The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

In this column we review the following books.

1. **Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenge** Edited by Michael H. Goldwasser, David S. Johnson, Catherine C. McGeoch. Reviewed by Maulik A. Dave. This book reports on a the DIMACS implementation where competitors really coded up different algorithms for the problems indicated.

2. **Genomic Perl: From Bioinformatics Basics to Working Code** by Rex A. Dwyer. Review by Raymond Wan. This book covers several topics in bioinformatics in the context of the Perl language.

3. **Graphs, Networks, and Algorithms** by Dieter Jungnickel. Review by William Fahle. This book starts from the basics but is fairly comprehensive.

4. **Immunocomputing: Principles and Applications** by Alexander O. Tarakanov, Victor A. Skormin, Svetlana P. Sokolova. Reviewed by Wenzhong Zhao. This book considers the bodies immune system as a computing agent.

5. **Term Rewriting Systems** by Terese. Reviewed by Frederic Loulergue. This book is meant to be good for both teaching and doing research on Term Rewriting Systems.

Books I want Reviewed

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu

Reviews need to be in LaTeX, LaTeX2e, or Plaintext.

Books on Algorithms

1. *Graphs, Algorithms, and Optimization* by Kocay and Kreher.

2. *Combinatorial Optimization: Packing and Covering* by Cornuejols.

3. *Algorithms: Design Techniques and Analysis* by Alsuwaiyel.

4. *Design and Analysis of Randomized Algorithms* by Hromkovic.

5. Combinatorial and Computational Geometry (MSRI) edited by Goodman, Pach, and Welzl.

Books on Cryptography

1. *Elliptic Curves: Number Theory and Cryptography* by Larry Washington.

2. *Coding for Data and Computer Communication* by Salomon.

3. *Crytography, Information Theory, and Error Correction: A Handbook for the 21st Century* By Bruen and Forcinito.

4. *Handbook of Elliptic and Hyperelliptic Curve Cryptography* by Cohen and Frey.

Books on Coding Theory

1. *Error Correction Coding: Mathematical Methods and Algorithms* by Moon.

2. *Introduction to Coding Theory* by van Lint.

3. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.

Misc Books

1. *Computer Viruses: from theory to application* by Filiol.

2. *Applied Combinatorics on Words* by Lothaire.

3. *Rippling: Meta-Level Guidance for Mathematical Reasoning* by Bundy, Basin, Hutter, Ireland.

4. *Domain Decomposition Methods– Algorithms and Theory* by Toseli and Widlund.

5. *Semantic Integration of Heterogenous Software Specifications* by Martin Groβe-Rhode.

6. *Handbook of Computational Methods of Integration* by Kyther and Schaferkotter.

7. *Theoretical and Experimental DNA Computatoin* by Amos.

8. *Polynomials* by Prasolov

Review of
Data Structures, Near Neighbor Searches, and Methodology:
Fifth and Sixth DIMACS Implementation Challenges[2]
Series: DIMACS Series in Discrete Mathand TCS#59
Editors: Michael H. Goldwasser, David S. Johnson, Catherine C. McGeoch
Publisher: American Mathematical Society, 2002

Reviewer: Maulik A. Dave

1 Overview

This book is in the area of searching data structures, and algorithms. It also has a discussion on experimental analysis of algorithms. The goal of DIMACS Implementation Challenges is projected as to promote top quality experimental research on algorithms, and data structures. The fifth challenge (1995-1996) was on dictionaries, and priority queues. The first part of the book is from the works in the fifth challenge. The sixth challenge (1998) was on near neighbor searching. The second part of the book is from the works in the sixth challenge. The third part of the book has general discussion articles on algorithm experiments.

[2]

2 Summary of Contents

The book is divided into three parts.

2.1 Part 1. Dictionaries and Priority Queues

This part starts with a work on tabu searching. For searching with combinatorial explosions, local searching is guided by history sensitive schemes. The paper introduces, and formulates the tabu searching in first two sections. The paper proposes to use dictionaries supporting the history sensitive heuristics. The persistent red-black trees, in particular, is proposed. The asymptotic analysis of the algorithm is discussed in details in the third section. The claim is that the proposed scheme improves both time, and space efficiency in comparison to other similar schemes. The section 4 contains description, and results of three sets of experiments with the proposed scheme. The first set of experiments consists of memory usage tests using random numeric keys. 27 dictionary trace files are used. The corresponding graphs are presented. The second set of experiments consists of memory usage tests for a reactive search application. The memory usage results for different instances of the Maximum Clique problem are illustrated by graphs. The third set of experiments consists of timing tests on the maximum clique problems. Comparative results on CPU times for ephemeral and persistent version on maximum clique problems are presented. The section 5 briefly concludes.

The next paper is on perfect hashing. Modifications of the FKS algorithm [1] by Fredman, Komlos, and Szemerdi is proposed. The tests are run on an alpha based machine, an RS/6000 based machine, and an x86 based machine. The key sizes considered, are varying from 4 bytes to 36 bytes; most of them being 16/20 bytes. By graphical illustrations, the improvements over FKS algorithm with respect to both space, and running time are shown. The comparison for different representations for space arrays is reported separately. Apart from these, some more space time trade off parameters are discussed, and their corresponding test results are reported. The paper ends with comparison of FKS algorithm with hashing with linear probing, and an implementation of associative arrays in Perl 5.

The last paper of the part is on heap-on-top (Hot) queues. The first two sections introduce the concepts of priority queues, monotone priority queues, and hot queues. The third section describes the multi level buckets theoretically. The complexity for operations, i.e., insertion, decrease-key, and extract-min are presented. This is followed by theoretical description of hot queues in forth section. The rest of the paper deals with experiments. The experiments compare six implementations of monotone priority queues : k-ary heaps with k=4; 2- and 3-level buckets; and 2- and 3- level hot queues; and a simple heap based algorithm. The machine used is Pentium Pro based running solaris. The test suite includes three different families of graph based problems, sorting tests, and nine families of event simulation. The timing results are presented using tables, and graphs.

2.2 Part 2. Near Neighbor Searching

This part starts with an experimental research paper in the area of data compression. Vector quantization (VQ) is a data compression method, where the near neighbor searching plays an important role. The first two sections introduce the VQ, and the associated codebook, and codewords. The third section discusses in details the various algorithms of near neighbor search chosen for the experiments. The algorithms are Orchard method, annulus method, double annulus method, k-d tree search, and PCP tree search. The experiments are done on an alpha processor based machine running linux operating system. The input consists of up to 7 images from an image database. The

results of the experiments are presented with two kinds of graphs, one plotting codebook size with codewords searched, and other one plotting codebook size with execution times. After describing experiments in section 4, an algorithm combining PCP, and k-d tree searches is discussed. The summary in section 6 contains qualitative conclusions drawn from the experimental results.

The next paper is on evaluation of disk based data structures. The multimedia applications use disk based data structures, where near neighbor searching plays an important role. After introduction, the paper compares the main memory data structures with disk based data structures. The R - tree, and its variants are discussed in details. The variants discussed are SS - tree, SR - tree, and VAMSplit R - tree. The discussion includes the tree construction algorithms. The experiments are done on machine with 360 MHz Ultrasparc II CPU, 512 MB main memory, and solaris operating system. The inputs are from photo and video archives of NASA. The performances for similarity in retrieval of images, and video scenes are presented by graphs. Graphs for R - tree variants are plotted separately. Further, k - d tree is compared with R - tree variants in details.

The next paper is on clustered point sets. The problem being addressed is of constructing a data structure for n points in d dimensions such that nearest data point for the query point can be reported efficiently. In presence of clustered point sets, splitting methods play important roles. The problem is discussed in details before discussing splitting methods. The splitting methods, namely, sliding midpoint, and minimum ambiguity are explained. The experiments consist of k - d tree implementations with these splitting methods. 4000 data points in dimension 20 with 12000 queries are used for the experiments. The distributions of data points studied are uniform, gaussian clustered, orthogonal ellipsoid clustered, and ellipsoid clustered. The number of nodes visited are plotted for each splitting method for various distributions. Further, construction times for trees are also plotted.

The next paper is on using the extended general space filling curves heuristic (SC) for approximate nearest neighbor search. The paper first presents a survey of various algorithms for near neighbor search. The survey includes tables showing complexities of various algorithms. The discussion on SC contains formal definition of space filling curves, algorithms for transformation functions, a small survey of applications of space filling curves, and explanation of theories of proposed extension. The approximate nearest neighbor search is presented as an application of SC. The data sets used in the experiments, have two kinds of distributions, namely, uniform, and normal bimodal. Experiments are done by varying transformations, number of prototypes, and number of dimensions. Apart from these variations, the sets of experiments also include real data sets experiments, and constant precision experiments.

The last paper in this part is a research paper on nearest neighbor search for euclidean hypercube with uniform distributions. After a brief survey of related works, the proposed algorithm is described. The analysis of the proposed algorithm is discussed in details. This is followed by presentation of experimental results. The major achievement of the proposed algorithm is that the search complexity is very nearly dimension invariant. Similar performance graph for k - d tree shows that search complexity varies with dimensions.

2.3 Part 3. Experimental Analysis of Algorithms

The first article contains a general discussion on the role of experiments in the algorithms. The study of algorithms is classified into two approaches : analytical, and empirical. The empirical approach can play a complementary role for analytical approach. The article also contains a discussion on what is a good or a bad empirical work.

The next paper proclaims experimental algorithmics as a discipline. After the introduction,

motivation for such a discipline is described. The modes of empirical assessment is classifies into seven non exclusive categories. The assessment of competing algorithms, and assessment of heuristics are described at length. The discussion also includes various other features of the discipline. The experimental set up, the measurement parameters, the presentation of data, and the analysis of the data are also discussed. Algorithms for constructing a minimum spanning tree is presented an illustration.

The next paper is a guide to the experimental analysis of algorithms. The guide is based on experience of the author over the course of more than a decade of experiments. The paper contains a full scale discussion on ten principles governing the writing of experimental papers. The principles include performing newsworthy experiments, connecting the paper to the literature, using instance test beds supporting general conclusions, using efficient experimental designs, and implementation; ensuring reproducibility, and comparability; reporting and presenting data; and drawing conclusions. The appendix of the paper lists 35 pet peeves, 6 pitfalls, and 8 suggestions.

The part ends with a bibliography of algorithm experimentation spanning 5 pages.

3 Style

The book is a collection of research papers, evaluation papers, general discussion papers, and a bibliography. The research papers typically devote half of the space for theoretical formulations, and other half for experiments. The theoretical formulations contain explanation for data structures, and algorithms in pseudo codes; apart from definitions, lemmas, and analysis of algorithms. The description of experiments contain experimental set ups, experimental results, and analysis of the results. The experimental results are presented by graphs, and tables. The discussion papers are descriptive in nature.

4 Opinion

The book contains a useful research work in the area of searching data structures. A reader new to the area of searching can easily understand the works. The k - d trees, some or other form, are referred by almost all the papers from first two parts. The software practitioners dealing with searching algorithms can find the book useful. The papers contain surveys of related works beneficial for researchers, and practitioners working in the area. The bibliography at the end, has a useful list of research works. The beginners in experimental algorithm designs can find the last part of the book very useful.

5 References

[1] M. Fredman, J. Komlos, and E. Szemerdi. Storing a parse table with O(1) worst case access time. *Journal of ACM*, 31(3); 538-544, july, 1984.

Review of **Genomic Perl**[3]
From Bioinformatics Basics to Working Code
Rex A. Dwyer
Publisher: Cambridge University Press, 2003
ISBN: 0-521-80177-X

Review by:
Raymond Wan (*rwan@cs.mu.oz.au*
University of Melbourne, Australia

1 Overview

Genomic Perl by Rex A. Dwyer covers several topics in bioinformatics in the context of the Perl language. According to the preface, the book selects some topics in bioinformatics, and presents them to the intended audience – a student in an "upper year level undergraduate or graduate level course".

Overall, I found the book interesting to read, but I believe it should be read with other resources available for reference.

2 Summary of Contents

The book consists of 17 chapters and appendices. While certain chapters build on the work from previous ones, each is self-contained and follows a fixed formula. The first half of a chapter provides the background in biochemistry that is essential to the chapter. Then, a Perl program is built up during the remainder of the chapter which addresses the problem, with difficult parts explained in the text. Every chapter has one to around ten questions for the student, with no solutions provided. The difficulty of the questions range from drawing a suffix tree for a word to extending the functionality of the Perl program from that chapter. Every chapter concludes with a bibliography. A brief summary of each chapter is as follows.

1. **The Central Dogma.** The book begins with an introduction to biochemistry, with emphasis on DNA, RNA, proteins, and the relationships between them. A Perl program that outputs the proteins that are encoded by a given DNA sequence is developed.

2. **RNA Secondary Structure.** An RNA molecule physically exists in three dimensional space, but is described at various levels. The nucleotides of an RNA sequence are written down as a sequence of characters at the lowest level. At the next level (also called its secondary structure), the molecule folds so that pairs of nonadjacent nucleotides form hydrogen bonds. Two programs examine how the sequence folds, while demonstrating recursion and dynamic programming.

3. **Comparing DNA Sequences.** The alignment of pairs of DNA sequences in the presence of errors is an example of approximate string-matching problems. A Perl program is given which addresses global alignment using the Needleman-Wunsch algorithm, another example of dynamic programming.

[3]Raymond Wan, ©2005

4. **Predicting Species: Statistical Models.** Probability, information theory, and entropy form the basis for a simple program which predicts the probability a DNA strand came from a particular species.

5. **Substitution Matrices for Amino Acids.** Substitution matrices extend the alignment techniques of Chapter 3 for proteins which differ due to evolution. The design of a Perl program which computes substitution matrices is shown.

6. **Sequence Databases.** Biological sequences are stored in public databases in several formats. A program using Perl's limited object-oriented facilities is developed which reads and processes the GenBank data format.

7. **Local Alignment and the BLAST Heuristic.** Chapter 3 describes an algorithm for aligning two entire sequences. In contrast, "local alignment" refers to the alignment of substrings of two sequences. The Smith-Waterman algorithm and BLAST heuristic are implemented in Perl for this purpose.

8. **Statistics of BLAST Database Searches.** The previous chapter is extended by looking at the statistics associated with aligning using the BLAST heuristic.

9. **Multiple Sequence Alignment I.** Another extension to Chapter 3 looks at the problem of aligning more than two sequences at a time by first extending the Needleman-Wunsch algorithm and then investigating incremental strategies, with a brief excursion into NP-completeness.

10. **Multiple Sequence Alignment II.** The discussion that began in the previous chapter continues by looking at how the efficiency of the algorithms for multiple sequence alignment can be improved.

11. **Phylogeny Reconstruction.** Phylogenies are trees which depict the course of evolution of several species. A Perl program is developed which creates a tree from a list of protein sequences.

12. **Protein Motifs and PROSITE.** Protein motifs are short sequences, each with a known biochemical function. PROSITE is a database of these motifs. Perl programs are implemented which first process motifs in the PROSITE file format, and then build a suffix tree so that all entries in the database can be efficiently compared with a sequence.

13. **Fragment Assembly.** Fragment assembly is an example of determining the shortest common superstring. That is, given a list of substrings representing fragments of a DNA sequence, what is the longest string that contains every substring in the list? A simplified version of the PHRAP program for addressing this problem is created in this chapter.

14. **Coding Sequence Prediction with Dicodons.** Statistical methods can be employed for finding new genes in a DNA sequence when a training set of known genes is available.

15. **Satellite Identification.** This chapter shows how identifying satellites, or tandem repeats in DNA, can be done in Perl. Both DNA fingerprinting and the reconstructing of phylogenies can benefit from these techniques.

16. **Restriction Mapping.** The restriction map of a DNA sequence is a list of locations where a restriction enzyme is known to cut the sequence. Creating a restriction map is one way of analysing a long sequence of DNA. Two techniques for achieving this in Perl are given, one which assumes the data set to be perfect, and another which allows for imprecise data.

17. **Rearranging Genomes: Gates and Hurdles.** One DNA sequence may contain the same genes as another, but in a different order. The number of steps required to transform the first sequence to the second can be used to determine how they are related, with respect to evolution.

18. **Appendices.** The appendices present a program for drawing the two dimensional RNA secondary structure diagrams (Chapter 2), some ideas for reducing space usage during the aligning of sequences, and a Perl solution to the *disjoint sets problem*.

3 Opinion

I approached this book with a few years of experience with Perl, and only limited knowledge of biochemistry. There were several aspects about the book which I liked. First, the structure of each chapter was simple and easy to follow. Second, most chapters are short and can cover the topic in just over 10 pages. That is, rather than going into detail with only a few topics, someone reading this book is given a taste of 17 topics. Third, while the bibliography of each chapter is short, only the most relevant citations are given.

While the theme of this book is bioinformatics, readers of this column will be interested that some time is spent on the underlying algorithms. Dynamic programming, recursion, recurrence relations, NP-completeness, regular expressions, and suffix trees are sprinkled throughout the book. The time and space complexity of algorithms are also occasionally given.

However, there are some parts of the book that did not appeal to me. Sometimes, the problem descriptions are too brief, and I often found myself asking questions whose answers were not available, even though they were not crucial. For example, in the first chapter, several definitions form the foundation of the introduction to biochemistry. But, due to the limited amount of space, a clear picture of how DNA, RNA, and proteins relate to each other is difficult to form. While a complete understanding of biochemistry is not needed to code a solution in Perl, the short background given in each chapter may cause some readers to yearn for more. Likewise, if someone does not have sufficient knowledge of Perl, then the latter half of each chapter may be hard to follow. The author has decided to spread coverage of Perl syntax throughout the book, rather than dedicating an introductory chapter or appendix to it. This choice almost forces readers to go through the book in order. Finally, the absence of pseudocode makes it difficult to generalise the solution to other languages. While Perl is used often by bioinformaticians, Perl source code is sometimes difficult to read, and text which explain sections of a program can easily become repetitive and uneventful.

Most of these shortcomings, though, are expected in a book such as this. Both biochemistry and computer science are vast fields which can only be fully understood by books dedicated to them. In fact, bioinformatics is a difficult field to write for since it juggles several seemingly disjoint areas including computer science, mathematics, and biochemistry. And, as the goal of this book is to explain bioinformatics using Perl as a framework, the reliance on Perl is unavoidable.

Because of this, the reader may need several other resources in order to follow this book. While *Genomic Perl* is suitable for an upper year level undergraduate course and above, a lecturer or book is necessary to fill in any missing gaps. These gaps include biochemistry, mathematics, Perl,

and algorithms, depending on the individual. That is, reading this book in isolation may prove difficult.

In conclusion, I am pleased to say that this book is a rare example of when one *can* judge a book by its cover! The title "Genomic Perl: From Bioinformatics Basics to Working Code" summarises the structure chosen by the author. Each chapter commences with only the rudimentary basics for each topic, and concludes with working Perl code, which is also included in the accompanying CD ROM. Excluding students, others with an adequate knowledge of bioinformatics may benefit from this book since the accompanying Perl source code can be easily extended and deployed.

Review of
Graphs, Networks, and Algorithms[4]
Second Edition
Author: Dieter Jungnickel
Springer-Verlag Berlin Heidelberg 2005
Review written by William Fahle

0.1 Overview

The field of graph theory with the accompanying important problems such as flow and matching is maturing and growing. There is a need for a solid treatment of these subjects which is both introductory and comprehensive. This book seems to fit that bill, beginning from the very basic definitions of graph theory, quickly building a catalog of theorems, and ending with a complex suite of algorithms on graphs and networks.

The early focus is mainly on graph theory, but the book includes discussions of flow, matroids, matching and paths to complete a course in combinatorial optimization. Applications are given for most of these subjects. At the end is a collection of NP-complete problems and an extensive bibliography.

This text is suitable for graduate courses in combinatorics and graph theory, as well as for independent study and research by students, mathematicians, and professionals. It is a welcome addition to the library of choices of textbooks for these subjects.

0.2 Graph Theory

An introduction leads us quickly into the origins of graph theory and the characteristics which distinguish it from other fields of mathematics. Rigorous mathematical definitions are given for graphs, networks, paths, walks, and so on. From these grow an array of small theorems, each proven from the definitions given. The history of Euler, Hamilton, and the birth of graph theory are covered well. Chapter One ends with a few applications in event scheduling.

The next two chapters introduce some computer science and complexity theory for readers without this background. However, most readers would do well to skip none of this section; many of the definitions and early proofs set the stage for later work. The end of chapter four gets back into the territory which will be less familiar to the average computer scientist, including Steiner trees and arborescences.

[4]William Fahle, ©2005

0.3 Matroids

Next up are matroids and the greedy algorithm. This complex subject is tackled well, and things begin to turn toward more computer-science related subjects, including linear programming and complexity theory. Characterizations are given for matroids, and matroid duality is explained, as well as using the greedy algorithm as an approximation algorithm for matroids.

0.4 Flow & Circulation

Ford and Fulkerson's theorems are covered in detail, and the augmenting path algorithm is described. As given, if one can find an augmenting path from the source to the sink in a network, that path can be used to add to the flow; when one cannot find such a path there is no more flow to be found. This result is similar to the min cut/ max flow theorem. Edmonds and Karp's important improvement for strictly polynomial time is discussed. In the Edmonds and Karp algorithm, the augmenting path chosen is always the shortest one. Many graphical examples are given in this section of the book to explain the complex phoenomena of flows and networks.

A brief treatment of colorings and Cayley graphs is followed by a more complete discussion of circulations. Circulations are similar to flows, except that the sink has an equalizing flow back to the source. This section includes information on Klein's algorithm for optimal circulations, potential functions and ϵ-optimality. Finally Goldberg and Tarjan's minimum mean cycle cancllling algorithm is given as an example of a much simpler way to determine optimal circulations.

0.5 Network Simplex

For this second edition, a new chapter on the network simplex algorithms has been added. This is an algorithm which adapts the regular simplex algorithm from linear programming to graph theoretical problems. The important problem of minimum cost flow is shown to have a solution with the network simplex method, and we also see other problems which translate easily to minimum cost flow. The problem of network synthesis is treated next; given paramters, we seek a network that matches those paramters.

0.6 Maximal Matchings

Good coverage is given of the bizzare and interesting algortihm by Edmonds for determining maximal matchings. This algorithm looks for what he terms blooms and trees to collapse a network down to a matching. Then blooms are reexpanded back to work through each of them. The algorithm is given in great detail with numerous examples and walkthoughs. Pseudo-code is also given. Other problems examined include matroid matching, weighted matchings, the Chinese postman problem, and the Hungarian algorithm.

0.7 NP-Complete problems

Up to this point, the problems treated are polynomial-solvable. In the final chapters, the Traveling Salesman Problem and other NP-hard or NP-complete problems are discussed. As is usually the case when discussing NP-complete problems, approximation algorithms and branch-and-bound are discussed.

0.8 Conclusion

The appendices include a small catalog of NP-complete problems, a detailed list of solutions to all the problems in the book, a list of special symbols used in the book, and hundreds of references to the field. A fifteen-page index rounds out the book, which overall is a thourough and excellent treatment of a field which is booming but thus-far short on textbooks. If there is one thing missing from this book, it is a discussion of the problems in the field which are not yet known to be polynomial solvable, but which are also not yet known to be NP-complete. Of course this information is due to change over time, and so would become quickly dated.

Review of

Immunocomputing: Principles and Applications[5]

2003

Authors: Alexander O. Tarakanov, Victor A. Skormin, Svetlana P. Sokolova

Publisher: Springer-Verlag New York, Inc.

Reviewer: Wenzhong Zhao

Department of Computer Science

University of New Mexico

wzhao@cs.unm.edu

1 Overview

The natural immune system is a complex system with several mechanisms for defense against pathogenic organisms. From the perspective of information-processing, the main purpose of the immune system is to solve recognition and classification tasks, and categorize cells or molecules as self or non-self. It learns through evolution to distinguish between foreign antigens (e.g., bacteria, viruses) and the body's own cells or molecules. The introduction argues that the immune system is much better understood than the nervous system.

As a highly parallel and distributed adaptive system with memory, recognition, learning, and decision-making capabilities, the biological immune system provides a remarkable information-processing model in the computational field. This emerging field is sometimes referred to as *Immunocomputing* (IC) or *Artificial Immune Systems* (AIS). Although it is so new (it was "officially" established in 1999 when Dasgupta published the first book in this area.), immunocomputing, with a strong relationship to other biology-inspired computing models, such as *Artificial Neural Networks* (ANN), *Cellular Automata* (CA) and *Evolutionary Computation* (EC), is establishing its uniqueness and effectiveness as a natural-computing media for solving computationally intense, complex problems.

"Immunocomputing: Principles and Applications", the book under review, introduces *immunocomputing* as a new computing approach that bridges between immunology and computer engineering. This new approach applies the principles of information processing by natural proteins and immune networks to the field of computation. It demonstrates how mathematical bases and immunology together form the new immunocomputing paradigm. It also integrates key aspects of pattern recognition, language representation and knowledge-based reasoning.

[5]Wenzhong Zhao,©2005

There are very few books with topics specific to immunocomputing and its applications on the market. This is a rare kind of book that explores the possibility of a computational approach that replicates the principles of the information processing in the immune system. The authors try to make the book accessible to audience with different backgrounds. Following the introduction to the key mechanisms of biomolecular computing, they cover, in considerable detail, the necessary mathematical bases for the major building components in the IC technology, including formal proteins, interaction between formal proteins, and formal immune networks.

2 Summary of Contents

The book consists of seven chapters, and a conclusion section. In addition, it features an extensive bibliography and glossary of symbols at the end.

In **Chapter 1**, *Introduction*, the authors try to convince readers that immunocomputing is a potential model for high performance computing. This chapter provides a sketch of the key biomolecular mechanisms of information processing. It introduces, from the computing viewpoint, natural proteins, which are the main components of the immune system as well as the mechanisms of protein behavior such as protein folding or self-assembly, in addition to covering molecular recognition, and the immune system.

Chapter 2, *The Mathematical Basis of Immunocomputing*, describes a rigorous mathematical model for immunocomputing. It introduces the notion of *formal protein* (FP), which abstracts the biophysical principle of free energy dependence from the spatial conformation of a protein. Based on the three dimensional geometry, the spatial conformation of a natural protein is determined by three fixed *valence angles* and two *torsion angles* between neighboring atoms. The mechanics of self-assembly, such as binding energy, the allosteric effect and networks of binding are also discussed.

The authors introduce the two models of immune cells: *formal B-cells*, and *formal T-cells*. These cells can either proliferate or die depending on whether they bind with an FP. They also develop a mathematical model for a *formal immune network*. (FIN) which is a network of bindings between FPs (such as receptors of *B-cells*, receptors of *T-cells*, and free FPs). The authors give a brief review of other related topics such as quaternion algebra and singular value decomposition.

Modern molecular biology has discovered that proteins utilize remarkably elegant, precise, and reliable mechanisms to recognize their own or foreign molecules or cells. These mechanisms are parallel and distributed, and will play a key role in immune and intellectual processes. **Chapter 3**, *Pattern Recognition*, develops an approach to pattern recognition by immunocomputing using the recognition between formal proteins. The main feature of this approach is that it considers an arbitrary pattern as a way of setting a binding energy.

A formal protein is represented as an *n-peptide* with n links, each of which can be represented by a special kind of *quaternion*. Recognition between two FPs is then defined as their binding. Two FPs recognize each other if they interact with a binding energy less than or equal to the particular threshold. The lower the binding energy, the better the recognition. The extreme values of the binding energy are used as criteria for finding a solution to a pattern recognition problem.

The authors also define the notion of *specificity of recognition* as the sum of squares of deviations of the binding energies over FPs. They show that folding a vector to a matrix does not decrease the specificity. Instead, the specificity either increases or remains the same. Finally, they describes

how the task of pattern recognition can be solved by immunocomputing through supervised and unsupervised learning.

Chapters 4, *Language Representation and Knowledge Based Reasoning*, provides a theoretical framework for language representation and knowledge-based reasoning with immunocomputing. The authors try to solve these problems using the representation and recognition of self and non-self in the natural immune system.

This approach considers a chain of linguistic symbols (i.e., a word) as an FP. An admissible word in the language is defined as a word for which the free energy of the corresponding FP does not exceed a particular threshold. Admissible subwords are considered as subpeptides or secondary structures of an FP. The authors show that the eigenvalues and eigenvectors of matrices over linguistic symbols have the potential for useful applications of immunocomputing in language representation.

The linguistic relations are described as interactions between FPs. Knowledge-based reasoning (e.g., an attributive context free grammar) is equivalent to a set of T-cells described by certain rules, with antigene corresponding to the axiom of the grammar, and so on. This approach can simulate grammars for solving inference problems.

After building a strong mathematical and theoretical framework for the new computing technology, the authors start to describe applications of the immunocomputing approach. **Chapter 5**, *Modeling of Natural and Technical Systems*, develops an immunocomputing approach to modeling natural and technical systems, including native proteins, computer networks, and virtual clothing. The authors demonstrate how these systems are formalized as special kinds of FPs and FINs.

In the case of modeling natural proteins, the spatial structure of a natural protein is represented with its secondary structures, namely *α-helix*, *β-sheet*, and *β-turn*. They establish an analytical model of the parameters of secondary structures of proteins as a special case of FP. They also argue that the spatial configuration and physiological properties of proteins are determined by amino acid sequence.

The authors also discuss how different events (i.e., for multicast protocols) in computer networks can be synchronized with an IC model, and how protein folding can be applied in virtual reality, such as folding or draping of cloth.

Further possible applications are discussed in detail in **Chapter 6**, *Applications*. These real-life applications include detection of dangerous situations in near-earth space, evaluation of complex ecological and medical indicators, surveillance of the plague in central Asia, and development of intelligent security systems.

Chapter 7, *Immunocomputing System: Architecture and Implementation*, discusses the concept of an *immunocomputer*. It describes a possible means of constructing immunocomputers utilizing the principles of information processing by natural proteins and immune networks. It suggests a hardware implementation of formal immune networks in a special *immunochip*, which would be used as the core of an immunocomputer. Finally, they describe a software emulator of the immunochip, as well as biochip technology which brings together computer chips with biomedical assays and laser-based detectors.

The last section, *Conclusion*, completes the book by offering a summary of the main features and innovations associated with the IC approach. The authors also discuss the reality of the IC approach, admitting that it is still in its early stage, and that several gaps need to be clarified and

resolved.

3 Opinion

This book introduces an immunocomputing (IC) approach, whose computing strategies are based upon the principles of information processing by natural proteins and the immune system. It bridges between immunology and computer science, and demonstrates how mathematics and immunology together form a new immunocomputing paradigm.

The book achieves its goal of being a thorough introduction to immunocomputing, although some readers might be uncomfortable with its intense mathematical formalizations of ideas. The book succeeds in establishing a rigorous mathematical basis for immunocomputing with recent findings in immunology and biochip development. It also integrates key aspects of pattern recognition, language representation, and knowledge-based reasoning.

This book is recommended for experts in computer science, artificial intelligence and biomolecular computing, immunologists, and students interested in immunocomputing. The authors include enough detail to make it self-contained, yet it is still accessible to readers with different backgrounds. However, some previous training in both linear algebra and three dimensional geometry would definitely be helpful.

Acknowledgments

The author thanks Chad Lundgren and Michael Singleton for proofreading. The author was supported by NIH Grant Number 1P20RR18754 from the Institutional Development Award (IDeA) Program of the National Center for Research Resources.

Review of *Term Rewriting Systems*[6]
by Terese
Cambridge University Press, 2003, 884 pages
ISBN: 0-521-39115-6

Reviewer: Frederic Loulergue
Laboratory of Algorithms, Complexity and Logic
University Paris Val de Marne
Creteil, France

1 Overview

TeReSe was the name of a Term Rewriting Seminar held by the authors at Vrije Universiteit in Amsterdam from 1988 to 2000. These twelve authors (including the three editors, Mark Bezem, Jan Willem Klop and Roel de Vrijer) have taken that name as the writers of this book.

Rewriting systems describe step-wise transformations of objects. In Term Rewriting Systems (TRS), an object, a term, is replaced by another one, accordingly to rules of the system. Computa-

[6]Frederic Loulergue, ©2005

tions are step-wise transformations of objects. Thus TRS are a foundational theory of computing. TRS have applications in many areas such as functional programming or automatic theorem proving.

This monograph is intended to be "useful for research purposes, but also for teaching purposes". Thus this book starts at an elementary level to provide a foundation for the rest of the text. Usually TRS are understood as first-order TRS. The presented work covers also higher-order TRS and other advanced topics. Much of the advanced material appears here for the first time in book form. In almost 900 pages, it gives an extensive presentation of the field of term rewriting systems. Background knowledge from set theory and logic are reviewed in the appendix.

2 Summary of Contents

The first four chapters provide the basic notions for the rest of the book. It starts (chapter 0) with motivating examples. Then Abstract Reduction Systems (ARS) are presented (chapter 1). Such systems are simply a set and a set of relations on this set. This notion allows to define aspects of rewriting independently of the objects which are rewritten. The key concepts *confluence* and *termination* are introduced. An ARS is confluent if for all element a such as it can be rewritten (in zero or several steps) to an element b and be rewritten to an element c, then there exists an element d which can be obtained in rewriting b and in rewriting c. An ARS is terminating when for all element a, every rewriting sequence starting from a is finite. If we think about computations rather than rewritings, it means that in a confluent and terminating system we cannot have two different results if we launch two times a computation on the same data and that we cannot have infinite computations.

Chapter 2 presents the notions attached to first-order term rewriting systems: terms, occurrences, contexts, substitutions, matching and examples. The chapter ends with the important notions of *overlapping* and *critical pairs*. Consider for example the following TRS where F, G denotes function symbols, a, b constants and x variable:

$$F(G(x)) \rightarrow x \quad (1)$$
$$G(a) \rightarrow b \quad (2)$$

The term $F(G(a))$ could be rewritten to a by the first rule but it could also be rewritten to $F(b)$ by the second rule. This phenomenon is called *overlap*. The critical pair in this case is $\langle F(b), a\rangle$. A critical pair is convergent if its two component can be rewritten to a common element. In a TRS if a critical pair is not convergent then the TRS is not confluent. The converse does not hold but we have weak confluence. Weak confluence and termination ensure confluence.

Chapter 3 is about several examples of TRS and other formats of rewriting. These chapters contains many exercises which help the reader to understand the many concepts introduced. In chapter 1, these exercises are placed at the end of the chapter but for the two next chapters they are included thorough the text. The solutions are not given in the book but some of them are given on the web site devoted to the book: http://www.cs.vu.nl/ terese/.

After this introduction, central concerns of TRS are exposed. One of them is orthogonality (chapter 4). We saw that overlap can make a TRS non confluent. Non-left-linearity of rules (the left-hand side contains several occurrences of the same variable) is also responsible for non-confluence.

Thus it is interesting to study the confluence of orthogonal systems which have no overlap between any rules and whose rules are all left-linear. Indeed orthogonal systems are confluent and this is shown using several methods.

Orthogonal systems are confluent and we can wonder if the confluence of a TRS can be decided in general. Chapter 5 is about decidability of confluence and termination of TRS. In general these properties are undecidable. For some classes of TRS some properties are decidable (for example normalization is decidable for recursive program schemes). In a second part of this chapter another approach is stressed: are confluence and termination preserved when two TRS having these properties are composed ? Confluence is preserved, it is said to be a modular property, but termination is not modular in general but it is modular for orthogonal TRS only.

Termination is undecidable. But techniques for proving termination of many TRS used in practice exist (chapter 6). There are three kinds of methods:

- Semantical methods. Roughly speaking, TRS are given a semantics which has a well-founded order and it is shown that for every term t if t is rewritten to s then the interpretation of t is strictly greater than the interpretation of s. In a well-founded order there are no infinite chain, so there cannot be infinite rewriting sequences.
- Syntactical methods unlike previous methods can be automated but they can be apply only to restricted classes of TRS.
- Transformational methods to transform TRS outside the previous classes into TRS of these classes in order to be able to apply a syntactical method.

An equational specification could be seen as a TRS without orientation of the rules. Chapter 7 presents Knuth-Bendix completion which constructs a TRS from an equational specification, the TRS solving the same equations than the original specification. The proof of the algorithm is given. E-unification (solving equations modulo an equational theory E) is also discussed.

All the previous chapters were on first-order TRS. λ-calculus and its properties are presented (chapter 10) as an introduction to higher-order term rewriting systems. Two main formalisms are presented and compared: Higher-order Rewriting Systems (HRS) and Combinatory Reduction Systems (CRS). Higher-order rewriting systems allows to deal with bound variables. For example we often have bound variables in mathematics:

$$\int (f(x) + g(x))dx \;=\; \int f(x)dx \;+\; \int g(x)dx \tag{3}$$

CRS have their origin in the study of extensions of the λ-calculus whereas HRS have their origin in the study of higher-order logic. It is shown that orthogonal HRS are confluent. Related formalisms are outlined.

The last chapter (chapter 15) has not a theoretical content: it is a review of more than 25 existing languages and systems related to TRS. The main features and the application areas of each system are given together with a link to web resources.

The other chapters not described previously contain very advanced material: Equivalence of reductions (almost a small monograph by itself, 170 pages!), Strategies, Infinitary rewriting, Term graph rewriting and Advanced ARS theory.

3 Opinion

The content of the book in huge, including both basic material and recent work from the research literature much of it of the authors and their collaborators. This support the author's intention that the book serve both as a textbook and a reference book.

Coverage is quite dense so as a textbook it is suited for use with an instructor and well prepared students. The broad variety of topics allows to find material suitable for a variety of audience. For example the four first chapters could be used as an introductory undergraduate course. Maybe [1, 2] are less frightening for (under)graduate students. The book is largely self contained but a prior experience of the material presented in the appendix is preferable. The exercises vary in difficulty but are not graded. Some solutions are given on-line, but for a very small subset of exercises (only for chapter 1 and 14).

As a reference book it was of course not possible to cover all the topics of term rewriting research, but the book presents all the basic material and many very advanced topics. There are also a detailed bibliography and three index: for notations, for authors and for subjects.

My recommendation is that anyone interested in rewriting needs this book.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, 1998.

[2] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–117. Oxford University Press, Oxford, 1992.