Schulze-Kremer

Molecular Bioinformatics
Algorithms and Applications

Steffen Schulze-Kremer

Molecular Bioinformatics

Algorithms and Applications

Walter de Gruyter · Berlin · New York 1996

Dr. Steffen Schulze-Kremer
Westfälische Str. 56
D-10711 Berlin
Germany

email:
steffen@chemie.fu-berlin.de or
steffen@mycroft.rz-berlin.mpg.de

WWW:
http://www.chemie.fu-berlin.de/user/steffen or
http://mycroft.rz-berlin.mpg.de/~steffen

With 124 figures and 36 tables.
Cover illustration: Wire-frame-model of DNA. Courtesy of Steffen Schulze-Kremer.

Library of Congress Cataloging-in-Publication Data

Schulze-Kremer, Steffen.
Molecular bioinformatics: algorithms and applications / Steffen Schulze-Kremer.
Includes index.
ISBN 3-11-014113-2 (alk. paper)
1. Molecular biology – Computer simulation. I. Title.
OH506.D346 1995 95-40471
574.8'8'0113–dc20 CIP

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Schulze-Kremer, Steffen:
Molecular bioinformatics: algorithms and applications /
Steffen Schulze-Kremer. – Berlin; New York; de Gruyter, 1995
ISBN 3-11-014113-2

♾ Printed on acid-free paper which falls within the guidelines
of the ANSI to ensure permanence and durability.

Converted by: Knipp Medien und Kommunikation, Dortmund. – Printing: Gerike GmbH, Berlin. Binding: Dieter Mikolai, Berlin. – Cover Design: Hansbernd Lindemann, Berlin.

To my friends,
especially Otto B., Jochen, Kathrin and Grit.

Preface

Molecular bioinformatics is a newly emerging interdisciplinary research area. It comprises the development and application of computational algorithms for the purpose of analysis, interpretation and prediction of data and for the design of experiments in biosciences. Background from computer science includes but is not limited to classical von Neumann computing, statistics and probability calculus, artificial intelligence, expert systems, machine learning, artificial neural nets, genetic algorithms, evolutionary computation, simulated annealing, genetic programming and artificial life. On the application side, focus is primarily on molecular biology, especially DNA sequence analysis and protein structure prediction. These two issues are also central to this book. Other application areas covered here are: interpretation of spectroscopic data and discovery of structure-function relationships in DNA and proteins. Figure 1 depicts the interdependence of computer science, molecular biology and molecular bioinformatics.

The justification for introducing a new label for a range of rather diverse research activities is motivated by the following four observations.

1) Exponential growth of data requires new ways of information processing.

A vast amount of genetic material has been sequenced to date. Many laboratories continue to output sequences of new genes world wide. There is an exponential growth[1] of known DNA sequences and protein structures. In March 1995, about 42.000 known protein sequences were present in the SwissProt[2] database and roughly 2900 three-dimensional structures of proteins, enzymes and viruses in the Brookhaven Protein Database[3]. The international Human Genome Organisation (HUGO) is currently attempting to sequence a complete human genome[4]. By March 1995, 3748 genes were mapped[5] in a total of approximately 25 million base pairs, which is still only about 0.8% of the total size in nucleotides of one entire human genome. Such quantities of data cannot be looked at and analysed without

1 L. Philipson (EMBL), *The Human Genome and Bioinformatics*, Proceedings of the Symposium **Bioinformatics in the 90's**, ASFRA Edam, Maastricht, 20. November, 1991.

2 R. D. Appel, A. Bairoch, D. F. Hochstrasser, *A new generation of information retrieval tools for biologists: the example of the ExPASy WWW server*, **Trends Biochem. Sci.**, vol 19, pp. 258-260, 1994. Internet World Wide Web Access at http://expasy.hcuge.ch or http://129.195.254.61. See Chapter 8.1 or ask your local computer specialist for details on WWW.

3 **PDB Newsletter**, Brookhaven National Laboratories, no 68, April 1994, Internet World Wide Web Server at http://www.pdb.bnl.gov.

4 T. Caskey, President of the Human Genome Organisation, HHMI Baylor College of Medicine I, Baylor Place, Houston, 77030 Texas, World Wide Web Server at http://www.bcm.tmc.edu.

5 HUGO Report Card March 1995, Internet World Wide Web Server of the Genome Data Bank at http://gdbwww.gdb.org.

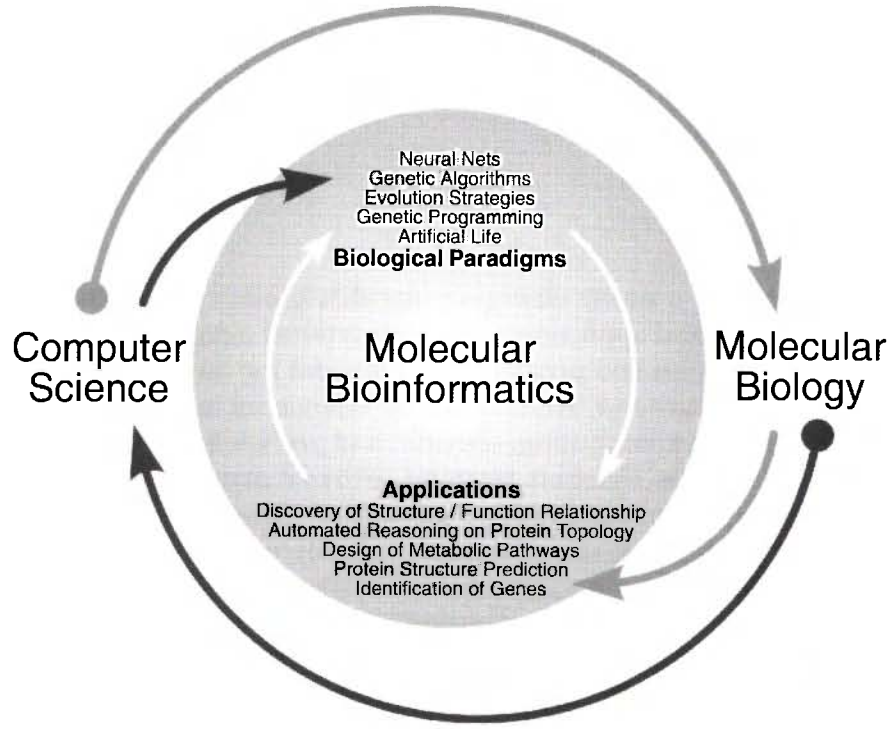

Figure 1: Scope of Molecular Bioinformatics.
Relation of molecular bioinformatics to computer science and biology. Computer science looks at problems in molecular biology and offers computational algorithms. Molecular biology looks at computer science and suggests new paradigms for information processing. These new approaches can be used to develop new applications which in turn may reveal new features in biology.

the help of computers. Specialised software obviously becomes an essential prerequisite for data storage and retrieval. This is a new challenge for traditionally empirically oriented bioscientists. It requires expertise in algorithmic theory and experience in programming computers *together* with a profound understanding of the underlying biological principles. Instead of being considered an appendix to either bioscience or computer science, such concentrated, interdisciplinary effort wants and deserves recognition on its own.

2) Better algorithms are needed to fully explore current biochemical databases.

In order to understand the purpose of a genetic sequence, it is in most cases not sufficient to perform a statistical analysis on the distribution of oligonucleotides. This is because patterns in DNA or protein tolerate slight alterations in some positions without losing their biochemical function. Furthermore, even for a simple, straightforward statistical analysis, e.g. on the frequency distribution of amino acid residue triplets to predict secondary structure, there is not (yet) enough data in our

databases[6]. More sophisticated methods are needed to explore the treasure of existing databases. *Data mining* becomes a key issue. A number of biocomputing related national and international research programs confirm this notion[7]. Computer scientists and mathematicians have been developing a variety of algorithms for quite some time, many of which bioscientists are not yet aware of. The establishment of molecular bioinformatics as a recognised, self-reliant discipline will be beneficial in promoting the dialogue between the two disciplines and to attract researchers from both areas.

3) Computer science learns from nature.

Molecular bioinformatics is not a one-way street. Biology has some interesting ideas to offer computer scientists, as can be seen in the case of artificial neural nets[8], genetic algorithms[9], evolution strategies[10], genetic programming[11], artificial life[12] and (from physics) simulated annealing[13]. To gather inspirations, to learn more from nature and to fully exploit existing approaches, a closer interaction between both disciplines is required. The professional endeavour to continue along these lines is covered neither by computer science nor biology. To fill this gap, molecular bioinformatics as a self-standing discipline can become the common platform for exchange and research in the aforementioned areas.

4) Creating a real-world application oriented forum.

Computer scientists sometimes are able to solve a particular type of problem efficiently and would like to apply their algorithms in a real-world situation to evaluate their performance and to gain new insights. If a computer scientist has not yet decided which area to turn to, he or she can profit from molecular bioinformatics. Here, they will find people they can communicate with in their technical language and who can introduce them to some of the most challenging and potentially rewarding problems science nowadays has to offer. One forum for such interaction has become the Internet course on Biocomputing at the Globewide Network

---

6 M. J. Rooman, S. J. Wodak, *Identification of predictive motives limited by protein structure data base size*, **Nature**, no 335, pp. 45-49, 1988.

7 Commission of the European Communities: Biomolecular Engineering (BEP 1982-1984), Biotechnology Action Program (BAP 1985-1989), Biotechnology Research for Innovation, Development and Growth in Europe (BRIDGE 1990-1993), Biotechnology (BIOTECH2 1994-1998). German Minister for Research and Technology: Molecular Bioinformatics (1993-1996).

8 M. Minsky, S. Papert, **Perceptrons – Expanded Edition**, MIT Press, Cambridge MA, 1988.

9 J. H. Holland, **Adaptation in natural and artificial systems**, University of Michigan Press, Ann Arbor, 1975.

10 I. Rechenberg, **Evolutionsstrategie**, Frommann-Holzboog, Stuttgart, 1973, 1994.

11 J. Koza, **Genetic Programming (I, II)**, MIT Press, Cambridge MA, 1993, 1994.

12 C. G. Langton C. Taylor, J. D. Farmer, S. Rasmussen (Eds.), **Artificial Life II**, Addison-Wesley, 1992.

13 S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, *Optimisation by Simulated Annealing*, **Science**, vol 220, no 4598, pp. 671-680, 1983.

Academy, Virtual School of Natural Sciences[14] that enables students and instructors from all over the world to participate in a university course.

This book attempts to provide an overview on advanced computer applications derived from and used in biosciences and in doing so to help define and promote the newly emerging focus on molecular bioinformatics. It is intented to serve as a guidebook for both biologists who are about to turn to computers and for computer scientists who are looking for challenging application areas. The reader will find a variety of modern methodological approaches applied to a number of quite divers topics. This collection gives a contemporary overview of what, in principle, can be achieved by computers in bioscience nowadays and what is yet difficult to grasp. In many places detailed information on the availability of software is included, preferably through Internet addresses of freely accessible World Wide Web Servers or by email. In the appendix, a list of Internet entry points to biocomputing facilities provides orientation to the novice in molecular bioinformatics and assists the specialist in staying abreast of the latest scientific developments.

The book is also intended to inspire further research. Researchers already familiar with molecular bioinformatics may still gather new ideas from the material presented as sometimes related methodologies are applied to unrelated problems and related problems are treated by completely different algorithms. The comparison in such cases may provide valuable insights.

In order to best serve an interdisciplinary audience, each chapter starts with an introduction into the mathematical basis of the algorithm used. Then, results of one or more original research papers applying that approach to different biochemical problems are presented and discussed. Finally, limitations and open questions are established and suggestions are offered along which lines to continue research. As in a manual, the material presented in this book should be sufficient to enable the reader to rebuild an application or to modify or extend it. Whether or not he or she intends to do so, the presentation of each topic should in any case be adequately detailed to allow the reader to decide if the proposed approach looks promising to be utilised for his or her own project.

Naturally, this book has its limits. Not all relevant work can be presented here. Notably, molecular mechanics, molecular dynamics and classic molecular modelling are important topics in molecular bioinformatics. However, these disciplines have matured and grown to such an extent that describing them thoroughly would make up a whole book by itself. In fact, such books have already been written[15,16]. The development of biological systems to act as data storage and com-

14 The author is one of the authors of the GNA-VSNS Biocomputing course. For more information, first send an email to the author at steffen@chemie.fu-berlin.de, steffen@mycroft.rz-berlin.mpg.de or contact vsns-bcd-faculty@bioinformatics.weizmann.ac.il.

15 C. L. Brooks III, M. Karplus, B. M. Pettitt, **Proteins: A theoretical perspective of dynamics, structures and thermodynamics**, Wiley, 1988.

16 A. M. Lesk, **Protein Architecture – A practical approach**, IRL Press, Oxford, 1991.

puting devices[17,18,19] is also not covered in this book. This more empirically oriented work contrasts the computational aspect of molecular bioinformatics as emphasised here.

A choice was made to include some early projects in molecular bioinformatics as these comprise the roots of this discipline. A number of important conclusions can be drawn from these works which are still valid and which will need to be considered in future applications. The larger part of this book, however, is devoted to fairly recent work. To aid the reader continue his or her way through molecular bioinformatics, the first chapter also contains references to a number of approaches that had to be omitted.

I hope this book will bring inspiration and information to its readers.

17 B. H. Robinson, N. C. Seeman, *The design of a biochip: a self-assembling molecular-scale memory device*, **Protein Engineering**, vol 1, no 4, pp. 295-300, 1987.

18 B.C. Crandall, J. Lewis (Eds.), **Nanotechnology: Research and Perspectives**, ISBN 0-262-03195-7, 1992.

19 **Proceedings of the International Conference on Molecular Electronics and Biocomputing**, September 1994, Goa, India, Tata Institute of Fundamental Research, Ratna S. Phadke, email ratna@tifrvax.tifr.res.in.

Table of Contents

1. Introduction

Although molecular bioinformatics has only recently become a self-reliant, recognised discipline[1,2,3], first work in the spirit of molecular bioinformatics dates back to the late 50's[4,5,6,7,8,9]. Since then the idea spread that computer scientists and biologists can profit from one another. Growing interest in interdisciplinary research connecting computer science and molecular biology came primarily from an exponential growth of known biological sequence[10] and structure[11] data, from the need for more sophisticated methods of data analysis and interpretation in biosciences, and from the discovery of nature as a source of models for efficient computation.

Computer scientists have discovered this trend and included workshops on molecular bioinformatics into most of their important international conferences. The proceedings and workshop notes of the Hawaii International Conference on System Sciences (HICSS)[12], the American Association for Artificial Intelligence (AAAI)[13], the conference on Parallel Problem Solving from Nature (PPSN)[14], the European Conference on Machine Learning (ECML)[15], the International Con-

1 Commission of the European Communities: Biotechnology Action Program (BAP 1985-1989), Biotechnology Research for Innovation, Development and Growth in Europe (BRIDGE 1990-1993), BIOTECH2 (1994-1998). German Minister for Research and Technology: Molecular Bioinformatics (1993-1996).

2 L. Hunter (Ed.), **Artificial Intelligence and Molecular Biology**, MIT Press, Cambridge MA, 1993.

3 S. Schulze-Kremer (Ed.), **Advances in Molecular Bioinformatics**, IOS Press, 1994.

4 F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, **Psychological Review**, vol 65, pp. 386-408, 1958.

5 M. Minsky, S. Papert, **Perceptrons**, MIT Press, Cambridge MA, 1969.

6 I. Rechenberg, **Evolutionsstrategie**, Frommann-Holzboog, Stuttgart, 1973, 1994.

7 J. H. Holland, **Adaptation in natural and artificial systems**, University of Michigan Press, Ann Arbor, 1975.

8 M. Stefik, *Planning with Constraints*, **Artificial Intelligence**, no 16, pp. 111-169, 1981.

9 B. G. Buchanan, E. H. Shortliffe, **Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project**, Addison-Wesely, 1984.

10 A. Bairoch, B. Boeckmann, *The Swiss-prot protein sequence data bank*, **Nucl. Acids Res.**, vol 20, pp. 2019-2022, 1992. Internet World Wide Web Access at http://expasy.hcuge.ch.

11 **PDB Newsletter**, Brookhaven National Laboratories, no 68, April 1994. Internet World Wide Web Server at http://www.pdb.bnl.gov.

12 **Proceedings of the 26th and 27th Hawaii International Conference on System Sciences**, IEEE Computer Society Press, Los Alamitos CA, 1993, 1994.

13 **Proceedings of the 9th Conference on Artificial Intelligence**, AAAI Press, Menlo Park CA, 1991.

14 **Proceedings of the 2nd Conference on Parallel Problem Solving from Nature**, (R. Männer, B. Manderick Eds.), North Holland, 1992.

15 **Proceedings of the European Conference on Machine Learning**, (P. B. Brazdil Ed.), Springer, 1993.

ference on Genetic Algorithms (ICGA)[16], the International Joint Conference on Artificial Intelligence (IJCAI)[17], and especially the International Conference on Intelligent Systems for Molecular Biology (ISMB)[18] are all valuable recourses of information. A freely accessible database of international researchers working in molecular bioinformatics is available[19].

Molecular bioinformatics comprises three fundamental domains: the development of algorithms and computer programs for applications in biosciences; the abstraction of biological principles for new ways of information processing; and the design and use of biochemical systems for data storage and as computing devices[20,21]. This book concentrates on the first two issues. The design of an artificial molecular computer is expected to become a feasible and profitable endeavour in the next decade but will not be treated here. Molecular mechanics, molecular dynamics[22] and classic molecular modelling[23] are also underrepresented here. This is because these topics have already been around for some time and an exhaustive treatment of their issues would fill several books on their own. Nevertheless, they contribute significantly to molecular bioinformatics.

Although computer science and biology can be treated separately, and, in fact, were so most of the time, there are four levels of interaction between these two disciplines.

1) Computer programs can help analyse biological data.
2) Computer models and simulation can help explain biological behaviour.
3) Atomic interactions can be used to build a molecular data processor.
4) Biology can serve as a source of models for computational algorithms.

Note the analogy between 1 and 3 and between 2 and 4. Topics 1 and 3 both use one discipline to process / enact the fundamental principle of the other. Topics 2 and 4 use one discipline to model / simulate the other's emergent behaviour. There is also a possibly recursive interaction. Contributions from biology can be used to create computer programs (or computers) that in turn can be used to operate on biological data. This happens, for example, in the application of genetic algorithms to the problem of protein folding. In this case, the original biological structure, a gene, and the biological algorithm, evolution, have been abstracted to give a general computational model. The artificial "genes" of that model may then be mapped

16 **Proceedings of the 4th International Conference on Genetic Algorithms**, (R. K. Belew, L. B. Booker Eds.), Morgan Kaufmann, Los Altos CA, 1991.
17 **Proceedings of the Artificial Intelligence and Genome Workshop 26 at the IJCAI-93**, (J.-G. Ganascia Ed.), Institut Blaise Pascal, Laforia, Universite Paris VI, CNRS, 1993.
18 **Proceedings of the 1st and 2nd International Conference on Intelligent Systems for Molecular Biology,** (Eds. I: L. Hunter, D. Searls, J. Shavlik, II: R. Altman, D. Brutlag, P. Karp, R. Lathrop, D. Searls), AAAI Press, Menlo Park CA, 1993, 1994.
19 L. Hunter, **AIMB database**, anonymous ftp at lhc.nlm.nih.gov (130.14.1.128), directory /pub/aimb-db, 1993. Internet World Wide Web Server at http://www.nlm.nih.gov.
20 B. H. Robinson, N. C. Seeman, *The design of a biochip: a self-assembling molecular-scale memory device*, **Protein Engineering**, vol 1, no 4, pp. 295-300, 1987.
21 B. C. Crandall, J. Lewis (Eds.), **Nanotechnology: Research and Perspectives**, ISBN 0-262-03195-7, 1992.
22 C. L. Brooks III, M. Karplus, B. M. Pettitt, **Proteins: A theoretical perspective of dynamics, structures and thermodynamics**, Wiley, 1988.
23 A. M. Lesk, **Protein Architecture – A Practical Approach**, IRL Press, Oxford, 1991.

onto a linear transformation (i.e. proteins) of the very objects from which they originally had been derived. – A schematic diagram on the interplay of computer science, molecular biology and molecular bioinformatics is shown in Figure 1 (in the PREFACE).

The prior example illustrates the possibility of raising confusion by the use of technical terms from different contexts. A gene in biology has one meaning, but a different one in the context of a genetic algorithm. The same goes for neural networks or evolution strategies. In this book, definitions from both biology and computer science will be provided and care is taken not to confuse them.

This book contains a collection of methodological approaches applied to a variety of information processing tasks in molecular biology. Emphasis is put on methods that explore algorithmic features found in nature. Preferred applications are protein structure analysis and interpretation of genomic sequences.

The main text of this book is divided into the following five chapters. Each chapter covers a group of algorithmically related methodologies of data processing, e.g. artificial neural networks or genetic algorithms. First in each chapter, general features of the computational methodologies used are described. Then a number of sections follow each containing an application of that method to a different biological problem. Each section starts with an introduction to the biochemical background of the problem and on any application-dependent modifications of the method. Then, specific details of the implementation for that application are described. Original results from one or more research groups are presented to illustrate and evaluate the performance of that approach.

This book can be used as a guidebook for novices in computer science and molecular biology to explore the scope of molecular bioinformatics. For specialists, it can serve as a manual to rebuild the methods and applications presented here or to extend and adopt them to fit one's own subject of interest. In this case, it is valuable to compare the results of different methodologies on the same problem, e.g. on prediction of protein secondary structure.

The final part of this book is for reference. The reader will find a list of Internet entry points that can be used by anybody with a computer connected to the Internet to navigate in the World Wide Web and retrieve information on people, institutions, research projects and software in molecular bioinformatics.

1.1. Methodologies

Table 1 gives an overview of the algorithmic techniques used for molecular bioinformatics. The main methodologies are briefly described in the following paragraphs.

Artificial intelligence, symbolic computation and expert systems. The shift from numeric to symbolic computation and the focus on information processing procedures analogous to human mental processes distinguishes artificial intelligence programming from classic, procedural programming. Although artificial intelligence programs are also finally translated into a series of sequential instructions for one or more central procession units of a computer, they often al-

Methodology	Technique	Paradigm
Artificial Intelligence		human
	Expert Systems	
	Inference	
	Imagery /Vision	
	Knowledge Representation	
	Linguistics	
	Pattern Matching	
	Planning	
	Qualitative Theory	
	Heuristic Search	
Artificial Life		evolution
Artificial Neural Nets		neurons
Classifier Systems		economy
Evolutionary Computation		nature
	Evolution Strategies	evolution
	Genetic Algorithms	genetics
Practical Experimental Work		none
	Circular Dichroism	
	Molecular Biology	
	Nuclear Magnetic Resonance	
	X-ray Crystallography	
Genetic Programming		genetics
Logic		none
	Predicate Calculus	
	Prolog	
Machine Learning		human
	Conceptual Clustering	
	Decision Trees	
	Inference	
Mathematics		none
	Dynamic Programming	
	Graph Theory	
	Information Theory	
	Probability Calculus	
	Statistics (Monte Carlo)	
Simulated Annealing		physics

Table 1: Methodologies and Algorithms used in Molecular Bioinformatics.
This is an overview on some techniques used in molecular bioinformatics. "Paradigm" tells if the technique has a model in nature. Mathematics and logic are thought of as *a priori* concepts.

low a complex task to be represented in a more natural, efficient manner. Semantic use of symbols other than numerical and character variables provide the means

for coding highly structured applications. Lisp[24] and Prolog[25] are special symbol manipulating programming languages which are often used for coding artificial intelligence programs. They allow easy implementation of self-modifying and recursive programs. In Lisp, data and program can alter each other during run time, which permits flexible flow control[26]. Artificial intelligence research has produced various sophisticated applications, three of which shall be briefly mentioned here. AM[27] discovered concepts in elementary mathematics and set theory on the basis of knowledge on *mathematical aesthetics*. AM combines the features of frame representation, production systems and best-first search. Eliza[28] simulates a psychiatrist talking to a patient. Several patients actually believed there was a human responding. Macsyma[29] was the first program to actually carry out symbolic differentiation and integration of algebraic expressions. Since then, it has been extended to become a versatile and flexible mathematical package not only on Lisp machines.

Artificial life. This rather new research area focuses on the realisation of life-like behaviour in man-made systems consisting of populations of semi-autonomous entities whose local interactions with one another are governed by a set of simple rules[30]. Key concepts in artificial life research are local definition of components, parallelism, self-organisation, adaptability, development, evolution, and emergent behaviour. Apart from being a philosophically fascinating area, artificial life in the future could help provide simulation models for interacting biological systems (e.g. cells, organisms) and support the design of *software robots*[31].

Artificial neural networks. The intention to abstract the behaviour of biological neural networks from their natural environment and to implement their function on a computer produced a large variety of so-called artificial neural networks[4,32]. These programs can often discriminate or cluster data better than statistics or probability calculus. Artificial neural networks are non-deterministic in the sense that their response depends on the order in which the training examples are presented to the net. The goal of simulating the behaviour of actual biological neurons and neural networks is not part of the mainstream in artificial neural network research. It turned out that an accurate, detailed simulation of the behaviour of only one biological neuron requires much more computational effort than is ben-

---

24 P. H. Winston, B. K. P. Horn, **LISP**, Addison-Wesely, 1984.

25 W. F. Clocksin, C. S. Mellish, **Programming in Prolog**, Springer, 1984.

26 E. Charniak, D. McDermott, **Introduction to Artificial Intelligence**, Addison-Wesley, 1985.

27 D. B. Lenat, *AM: Discovery in mathematics as heuristic search*, in **Knowledge-Based Systems in Artificial Intelligence** (R. Davis, D. B. Lenat Eds.), pp. 1-225, McGraw-Hill, New York, 1982.

28 J. Weizenbaum, *ELIZA – A computer program for the study of natural language communication between man and machine*, **Communications of the ACM**, vol 9, no 1, 1965.

29 C. Engleman, W. Martin, J. Moses, M. R. Genesereth, Macsyma Reference Manual, Technical Report, Massachusetts Institute of Technology, Cambridge MA and Symbolics Inc., 1977.

30 C. Langton (Ed.), **Artificial Life**, Addison-Wesley, p. xxii, 1989. Internet World Wide Web Server at http://alife.santafe.edu.

31 Softbods are autonomous agents that interact with real-world software environments such as operating systems or databases. Get more information from the Internet World Wide Web server at http://www.cs.washington.edu/research/projects/softbots/www/softbots.html.

32 J. A. Freeman, D. M. Skapura, **Neural Networks**, Addison-Wesley, 1991.

eficial for training artificial neural networks to perform data analysis tasks. Using more biologically detailed models has not yet significantly improved prediction performance of artificial neural networks.

Genetic algorithms and evolutionary computation. The idea to let a computer develop a solution for a problem like nature produces fit individuals by evolution led to the invention of evolution strategies[6] and genetic algorithms[7]. These approaches use "genetic" operators to manipulate numeric or string representations of potential solutions ("individuals"). A fitness function ranks all "individuals" of one "generation", the best of which are allowed to "survive" and to "reproduce". This is repeated for a fixed number of cycles or until a particular fitness criterion is fulfilled. Genetic algorithms operate on the "genotype" of a potential solution, evolution strategies on its "phenotype". Both methods have been shown to be superior to e.g. Monte Carlo search in extremely difficult search spaces. They are also non-deterministic because genetic operators are performed probabilistically.

Knowledge representation formalisms. Knowledge representation is a central issue in artificial intelligence research. As with humans[33], language also determines the limits of a computer's operation. The objective is therefore to find a representation formalism that can capture all essential details of an application, one that is compatible with the algorithm used and which can be coded efficiently. It should also allow intuitive use and easy maintenance of a knowledge base[34]. Prominent knowledge representation formalisms are objects[26], frames[35], scripts[36], production rules[9], predicates[37], decision trees[38], semantic nets[39], λ-calculus and functions[40], and classic, procedural subroutines. Currently, object-oriented programming style is becoming very popular, as can be seen in the wide-spread use of the object-oriented programming language C++ with its consequent use of classes[41].

---

33 L. Wittgenstein, **Tractatus Logico-Philosophicus**, Suhrkamp Edition, 1982.

34 The notion of a database traditionally means keeping data, possibly sorted, in one place. A knowledge base emphasises the data to be stored in a higher structured format, e.g. as rules or objects. Knowledge bases tend to be used interactively, as e.g. in expert systems. Also, a knowledge base is automatically updated and sometimes even generated automatically. There may be a mechanism to guarantee truth maintenance. In this book, the concepts of database and knowledge base are used almost interchangeably, as the difference between them is more one of emphasising their use and context rather than their contents. Almost any database can be used as a knowledge base, if properly accessed by sophisticated algorithms. In this book, the concept of a data bank is used to refer to (commercially) available collections of "raw" data, as e.g. a sequence data bank or a protein structure data bank.

35 M. Minsky, *A Framework for Representing Knowledge*, in **The Psychology of Computer Vision**, (P. H. Winston Ed.), MacGraw-Hill, New York, 1975.

36 R. C. Schank, R. P. Abelson, **Scripts, Plans, Goals and Understanding**, Erlbaum Hillsdale, New Jersey, 1977.

37 W. F. Clocksin, C. S. Mellish, **Programming in Prolog**, Springer, 1984.

38 J. R. Quinlan, *Discovering Rules by Induction from Large Collections of Examples*, in **Introductory Readings in Expert Systems**, (D. Michie Ed.), pp. 33-46, Gordon and Breach, London, 1979.

39 R. Quillian, *Semantic Memory*, in **Semantic Information Processing**, (M. Minsky Ed.), MIT Press, Cambridge MA, 1968.

40 J. McCarthy, *Recursive Functions of Symbolic Expressions and their Computation by Machine*, **Communications of the ACM**, vol 3, no 4, pp. 185-196, 1960.

41 M. A. Ellis, B. Stroustrup, **The Annotated C++ Reference Manual**, Addison-Wesley, 1990.

Machine learning. The field of machine learning studies computational methods for acquiring new knowledge, new skills, and new ways to organise existing knowledge[42]. This includes the modelling of human learning mechanisms. For example, Checkers[43] is a program that actually learned to play checkers better than expert human players. It did so by repetitive training, selecting relevant features from the board and weighting them properly. There are a number of ways to learn: rote learning, learning from instruction, learning by analogy, learning by deduction, learning by induction, learning by abduction, supervised learning and unsupervised learning. Computers can already be programmed to exhibit these types of learning on restricted domains. Machine learning algorithms can sometimes find hidden regularities and patterns in a biological database, which are invisible to purely statistical methods.

Logic and predicate calculus. The application of basic logic operators (not $\neg$, and $\wedge$, or $\vee$, follows $\rightarrow$, equivalent $\equiv$, there exists $\exists$, for all $\forall$) on simple predicates defines first order logic. Using first order logic as the basic principle for a computer programming language led to the origin of Prolog[37]. In contrast to procedural programming languages like Basic or Fortran, in Prolog emphasis is on how to *describe a problem*, not *the solution*. A built-in deductive inference machine then performs inverse resolution to derive valid solutions. The power of first order logic and Prolog comes from the ability to easily express non-numerical constraints, as is fundamental e.g. for the description of protein structure topologies.

Simulated annealing. The process of crystallisation at gradually decreasing temperatures can be abstracted to give a general optimisation procedure. Simulated annealing[44] performs a search in a multi-modal search space by exploring the valleys (if a global minimum is desired) in a random manner. As the (simulated) temperature decreases, less (fictitious, simulated) energy is becoming available to overcome the hills between neighbouring valleys. Typically, at the end of the run the probe arrives at a rather deep valley, which indicates a good solution. However, the algorithm does not guarantee that the global optimum be found. Simulated annealing is therefore preferably used on analytically intractable problems where one has no choice but to explore a large search space. The *ab initio* prediction of energetically favourable protein conformations is one such problem.

Statistics and probability calculus. Classic statistics and probability calculus provide well founded methods to analyse numerical databases. These methods still serve as the standard against which other approaches are measured. Any new methodology has to prove its merit by producing at least comparable results to statistics and probability calculus or by presenting qualitatively new statements that could not be derived by either of them.

---

42 **Encyclopedia of Artificial Intelligence**, (S. C. Shapiro Ed.), Wiley-Interscience, pp. 464, 1987.

43 A. L. Samuel, *Some Studies in Machine Learning using the Game of Checkers*, in **Computers and Thought**, (E. A. Feigenbaum, J. Feldman Eds.), pp. 71-105, McGraw-Hill, New Yorck, 1963.

44 S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, *Optimization by Simulated Annealing*, **Science**, vol 220, no 4598, pp. 671-680, 1983.

Classifier systems. One type of message passing, rule-based production systems in which many rules can be active simultaneously is called classifier system[45]. In such systems each rule can be interpreted as a tentative hypothesis on some factors relevant to the task to be solved, competing against other plausible hypotheses that are carried along and evaluated in parallel. Classifier systems got their name from the fact that they classify incoming messages from the environment into general sets. The condition parts of the production rules are matched against incoming messages from the environment and the action parts effect changes in the environment. Rules are assigned a strength value on the basis of their observed usefulness to the system. New rules are generated by genetic operators as in genetic algorithms. Classifier systems have been shown to learn strategies for successful, optimised behaviour in an adapting environment (e.g. poker play[46]).

Genetic programming. Derived from genetic algorithms is the concept of genetic programming[47]. Here, not a single solution to a particular problem is processed in an evolutionary manner but whole computer programs instead. Here again, Lisp is the programming language of choice. Similar to artificial neural networks, a fitness function measures how well the generated program reproduces known input / output values. When the training is finished, the program may be inspected, simplified and tried on a new set of input values with prior unknown results. One particular advantage of this approach is the ability to discover a symbolic expression of a mathematical function, not only a numerical approximation. By defining a proper set of primitives as basic building blocks (e.g. SIN, COS, EXP, LN for mathematical problems, or other, utterly application-specific user-defined operators) one can predetermine the appearance of the final solution.

1.2. Application Areas

The main application areas covered in this book are proteins and genes, but there are also a few other tasks not directly linked to either of them. Table 2 gives an overview of application areas in molecular bioinformatics. There is certainly more work that would fit under the label of molecular bioinformatics. However, this book is intended to give a representative selection, not a complete account of all work done. Owing to space limitations, a number of applications cannot be described here in detail. Some of them are mentioned and referenced below or in the respective chapters. The following paragraphs briefly describe the key issues in the main application areas.

Biotechnology. Major objectives in biotechnology are the industrial exploitation of micro-organisms, the use of biomolecules in technical applications and the

45 J. H. Holland, *Escaping Brittleness, The possibilities of general-purpose learning algorithms applied to parallel rule-based systems*, in **Machine Learning II**, (R. S. Michalsky, J. G. Carbonell, T. M. Mitchell Eds.), Morgan Kaufmann, Los Altos CA, pp. 593-623, 1986.

46 S. Smith, *A Learning System based on Genetic Algorithms*, **Ph.D. Dissertation**, Department of Computer Science, University of Pittsburg, PA, 1980.

47 J. Koza, **Genetic Programming (I, II)**, MIT Press, 1993, 1994.

invention of tools to automate laboratory work. Biochips[20,21,48] are mixtures of biological macromolecules that spontaneously assemble into an ordered, three-dimensional arrangement. Self-assembly of these complexes can be guided by stretches of complementary RNA. Such an association of macromolecules may then be used as a memory device. In that context, excitation of chemical groups by light or electrical stimulation can make molecules temporarily change their conformation or composition. Later, a different stimulation can be used to read the stored information or to clear it. Advantages of biochips are their ability to self-assemble and the use of inexpensive, organic components which can be produced by gene technology and micro-organisms.

Biosensors[49] consist of immobilised enzymes catalysing a reaction which consumes the compound to be measured while producing ions. A small, semipermeable tube containing the enzyme and an electrode is introduced into the probe. The reaction of enzyme and compound produces a concentration change of permeable ions. This results in a electrical signal that can be measured and amplified. The amplitude of that signal is correlated to the concentration of the compound. Biosensors are useful for monitoring concentration changes of critical metabolites in medicine. They are available for a wide range of reactions and are highly specific.

Databases. The recent increase in genetic sequence and protein structure data, and also information about micro-organisms, enzymes and chemical reactions required the design and development of specialised databases. Some established concepts for database design are: object-oriented, relational, predicate-based, or combinations of these. *Object-oriented* means that the basic items stored in the database can be accessed as abstract, semantic objects (in contrast to simple variables or strings). They can be related to each other within a class hierarchy (e.g. the class of *Escherichia coli* is a subset of the class of bacteria). Individual objects (e.g. a single *Escherichia coli* bacterium under the microscope) can inherit all characteristic class properties and methods from its class object. These properties are stored only once in the database but can be accessed through all related instances. Object-oriented databases are useful for storing knowledge about hierarchically structured domains (e.g. a taxonomy of enzymes or micro-organisms). *Predicate-based databases* store information in the form of facts (e.g. "Residues 7-17 of Crambin form a helix.") or rules (e.g. "A helix is defined by a certain number of consecutive turns."). This type of representation is convenient if an inference engine is linked to the database because then facts and rules can immediately be used for deductive inference. *Relational databases* can be visualised as a set of cross-indexed tables. They are easily maintained and allow fast access to a single item when information in the tables is kept sorted. However, if a complicated request needs to follow the links through several tables, this can slow down retrieval speed significantly. Also, expressing complex search constraints in terms of a relational database query language can sometimes become difficult or even impossible.

48 **Proceedings of the International Conference on Molecular Electronics and Biocomputing**, September 1994, Goa, India, Tata Institute of Fundamental Research, Ratna S. Phadke, email ratna@tifrvax.tifr.res.in.

49 F. Scheller, R. Schmid, *Biosensors: Fundamentals, Technologies and Applications*, in **GBF Monograph 17**, Verlag Chemie, Weinheim, 1992.

Biotechnological Applications
Biochips
Biosensors
Gel Reading Automata
Database
Object-Oriented Representation
Predicate Representation
Relational Paradigm
Processed / Selected Data
Medical Diagnosis Systems
DNA / RNA
Prediction of Gene Structure (Exon, Intron, Splice Site, Promoter, Enhancer)
Hydration and Environment
Genome Mapping
Secondary Structure Prediction
Sequence Analysis (Alignment, Search for Patterns)
Proteins
Classification according to Structure, Sequence or Function
De novo Design
Discovery of Structure / Function Relationships
Docking and Enzyme – Substrate Binding
Evolutionary Relationships
Folding Process and Motion
Force Fields and Energetics
Homology-based 3D Modelling
Inverse Folding Problem
Localising, Targeting and Signal Sequences of Membrane Proteins
Packing, Accessibility and Hydrophobicity
Prediction of Structure / Function Motifs by Multi-Criteria Comparison
Reconstruction of Backbone from Cα-Atoms
Reconstruction of Tertiary Structures from Backbone
Secondary Structure Prediction
Sequence Analysis (Alignment, Pattern Search)
Side Chains and Rotamers
Simulation of Metabolism
Solvent Interactions
Super-Secondary Structure and Hierarchy of Protein Structures
Tertiary Structure Prediction and Refinement
Toxicology
Spectrum Interpretation
Mass Spectra
Nuclear Magnetic Resonance Spectra
Planning of Experiments
Cloning
Protein Purification

Table 2: Selection of Applications in Molecular Bioinformatics.
This is a (certainly non-exhaustive) list of applications relevant in molecular bioinformatics.

DNA and RNA. Genomic information is stored in RNA and DNA. Although the basic nature of the genetic code was solved some 40 years ago[50,51], details of the organisation of chromosomes and genes are still not completely understood. Genes are known to have certain functional regions, e.g. exons, introns, splice sites, promoter sites and enhancer regions. *Exons* are DNA sequence segments that supply the information for protein formation. In eucaryotes, exons are often interrupted by non-coding portions which are called *introns* (from "intervening sequences"). Introns have to be excised to get a proper RNA copy of a gene. The boundaries of exons and introns are called *splice sites*. Reliable prediction of splice sites is desirable since this allows determination of the uninterrupted gene and the amino acid sequence of the corresponding protein (i.e. its *primary structure*). *Promoter sites* are regions in DNA that determine the start of transcription for a gene. *Enhancer elements* control the extent and speed of transcription. Important features of DNA are its *secondary structure* and interactions with a solvent. The mechanisms of the mentioned genetic elements can be described on a molecular level but these models are not yet accurate enough to allow confident prediction of gene structure in large quantities of unannotated sequences. *Sequence alignment* of DNA sequences is important for identifying homologous sequences and for searching related patterns. An immediate medical application is the development of antisense drugs that can specifically inhibit expression of malfunctioning genes. One problem with aligning sequences is the occurrence of insertions and deletions[52] which makes it difficult to determine the corresponding positions in either sequence; another problem is the exponential increase in complexity when comparing multiple sequences. *Genome mapping* is the localisation of DNA fragments, restriction sites and genes on a chromosome. This usually involves deriving a set of linear arrangements that satisfy a number of neighbourhood constraints. In practice, this is difficult because the information available is often incomplete, noisy and / or redundant. Complete exploration of all possible arrangements requires traversing a search space that increases exponentially with fragment number.

Proteins. The chromosomes of eucaryotes carry information on how to synthesise tens of thousands of different proteins. Proteins are multi-functional macromolecules, each a string made of 20 different amino acid residue types which are assembled by the ribosome and which spontaneously fold up into complicated conformations. They are used for a wide range of purposes: enzymatic catalysis of chemical reactions, transport and storage of chemical compounds, motion, mechanical support, immune protection, generation and transmission of nerve impulses, growth control and differentiation. The key to the function of a protein is its three-dimensional structure. The spatial arrangement of the atoms in the *active site* of an enzyme is made so that a substrate of roughly complementary geometry can bind to it and be subsequently modified in a chemical reaction. As was shown

50 J. D. Watson, F. H. C. Crick, *Genetic implications of the structure of deoxyribonucleic acid*, **Nature**, vol 171, pp. 964-967, 1953.

51 F. H. C. Crick, L. Barnett, S. Brenner, R. J. Watts-Tobin, *General nature of the genetic code for proteins*, **Nature**, vol 192, pp. 1227-1232, 1961.

52 Insertions and deletions are subsumed under the term "indels", as it is often impossible to decide whether an insertion in one sequence or a deletion in the other occurred.

in the Nobel-prize winning experiments by C. Anfinsen[53], the three-dimensional structure of a protein can be completely determined by its amino acid sequence. Although additional, non-spontaneous mechanisms of folding have been observed since then[54], the general view still holds that *sequence implies structure.*

The analysis of protein architecture includes comparison on the level of *primary structure* (i.e. the order of amino acids along the polypeptide chain), *secondary structure* (i.e. short stretches of amino acid residues in particularly regular conformation), and *tertiary structure* (i.e. the exact conformation of a whole protein). The so-called *protein folding problem* is to establish detailed rules defining the relationship between primary and tertiary structure. These rules could help predict conformations for the many known protein sequences of unknown structure and also for some newly invented ones. Biochemists could then reason about the function of those proteins on the basis of their predicted conformations. A general solution to this problem is not yet known but for a number of special cases algorithms for predicting secondary and tertiary structure have been developed. The reliable and accurate *prediction of secondary structure* is of interest as it allows the assembly of a whole protein in terms of almost rigid building blocks. This reduces the number of potential conformations by several orders of magnitude. Unfortunately, secondary structure prediction is rather difficult due to the effect of *long range interactions*. This means that identical short sequences of amino acids can adopt different secondary structures in different contexts, i.e. secondary structures are not purely locally defined by their sequence.

Other applications in molecular bioinformatics are the interpretation of NMR and mass spectra, and the planning of cloning and protein purification experiments.

53 C. B. Anfinsen, C. B. Haber, M. Sela, F. H. White, *The kinetics of the formation of native ribonuclease during oxidation of the reduced polypeptide chain*, **Proc. Natl. Acad. Sci. USA**, vol 47, no 9, pp. 1309-1314, 1961.

54 P. J. Kang, J. Ostermann, J. Shilling, W. Neupart, E. A. Craig, N. Pfanner, *Requirement for hsp70 in the mitochondrial matrix for translocation and folding of precursor proteins*, **Nature**, vol 348, pp. 137-143, 1990.

2. Artificial Intelligence & Expert Systems

This chapter concerns the application of expert systems and other techniques from artificial intelligence to various problems in bioscience. The methodologies described here clarify fundamental issues in symbolic information processing and illustrate their use. Section 2.1 elaborates on some key principles and algorithms in artificial intelligence programming. Section 2.2 then presents a number of applications using these techniques.

2.1. Methodology

One important feature of artificial intelligence programming and expert systems is *symbolic computation* as opposed to *numerical programming* and *string processing*. Symbolic computation is a prerequisite for implementing and manipulating knowledge bases. Typical representatives of symbolic programming languages are Lisp and Prolog. Here, we will take a brief look at programming techniques used with Lisp while programming in Prolog will be dealt with in a later chapter.

Lisp[1] is the second oldest programming language and emerged at about the same time as Fortran. During the 1970's and 80's, Lisp was run on dedicated hardware which were then rather expensive computers like the Symbolics™ Lisp machine. Today, fast Lisp interpreters are available for most common platforms at a reasonable price. However, there are two problems with Lisp. First, although there is a specification of Common Lisp[2], porting between different Lisp dialects tends to remain difficult[3]. Second, Lisp was designed to facilitate symbolic computation and not primarily numerical processing. Applications that require many floating point operations can be rather slow in Lisp. This problem can largely be overcome by interfacing Lisp to other languages, e.g. C or Assembler. Unlike C, Assembler or Pascal, Lisp allows rapid prototyping of rather complex systems through symbolic computation.

Lisp is traditionally an interpreted language which means that the user types in a command, e.g. an atom to be evaluated or a function to be called, and the result

1 J. McCarthy, Recursive Functions of Symbolic Expressions and their Computation by Machine, **Communications of the ACM**, vol 3, no 4, pp. 185-196, 1960.

2 G. L. Steele Jr., **Common LISP – The Language**, Digital Press, 1984.

3 S. Schulze-Kremer, *Common Lisp – ein geeigneter Lisp Standard?*, **Praxis der Informationsverarbeitung und Kommunikation**, vol 11, pp. 181-184, Carl Hanser Verlag, München, 1988.

appears on the screen without an explicit call of a compiler. This interactive way of programming encourages the developer to first build small components which are then used as building blocks to create more complex functions. Of course, all current Lisp systems also have compilers which are used to translate the whole system into a faster running application once it is debugged and tested. The use of an interpreted language may be helpful in keeping track of the incremental growth when developing an experimental system. Lisp is an abbreviation of list processing, or, misconstrued by those failing to appreciate its flavour, "lot's of insidious, silly parentheses".

2.1.1. Symbolic Computation

The basic data type in Lisp is a symbol, called an *atom*. The basic computational unit is a *function*. Symbols are e.g. 123.45, PROTEIN, DNA or RNA. The power of symbolic computation is based on the ease of processing semantic concepts of an underlying model instead of mere variables. Access and reference to symbols can reflect their properties, internal structure or their behaviour in response to external stimuli. Ideally, calling or evaluating a symbol displays the result of its interaction with other concepts. Properties and features of a concept can be stored in so-called *property lists*, *associative lists* or in functions. Atoms are kept in lists and a number of primitive functions are provided to process those lists. They can, for example, retrieve the first element (CAR) or the rest of a list (CDR) or they can define a new function (DEFUN). The user does not have to care about integer, double or character data types as in other programming languages. If the first atom in a list is a function name, the remaining atoms or lists that follow are treated as arguments. Atoms and lists are summarised in the term *symbolic expression*. A Lisp interpreter is the top-level Lisp function which receives input from the keyboard and then tries to evaluate a symbolic expression. If that is an alphanumeric atom, its global value is returned. If it is a numerical atom (i.e. an ordinary number), the number itself is returned. If it is a list, the function denoted by the first atom is called with all following symbolic expressions as its arguments. The arguments are themselves evaluated before being passed over to the calling function. Example 1 illustrates some basic Lisp functions.

The most intriguing fact about Lisp (and also Prolog) is that *data and program are indistinguishable*. Lists hold atoms and other lists for storage of data. At the same time, any list can be interpreted as the definition of a function or as a function call. From this follows that in Lisp *new programs* can be automatically generated and immediately executed during run time. Those programs may then, in turn, generate other programs or modify their parent programs, and so on. This capability allows the implementation of flexible, self-modifying programs that evolve and potentially learn or improve during run time. This capability of Lisp was recently used in the context of genetic programming[4]. An example for the run-time construction of a (simple arithmetic) program and its execution is given in Example 2 where a function call to add 4 numbers is created during run-time and evaluated.

4 J. Koza, **Genetic Programming (I, II)**, MIT Press, Cambridge MA, 1993, 1994.

```
Input                                    Result
-------------------------------------------------------
⇒ (+ 1 2 3)                              6
⇒ (* (+ (- (/ 55 11) 10) 9) 8)           32
⇒ (CAR '(PROTEIN DNA))                   PROTEIN
⇒ (CDR '(PROTEIN DNA RNA))               (DNA RNA)
⇒ (CADADR '(PROTEIN (DNA RNA)))          RNA
⇒ (CONS 'PROTEIN '(DNA RNA))             (PROTEIN DNA RNA)
⇒ (LIST 'PROTEIN 'DNA 'RNA)              (PROTEIN DNA RNA)
```

Example 1: Basic Lisp Functions.
Input to the Lisp interpreter on the left side produces output on the right. The first line is a function call to add three numbers. The second line is a more complicated arithmetic expression. PROTEIN, DNA and RNA are alphanumeric symbols. The function CAR returns the first element of a list, the function CDR returns the rest. CAR and CDR can be nested up to four levels in one word, where the sequence of A's and D's *read from the end of the word* identifies the order of CAR and CDR evaluation. CONS pushes a symbolic expression into a list at the first position. LIST creates a list of an arbitrary number of arguments. Most Lisp interpreters make no distinction between upper and lower case letters except when they are enclosed in double quotes.

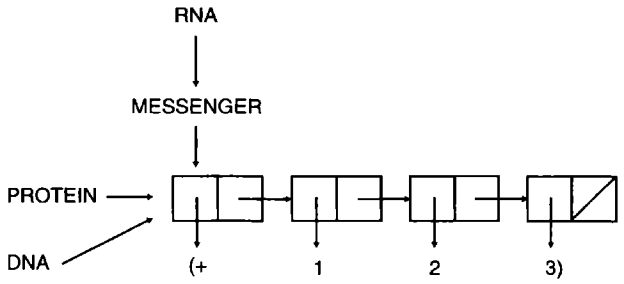

Figure 1: Lisp Pointer Structure.
This figure explains the pointer structure underlying the code of Example 2. RNA points to MESSENGER (both global symbols). A CONS cell has two pointers, one to the first element of a list, another to the beginning of the rest of the list. MESSENGER, PROTEIN and DNA point to a CONS cell, that points to the "+" sign and yet another CONS cell. Three more CONS cells define the rest of the list (+ 1 2 3).

In Example 2, the first line binds the list (+ 1 2 3) to the symbol PROTEIN. This is the standard way of defining a global variable. The second line binds *the value* of PROTEIN (that's why there is no quotation mark before PROTEIN) to the symbol DNA. The CAR (first element) of *the value* of DNA is the symbol "+". Now, the symbol RNA gets as its new value the symbol MESSENGER, which is duly echoed by the Lisp interpreter. Then, *the value* of RNA, which at this time is the symbol MESSENGER, is given *the value* of DNA. By evaluating MESSENGER in the next line we see the Lisp interpreter confirm that operation. The function EVAL does one extra evaluation. First, all arguments are evaluated before being passed to EVAL, then EVAL evaluates them once more. (EVAL 'RNA) returns MESSENGER, *as this is the value of*

```
Input                                    Result
-----------------------------------------------
⇒ (SET 'PROTEIN '(+ 1 2 3))              (+ 1 2 3)
⇒ (SET 'DNA PROTEIN)                     (+ 1 2 3)
⇒ (CAR DNA)                              +
⇒ (SET 'RNA 'MESSENGER)                  MESSENGER
⇒ (SET RNA DNA)                          (+ 1 2 3)
⇒  MESSENGER                             (+ 1 2 3)
⇒ (EVAL 'RNA)                            MESSENGER
⇒  RNA                                   MESSENGER
⇒ (EVAL RNA)                             (+ 1 2 3)
⇒ (EVAL 'MESSENGER)                      (+ 1 2 3)
⇒ (EVAL MESSENGER)                       6
⇒ (EVAL
    (CONS (CAR PROTEIN)
     (LIST (CADR PROTEIN)
           (* 4 9)
           (CADDDR PROTEIN)
           (CADDR PROTEIN))))            42
```

Example 2: Simple Automatic Program Generation in Lisp.
Input to the Lisp interpreter on the left side produces output on the right. See main text for explanation.

RNA. The quotation mark in front of RNA neutralises the first evaluation of EVAL. This is the same as if we gave the symbol RNA to the Lisp interpreter. EVALuating RNA without a quotation mark produces the list (+ 1 2 3). EVALuating MESSENGER quoted gives the same list, but for MESSENGER unquoted the list itself is evaluated once more thereby adding 1+2+3. Finally, instead of directly typing in a short list for adding four numbers, we use EVAL to assemble the function call (+ 1 36 3 2) via CONS, LIST, the CAR/CDR functions and the value of PROTEIN. After construction of that mini-program it is evaluated by EVAL and the result resturned by the Lisp interpreter. Similarly, more complex programs can be written if nested function calls with different functions are assembled (see also Example 5).

Recursive programming is easily done within Lisp. Many search algorithms can be formulated in a recursive manner. The general approach to define a recursive function is as follows. First, specify the terminating clause. This is where recursions end and a default value is returned. Then, the case with only one element left in the argument list is handled. That element is processed and its result returned. Finally, the recursive clause appears. If a composite structure is seen, it is decomposed into the head and the rest of that structure. Each part is then passed to the original function. The results of those recursive calls are then joined taking care to maintain their order as in the original call. The definition of a recursive function FLATTEN is given in Example 3. FLATTEN extracts all atoms from within an arbitrarily nested list. More on recursion in Lisp is charmingly presented elsewhere[5].

The function FLATTEN has one argument, locally named *arg*. Uppercase words denote predefined Lisp functions. COND is a conditional function which resembles

5 D. P. Friedman, M. Felleisen, **The Little LISPer**, MIT Press, 1987.