-Time Processing of Speech Signals*. Willey-IEEE.

Dilkes, F. (2004, November). Statistical cross-correlation algorithms for temporal pulse profiles. Technical Report DREO TM 2004-220, Defence Research & Development Canada, Ottawa, Canada.

Dilkes, F. (2005, January). Statistical decision thresholds for pulse template cross-correlators. Technical Report DREO TM 2005-XXX, Defence Research & Development Canada, Ottawa, Canada.

Dilkes, F. and N. Visnevski (2004, October). Non-self-embedding context-free grammars for electronic warfare. Technical Report DREO TM 2004-157, Defence Research & Development Canada, Ottawa, Canada.

Djuric, P. M., J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez (2003, September). Particle filtering. *IEEE Signal Processing Mag. 20*(5), 19–38.

Durbin, R., S. Eddy, A. Krogh, and G. Mitchison (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press.

Elliott, R. J., L. Aggoun, and J. B. Moore (1995). *Hidden Markov Models: Estimation and Control*. New York, NY: Springer Verlag.

Ephraim, Y. and N. Merhav (2002, June). Hidden markov processes. *IEEE Trans. Inform. Theory 48*(6), 1518–1569.

Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition*. New Yourk, NY: Academic Press.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fu, K. S. (1982). *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

Goos, G. and J. Hartmanis (Eds.) (1974). *L Systems*. Lecture Notes in Computer Science. New York, NY: Springer Verlag.

Griffiths, L. J. and C. W. Jim (1982, January). An alternative approach to linearly constraint adaptive beamforming. *IEEE Trans. Antennas Propagat. 30*(1), 27–34.

Haykin, S., N. Visnevski, F. Dilkes, V. Krishnamurthy, and P. Lavoie (in preparation). Stochastic context-free grammars for multi-function radars: Modeling, joint classification, and state-estimation. *Proc. IEEE*.

Haykin, S., N. Visnevski, F. Dilkes, V. Krishnamurthy, and P. Lavoie (to appear, Mar. 2005). Stochastic context-free grammars for multi-function radars: Modeling, joint classification, and state-estimation. In *Proc. Workshop on Defense Applications in Signal Processing (DASP'2005)*, Homestead, Utah, USA.

Hopcroft, J. E., R. Motwani, and J. D. Ullman (2001). *Introduction to Automata Theory, Languages, and Computation* (Second ed.). Reading, MA: Addison-Wesley.

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

Jaeger, H. (2000, June). Observable operator models for discrete stochastic time series. *Neural Computation 12*(6), 1371–1398.

Knuth, D. E. (1997). *The Art of Computer Programming* (Third ed.), Volume I: Fundamental Algorithms. Reading, MA: Addison-Wesley.

Kuich, W. and A. Salomaa (1986). *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science. New York, NY: Springer Verlag.

Lavoie, P. (2001, October). Hidden Markov modeling for radar electronic warfare. Technical Report DREO TM 2001-127, Defence Research & Development Canada, Ottawa, Canada.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development, i and ii. *J. Theoret. Biology 18*, 280–315.

Lindenmayer, A. (1971). Developmental systems without cellular interaction, their languages and grammars. *J. Theoret. Biology 30*, 455–484.

Lindenmayer, A. (1975). Developmental algorithms for multicellular organizms: A survey of l systems. *J. Theoret. Biology 54*, 3–22.

Matuszewski, J. and L. Paradowski (1998, May). The knowledge based approach for emitter identification. In *12th International Conference on*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Microwaves and Radar (MIKON'98), Volume 3, Krakov, Poland, pp. 810–814.

Moore, J. B. and V. Krishnamurthy (1994, November). Deinterleaving pulse trains using discrete-time stochastic dynamic-linear models. *IEEE Trans. Signal Processing 42*(11), 3092–3103.

Nederhof, M. (2000, January). *Regular Approximations of CFLs: A Grammatical View*, Chapter 12, pp. 221–241. Advances in Probabilistic and other Parsing Technologies. Boston, MA: Kluwer Academic Publishers.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Pothen, A. and C.-J. Fan (1990, December). Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software 16*(4), 303–324.

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

Rabiner, L. R. (1989, February). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. IEEE 77*(2), 257–286.

Ray, P. S. (1998, July). A novel pulse TOA analysis technique for radar identification. *IEEE Trans. Aerosp. Electron. Syst. 34*(3), 716–721.

Robert, C. P. (2001). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation* (Second ed.). New York, NY: Springer Verlag.

Rozenberg, G. and A. Salomaa (1980). *The Mathematical Theory of L Systems*. New York, NY: Academic Press.

Salomaa, A. (1969). *Theory of Automata*. Elmsford, NY: Pergamon Press, Ltd.

Salomaa, A. (1973). *Formal Languages*. New York, NY: Academic Press.

Salomaa, A. (1981). *Jewels of Formal Language Theory*. Rockville, MD: Computer Science Press.

Salomaa, A. (1985). *Computation and Automata*. Cambridge, UK: Cambridge University Press.

Schleher, D. C. (1999). *Electronic Warfare in the Information Age*. Norwood, MA: Artech House.

Schlesinger, R. J. (1986). *Principles of Electronic Warfare*. Los Altos, CA: Peninsula Publishing.

Skolnik, M. I. (Ed.) (1990). *Radar Handbook* (Second ed.). New York, NY: McGraw-Hill.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Skolnik, M. I. (2002). *Introduction to Radar Systems* (Third ed.). New York, NY: McGraw-Hill.

Tuttlebee, W. H. W. (Ed.) (2002a). *Software Defined Radio: Enabling Technologies*. New York, NY: John Wiley & Sons.

Tuttlebee, W. H. W. (Ed.) (2002b). *Software Defined Radio: Origins, Drivers and International Perspectives*. New York, NY: John Wiley & Sons.

Vakin, S. A., L. N. Shustov, and R. H. Dunwell (2001). *Fundamentals of Electronic Warfare*. Norwood, MA: Artech House.

Visnevski, N., F. Dilkes, S. Haykin, and V. Krishnamurthy (2005, May). Non-self-embedding context-free grammars for multi-function radar modeling – electronic warfare application. In *Proc. IEEE International Radar Conference (IRC'05)*, Arlington, VA, USA, pp. 669–674.

Visnevski, N., F. Dilkes, V. Krishnamurthy, S. Haykin, and P. Lavoie (in preparation). A special case of the radar pulse train analysis algorithm for static radar signal templates. *IEEE Trans. Aerosp. Electron. Syst.*.

Visnevski, N., F. Dilkes, P. Lavoie, V. Krishnamurthy, and S. Haykin (submitted, Dec. 2004). Hidden markov models for radar pulse train analysis in electronic support systems. *IEEE Trans. Aerosp. Electron. Syst.*.

Visnevski, N., S. Haykin, F. Dilkes, V. Krishnamurthy, and P. Lavoie (to appear, Jan. 2006). Syntactic modeling of multi-function radars. *IEEE Signal Processing Mag.*.

Visnevski, N., S. Haykin, V. Krishnamurthy, F. Dilkes, and P. Lavoie (2005, March). Hidden markov models for radar pulse train analysis in electronic warfare. In *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, Philadelphia, PA, USA, pp. V597–V600.

Visnevski, N., V. Krishnamurthy, S. Haykin, B. Currie, F. Dilkes, and P. Lavoie (2003, December). Multi-function radar emitter modelling: A stochastic discrete event system approach. In *Proc. IEEE Conference on Decision and Control (CDC'03)*, Maui, Hawaii, USA, pp. 6295–6300.

Viterbi, A. (1967, April). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Inform. Theory 13*(2), 260–269.

West, M. and J. Harrison (1997). *Bayesian Forecasting and Dynamic Models* (Second ed.). New York, NY: Springer Verlag.

Wiley, R. G. (1985). *Electronic Intelligence: The Interception of Radar Signals*. Norwood, MA: Artech House.

Wiley, R. G. (1993). *Electronic Intelligence: The Analysis of Radar Signals* (Second ed.). Norwood, MA: Artech House.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Zhu, W. and J. Garcia-Frias (2002, May). Modeling of bursty channels using stochastic context-free grammars. In *Proceedings of the 55th Vehicular Technology Conference (VTC 2002)*, Volume 1, Birmingham, AL, USA, pp. 355–359.

Zhu, W. and J. Garcia-Frias (2004, May). Stochastic context-free grammars and Hidden Markov Models for modeling of bursty channels. *IEEE Trans. Veh. Technol. 53*(3), 666–676.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Nomenclature

$\alpha_t(i)$ Forward variable, see equation (4.6), page 41

$\Delta\tau_i$ TOA shift of the i^{th} pulse in the reconstructed domain, see equation (C.7), page 131

δ Transition function of the FSA, see equation (3.6), page 28

$\delta_t(i)$ Viterbi score, see equation (4.12), page 43

Γ Set of syntactic production rules, see equation (3.4), page 25

γ Markov chain, see equation (4.1), page 37

Λ Finite State Automaton, see equation (3.6), page 28

λ Hidden Markov Model, see equation (4.4), page 39

$\mathbf{A}$ Transition probability matrix, see equation (4.1), page 37

$\mathbf{B}$ Observation probability matrix, see equation (4.4), page 39

$\mathrm{M}(G)$ Transition matrix of the production graph of the grammar G, see equation (3.10), page 32

$\mathrm{M}^{\leq N}(G)$ Steady-state matrix of the production graph, see equation (5.1), page 49

$\mathcal{A}$ Alphabet, see equation (3.3), page 24

$\mathcal{C}$ Conceptual basis of a model, see equation (1.1), page 7

$\mathcal{E}$ Set of nonterminal symbols, see equation (3.3), page 24

$\mathcal{F}$ Model form, see equation (1.1), page 7

$\mathcal{M}$ Model structure, see equation (1.1), page 7

$\mathcal{Q}$ Quantitative basis of a model, see equation (1.1), page 7

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

π Initial Markov chain state distribution, see equation (4.1), page 37

ρ Spurious pulse density, see equation (6.9), page 65

Σ Set of input symbols of the FSA, see equation (3.6), page 28

φ Quantization phase, see equation (6.3), page 62

ξ_k Static radar word template, see equation (C.2), page 130

c_t Forward variable scaling factor, see equation (4.10), page 42

F Set of final states of the FSA, see equation (3.6), page 28

G Grammar, see equation (3.3), page 24

$n_i(\varphi)$ Pulse quantization index, see equation (6.3), page 62

$P(G)$ Production graph of a CFG, see equation (3.8), page 31

p_{miss} Probability of missed pulse observation, see equation (6.9), page 65

p_{spur} Probability of observing a spurious pulse, see equation (6.9), page 65

p_i Pulse splitting probability, see equation (6.5), page 63

$P_l(G)$ Labeled production graph of a CFG, see equation (3.8), page 31

P_s Set of probability distributions over the syntactic production rules, see equation (3.14), page 33

P_w Set of weight coefficients of the syntactic production rules, see equation (3.12), page 32

Q Set of states of the FSA, see equation (3.6), page 28

t'_i TOA of the i^{th} pulse in the reconstructed domain, see equation (C.5), page 131

T_{obs} Observer clock period of the receiver, see equation (6.3), page 62

t_i TOA of the i^{th} pulse, see equation (6.3), page 62

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Index

algorithm
    dynamic programming, 42
    maximum likelihood, 42, 107
    Viterbi, 42, 107
alphabet, 9, 24, 47

bioinformatics, 11
blind inference, 12
breadth-first algorithm, 55

channel
    binary, 61, 64
    binary erasure, 61
    erasure, 83
    Markov-modulated, 61
    pulse-propagation, 61
clause, 17
closure
    Kleene, 24
    positive, 24
    reflexive transitive, 24
    transitive, 24
complexity
    computational, 42, 107
    model, 8
component
    acyclic, 69
    cyclic, 69
    strongly-connected, 52, 87, 98
composition, 51, 55, 57
concatenation, 24

de-interleaving, 60, 61, 106
decomposition
    Dulmage-Mendelsohn, 52, 87, 98
    grammatical, 48, 51, 52
depth-first algorithm, 55
design, 72, 73
Discrete Event System, 9–11, 115
divide and conquer, 2, 11, 16, 17
Dulmage-Mendelsohn decomposition, 52, 87, 98
dynamic programming, 42

Electronic Intelligence, 2
    data-centric, 1, 5, 11
    model-centric, 2, 8, 11, 69, 114
Electronic Threat Library, 2
    data-centric, 3, 5
    model-centric, 8, 67, 69
Electronic Warfare, 2
entropy, 10

finite-state automaton, 23, 27–31, 34, 35, 38, 48, 53–55, 63, 68, 71, 77, 87, 103, 116
forward algorithm, 41
forward variable, 41
functional state, 78

genomics, 11, 66
grammar, 9, 22, 24
    characteristic, 33, 77, 83, 91, 98
    context-free, 26, 29, 47, 51
    context-sensitive, 26
    deterministic, 77, 78, 81, 91
    finite-state, 26
    generative, 22
    indexed, 117
    non-self-embedding, 30, 51, 68
    programmed, 117
    regular, 26, 27

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

self-embedding, 30
stochastic, 33, 77, 85, 91, 98
transformational, 22, 24
type-0, 26
unrestricted, 26
weighted, 32, 85
grammatical decomposition, 48, 51, 52

Hidden Markov Model, 34, 37, 39, 59, 63, 65, 107
ergodic, 41
left-to-right, 40, 65
pulse-level, 65

implementation, 72, 73

L-systems, 36
language, 9, 22, 24
context-free, 27
context-sensitive, 27
finite-state, 27, 47
regular, 27, 28
stochastic, 32
type-0, 27
unrestricted, 27
language processing, 10
lexical analysis, 10
lexicon, 9
Lindenmayer systems, 36

Markov chain, 34, 37
left-to-right, 64
pulse interval, 64
maximum likelihood estimation, 42, 107
Mealy machine, 39
Mercury emitter, 77, 118
missing pulse, 106
mode, 4
model
phrase-level, 21
pulse-level, 14, 21, 59, 65
syntactic, 9, 47, 77
word-level, 14, 21, 47, 59, 77
model complexity, 8
Moore machine, 39

Observable Operator Model, 45

palindrome, 29
parsing, 22
Phased Array Antenna, 3, 5
phrase, 9, 17
phrase-level model, 21
Pluto emitter, 77, 124
Poisson process, 61
problem
$\mathcal{NP}$, 42
$\mathcal{NP}$-complete, 42
$\mathcal{P}$, 42
decidable, 42
intractable, 42
tractable, 42
undecidable, 42
production graph, 31, 87, 98
labeled, 31
production rule, 25
pulse interval, 63
pulse jitter, 66
pulse repetition interval, 16
pulse splitting, 63
pulse to pulse interval, 3, 5, 8, 17
pulse train analysis, 59, 66, 106
pulse-level model, 14, 21

quantization, 61
quantization index, 62

radar
low probability of intercept, 5
multi-function, 3, 5, 8, 13
regular expression, 27, 28, 59, 63
regular set, 27
requirements, 72, 73
ring, 49

semi-ring, 49, 50
sentence, 9
simulation, 7

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Software-Defined Radio, 3
specification, 72, 73
spurious pulse, 106
steady-state matrix, 49, 87, 98
structural synthesis, 12
synchronization, 61
syntactic model, 9, 47
syntax, 9
syntax analysis, 10, 11, 22, 115

time of arrival, 62
time-division multiplexing, 3, 8
transition matrix, 32, 87, 98
trellis, 108

Viterbi algorithm, 42, 43, 107
vocabulary, 9, 47

word, 9, 17, 47
word-level model, 14, 21, 47

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.