Soft Computing 5 (2001) 39–44 © Springer-Verlag 2001

Molecular approach to informal computing

K.-P. Zauner, M. Conrad

Abstract Cells and organisms are natural molecular computers. The problem domains effectively addressed by these systems are complementary to, and in fundamental respects far exceed, the domains addressable by current computing devices. The vast majority of information processing problems do not have sufficiently compact formal specifications to fall within the reach of programmable machines. Our working hypothesis is that the unique properties of molecular materials are the key to extending information processing technology beyond the narrow limits of formal computing.

Keywords Coding, Interpretation, Algorithmic complexity, Protein conformation, Malate dehydrogenase

1 Anatomy of information processing

Three choices must be be made in order to construct an information processor (Fig. 1). It is necessary to define a coding that translates the problem to be solved into input. The processing map that assigns to each input a corresponding output needs to be selected. And third, an interpretation of the output that translates the response of the processor into the solution of the problem has to be chosen. These choices are not independent. The required processing map can be simplified to any degree by the selection of the coding and the interpretation maps. The extreme case would be to encode the problems by their solutions. Another example is the smoothing of an evolutionary fitness landscape by choosing a suitable genotype–phenotype map [1]. The price for simplifying the processing is paid by reduced problem domain applicability. The greater the problem specificity of coding and interpretation the smaller the number of problems that can be addressed.

The processor can be implemented by constructing a physical realization of the processing map, or alternatively a physical system can be selected and the coding and interpretation tailored to its behavior. A computer is a system that, starting from a state which encodes a problem description, will change following the laws of nature to a state interpretable as the solution to the problem. The fixed parameters in a scientific experiment can, from this broad point of view, be taken as the problem description and the values measured interpreted as the solution to the problem [2]. Correspondingly, a conventional computer can be seen as an apparatus for performing complex electrodynamical experiments, the progress of which is monitored through an array of pixel-size measurement instruments. The user's interpretation of the measured values endows the experiment with significance that extends beyond the system's behavior in and of itself.

In general it is advantageous to use codings and interpretations that are widely applicable. The realization of the processing map therefore becomes an intricate issue. The common approach today is to use a general purpose automaton together with a formal specification of the input–output map. This method however, can only accommodate a small fraction of information processing problems, as will be shown immediately below.

2 Limits to formal computing

The processing map will here be taken (in the case of deterministic systems) as a function from input patterns to output patterns. The input and output patterns are temporal or spatial arrangements of elementary inputs and outputs. Thus we can picture the processing map as a table with two columns, one for arguments and one for values. The arguments column lists all possible input patterns and the values column the corresponding output patterns. We will view a program as a formal specification of the processing map relative to either a physical system that can implement it, or relative to a formal system that can be implemented by a physical system. The formal system is equivalent to a user manual that defines the semantics of the program. But note that knowing the program and the user manual does not in general mean knowing what processing map it specifies.

The processing map and the program are abstractions. The goal is to obtain the physical implementation. There are two possibilities. We can select and use a specific physical system and choose a coding and interpretation. Interpreting the image at the focal plane of a lens as a Fourier transform is an example. Or we can engineer the system to conform to a desired user manual. It is convenient to engineer a general purpose system, one that can be used to implement many processing maps. In this case there must be some state change to choose which map is

K.-P. Zauner, M. Conrad (✉)
Department of Computer Science,
Wayne State University, Detroit,
MI 48202, USA
E-mails: kjz@cs.wayne.edu, conrad@cs.wayne.edu

This research was supported by the National Science Foundation under Grant Nos. ECS-9704190, CCR-9610054 and EIA-9729818.

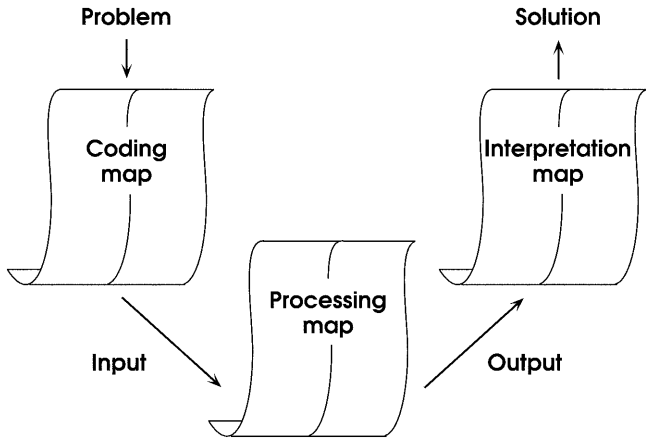

Fig. 1. General scheme of information processing

implemented. We here view this state setting as the program. We could alternatively view the state setting as input and the processing map as specific, but this is not the natural view for a system that can be used in a general purpose fashion. An architecture will be called structurally programmable if programs can be encoded in it using a simple user manual.

Consider first the case of a specific physical system. The processing map is specified by the equation of motion together with the initial and boundary conditions. Given the conventions used here these conditions serve as the input. If the system is used for computing something other than itself it is necessary to find a useful coding and interpretation, as in the Fourier transform example noted above.

With a general architecture it becomes convenient to implement any map that has a short formal specification on it. But care must be taken here. Such an architecture pre-selects the possible processing maps that can be implemented. The structural programmability feature requires a formal description that is compact enough to be communicated to the machine.

Let us put this in quantitative terms. Suppose that we require a processing map to respond to n input bits with an m-bit output. The map thus assigns to each of 2^n possible inputs one of the 2^m possible outputs. The number of such maps is $2^{(m2^n)}$. To select a particular map in general requires $m2^n$ bit long specification (where, as above, we view the bit sequence that selects or specifies the map as the program). This is clearly infeasible even for moderate problems [3, 4]. Consider, for example, a 100×100 bit input image that is to be classified whether it does or does not contain a certain feature, i.e. a one bit output. Specifying the processing map for this problem may require 2^{10000} bits!

There are exceptions. Some of the $m2^n$ bit long specifications may be expressed in programs that are far shorter. However, there exist far more long specifications than short programs. This follows directly from considerations of algorithmic complexity theory [5, 6]. A program that is shorter than the mapping it specifies can be considered as a compressed form of the map. The algorithmic complexity of the majority of strings of length n is approximately n, i.e., most strings are not to any substantial extent compressible [7]. As a corollary most processing maps do not have short specifications. This means for almost all large processing maps, there exist no programs of feasible length.

We can be more specific. Take the maximum feasible length of a program as b bits. Then the fraction η of processing maps with n-bit input and m-bit output that can be specified with a program is

$$\eta = 2^{(b-m2^n)}$$

where b is the bound on the practical length of programs. The fraction of maps that can be specified by a program thus decreases exponentially with n, the number of input bits. The implication is this: there is no hope of programming arbitrary processing maps with more than log b bits of input on a general purpose computer. Some maps with more input bits may be programmed; but most cannot be.

The situation can be summed up thus. Compression is sometimes possible, but only rarely, since there exist so many more long processing maps than short programs. The inescapable fact is that most processing maps cannot be specified by programs of acceptably short length. But this does not mean that physical realizations of such maps are impossible. We take it as the challenge for future computing technology, in particular technologies of a molecular nature, to find implementations for maps that do not have compact formal specifications.

3 Transcending the limits

Why might such maps be interesting? Clearly humans and organisms perform numerous activities that are entirely out of reach so far as today's artificial intelligence and simulation programs are concerned. Pattern and object recognition in an ambiguous environment are examples. Language processing, creative mathematical work, design and creation of computer programs are higher level examples. There are at least three reasons for supposing that capabilities such as these require processing maps that do not admit a sufficiently compressed formal description to be communicated to a structurally programmable machine.

The first is what might be referred to as situational transformability. Suppose that we can provide a precise description for a system, say a tool such as a hammer. Then there is little doubt that we could express this description in a formal computer program. But in fact we cannot give such a description because there are practically an indefinitely large number of situations that the hammer can be in and an indefinitely large number of purposes it could be used for. It could be used as a judge's gavel or as a weapon, or as a way of testing knee reflexes, and so on. No finite description exists that is complete for the simple reason that new contexts can never be excluded. Evolutionary approaches are open to seeing the hammer in de novo contexts. This is exemplified by recent experimental work showing that evolving electronic circuits can exploit physical effects ignored in the user manual description of the components [8]. Evolution is free to discover and utilize such effects, whereas this is a priori excluded in designs that are based on finite user manual descriptions. The evolutionary process here exploits context sensitivity of the componentry that the engineer has attempted to suppress.

The second reason is connected with degree of context sensitivity available. This has been limited as much as possible in engineered systems. Natural systems are not subject to such constraints. For evolution high context sensitivity is a resource so far as realizing complex processing maps is concerned. Recognizing an object in a complex background is the type of task that requires such a processing map. The computer languages that we use to program conventional machines are context free. Context sensitivity has to be built in at a virtual level, therefore simulated on top of the base machine. This implies a program of great length. The context sensitivity exhibited by organisms and humans in their daily activities may be so high that the processing maps required have no formal specification of practical length, and therefore admit no program description that could be communicated to a general purpose computer.

Underlying the above functional difference is the third reason: the structure-function relations of organisms are radically different than those of today's general purpose machines. The latter are built out of components that can be given a static, context free description. The structure-function relations are programmable. The program that formally specifies the processing map is encoded in the states of the components and connections among them according to simple, definite rules. This is a powerful feature, since it allows the user to prepare the processing map without considering the physics of the system. But there is a price in terms of efficiency and evolvability [9, 10]. The vast majority of interactions that could contribute to implementing the processing map must be frozen out, otherwise the requisite context freedom of the components will be lost. Additionally the processing map implemented is related to the state of the machine in an extremely fragile manner, since any variation in the state is tantamount to a variation in the formal specification of the processing map. This follows from the fact that a one bit change in the compressed description of the processing map, M, will in general result in a compressed description of a new processing map, M', that is many bits different from M. The advantage of structural programmability is that the system's state can be set to execute any processing map that is describable by a program that can be fit into it. Nonprogrammable systems have the advantage that they can realize processing maps whose formal description cannot be fit into any structurally programmable machines.

If a system is structurally programmable it has a compressible description. All the underlying physics is rendered irrelevant. Clearly the vast majority of systems are not structurally programmable, just as the number of compressible descriptions is very much smaller than the number of noncompressible descriptions. The reasonable assumption is that the human brain is capable of performing the highly context sensitive, situationally transformable tasks indicated above because it is an essentially noncompressible system so far as formal description is concerned, or at least far less compressible than a structurally programmable machine.

Our working hypothesis is that it is through structurally nonprogrammable systems that the limits on formal computing can be transcended. Biological macromolecules, because of their context sensitivity and consequent situational transformability, have rich potentialities in this respect, and it is to these that we now turn.

4
Dynamic molecular approach

Let us focus our attention on proteins, in particular proteins with enzymatic functionality. The main job of the enzyme is to recognize a specific substrate molecule in a complex milieu and to make or break a particular bond in this molecule to form a specific product. How fast the enzyme works is sensitive to some milieu molecules and indifferent to others. The presence or absence of particular molecules (control molecules) may be necessary in order for the enzyme to recognize and act on the substrate. Other milieu molecules may exert modulatory influences. The enzyme, in short, is a context sensitive pattern recognizer.

Our goal is to capture this inherent context sensitivity for higher level signal processing, the kind that could for example couple to an electronic machine. Consider a conceptual device (Fig. 2) that illustrates how this can be done and that serves as the basis for our laboratory prototype. The device contains enzymes embedded in a chemical medium. Input signals are coded into chemical signals (e.g. various types of ions and organic molecules) that are injected into the medium. The pattern of input signals is thus transduced to a milieu context that determines the rate at which the enzymes convert substrate to product. Optical monitoring of the progress of the reaction is used to determine the output. The symbolic signal pattern recognition problem is thus solved by utilizing the physical mechanisms of biochemical dynamics.

The whole reaction medium implements the processing map. The coding is determined by the choice of signal substances to be injected and the manner in which the elements of the input situation are represented by these signals. The result of the computation is determined by the

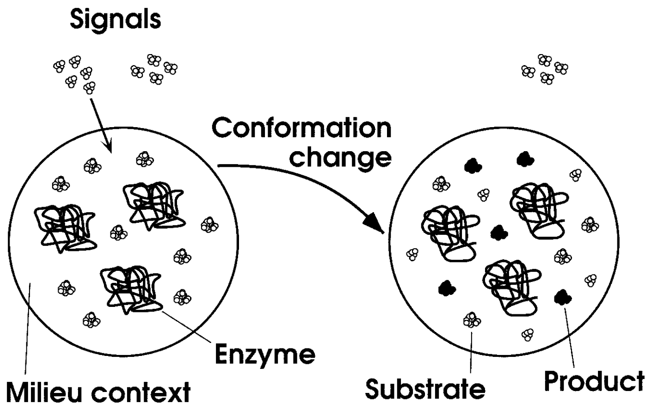

Fig. 2. Schematic picture of conformational pattern processing

manner in which the user interprets the spectroscopic measurements on the reaction medium. For example, a different choice of wavelength or threshold can determine a different classification of the input signal pattern. The processing map can be altered by choosing different initial milieu conditions or altering the enzyme. Altering the type and concentration of chemicals used as input signals corresponds to the choice of coding. Different specific systems can be prepared by altering the reaction conditions or the enzymes used. Choice can also be exercised over what components of the reaction medium are to be monitored for output. This corresponds to selecting the interpretation map. If a multienzyme medium is used the different enzymes will be linked by their effects on the reaction milieu and by any metabolites that they share as substrates. The whole dynamics then becomes increasingly rich and increasingly amenable to different interpretations (which is another way of saying that it is more situationally transformable from the point of view of the user).

The dynamics are clearly not of the structurally programmable type. There is no way of knowing in advance what the result of a different choice of milieu conditions will be, and similarly for different choices with respect to enzymes, signal substances, and features tracked for output. An essentially evolutionary methodology must be used to obtain desired input–output behavior.

So far we have taken the pattern recognition capacities of enzymes as a phenomenological fact. It is instructive to consider what it would take to duplicate this using a structurally programmable machine. The main fact is that the enzyme forms a complex with the substrate, largely on the basis of dynamic shape complementarity [11]. This means first of all that the enzyme and substrate explore each other's shapes. The vast random search power inherent in the heat bath is brought to bear. It is of course brought to bear in a highly coherent way by the the various conformational motions of the protein. Here we have typically between 200 to 400 amino acids, with roughly 60 electrons per amino acid. The enzyme can thus be thought of as a vast processing network of nuclei and electrons that fuses the input signals through its conformational dynamics. All the milieu factors that affect these dynamics determine the rate at which the enzyme draws in substrate and releases product. The whole process utilizes the heat bath as powerful search engine. The enzyme selectively exploits the complexity of the heat bath by migrating through a series of low energy conformational states [12]. Quantum features of loosely bound electrons likely speed up the search process [13, 14].

Let us now imagine that we can write down the quantum mechanical equation of motion (the Schrödinger equation) for the device. Clearly solving this equation is out of the question; treating even a small cluster of particles ab initio by the Schrödinger equation is out of reach for the most powerful supercomputers. We take this difficulty as a foregone conclusion. Our question is whether we could fit the equation into a structurally programmable machine. On the surface it might seem that the Schrödinger equation provides a compact physical description and therefore all physical-dynamical systems should have short formal descriptions. But in fact the potential function, if written out in detail, would have an enormous number of terms in it, due to the vast number of particles in the whole reaction medium that contribute to the time evolution. Specifying the initial and boundary conditions must be included as part of the system description. Of course many of the terms in the potential function and therefore many of the initial-boundary conditions, can be ignored or lumped into equivalent groupings. In general the system's description can be compressed to some degree. But the degree of compression is never as much as in the extreme case of a structurally programmable machine. So many of the interactions are irrelevant in programmable systems that there is no need to consult physical equations at all. This is the whole point. The description of a structurally nonprogrammable enzymatic reaction system is much less compressible than the description of a structurally programmable machine with a comparable number of particles. The goal of nonprogrammable computing is to employ this difference in complexity for useful computational functionality.

We must emphasize that a statistical description of a physical system in terms of distribution functions (e.g., the Maxwell–Boltzmann distribution) is not a compression in the above sense. Taking averages eliminates the complexity of particular cases. It would be impossible to retrieve the complexity of even a single instance from a statistical characterization. This is exemplified by the fact that it is impossible to use a statistical distribution function as a practical source of complexity in a computer simulation.

5 Towards implementation

We have constructed a table top prototype along the above lines that allows for convenient testing of reaction conditions, including variant signal codings and output interpretations (Fig. 3). Signals encoded by carrier substances combine in a mixing chamber to form a chemical milieu. The input pattern is represented in the milieu as a linear combination of the substance concentrations present in the signal reservoirs. The chemical milieu also contains reactants. The concentration of these is independent of the input signal patterns. The addition of an enzyme/coenzyme solution to the contents of the mixing chamber initiates a chemical reaction catalyzed by the enzyme. The fluid reaction phase is pumped in an air-filled tubing from the mixing chamber to the detector. Product is formed

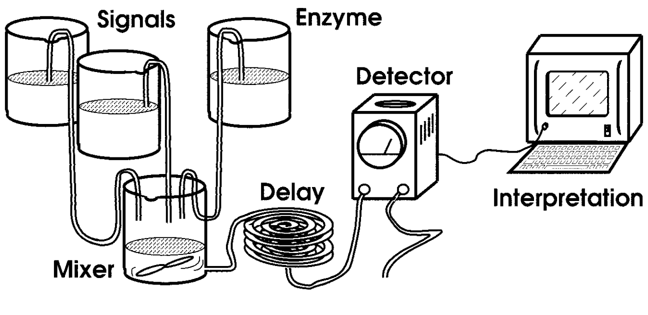

Fig. 3. Principal components of an enzyme-based pattern processor

while the reaction phase is traveling and is monitored when it reaches the detector. The measured product serves as the output signal. The activity of the enzyme determines the speed of the reaction and hence the amount of product present at the time when the output is measured. The interaction of the enzyme with the chemical milieu (which contains the input signals) affects its conformational dynamics. The conformational effect of the input signals could be direct or indirect (i.e., through effects on the medium). In either case the processing of the input signals occurs through their nonlinear effect on the enzymatic activity.

The actual setup is shown in Fig. 4. Two types of signal solutions, representing 0 and 1, are contained in bottles labeled 0 and 1 in the photograph. Each of the bottles is connected to two valves and two syringes that form a manual two-piston pump. The input signal pattern is delivered to the mixing chamber by means of the syringes. An additional single piston pump is used to inject the enzyme solution into the mixing chamber. A peristaltic pump is used to transport the reaction phase from the mixing chamber to a spectrophotometer that serves as the detector. The time for the reaction medium to reach the detector, which determines the response time, is ten seconds in the current setup. The spectrophotometer is controlled by a digital computer which determines the timing of the measurement and determines the output by thresholding the amount of product measured. This corresponds to the interpretation map. Several valves allow the system to be flushed with water before the next pattern is processed. The entire system could be miniaturized using lab on a chip technology [15, 16]. The speed of the process would be limited only by the turnover rate of the enzyme and the detection limit, and could be far higher than is possible in our current setup. However, at this stage our interest is not miniaturization and speed but finding biochemical media that perform interesting information processing functions.

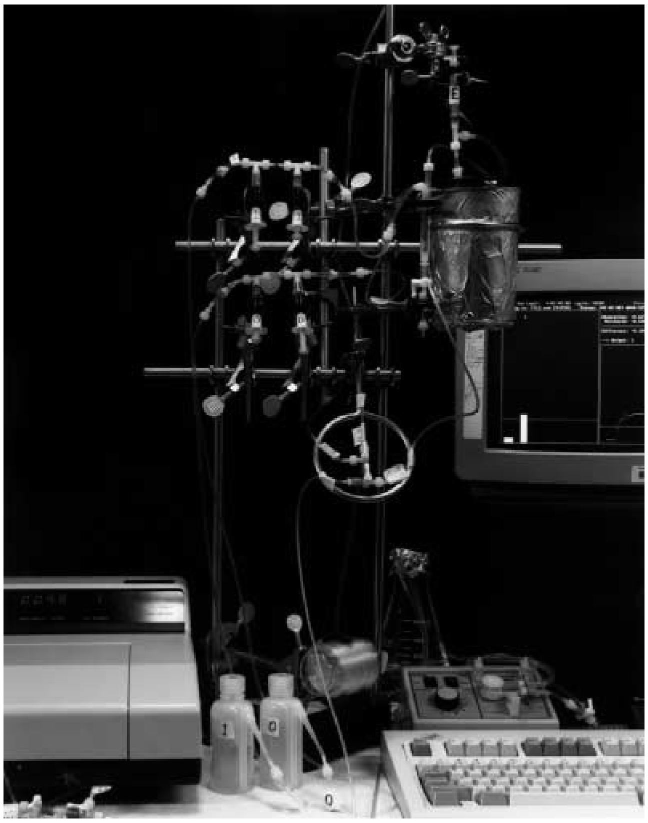

Fig. 4. Table top implementation of the enzyme-based pattern processor outlined in Fig. 3

We have used the enzyme malate dehydrogenase for simple two bit pattern processing. The simplest linearly inseparable task is the exclusive-or operation. This requires that the enzyme respond in a nonmonotonic way to the substance used as the signal carrier, i.e., it should be activated more by an input pattern that contains one dose of signal substance than by two doses. We used $MgCl_2$ as the signal carrier. $CaCl_2$, with slightly reduced output signal strength, can be substituted for $MgCl_2$. The processing map mediated by malate dehydrogenase thus serves as a transform that converts a linearly inseparable problem to a linearly separable problem [17]. Thus the final classification, done by the digital machine, can be performed with a single threshold. We can note that the exclusive-or problem (and linearly inseparable problems generally) cannot be solved by a single layer perceptron [18] or by any individual element, such as a transistor, that responds to the average of its signal inputs. Whether natural biological neurons exploit the recognition power of enzymes to group different patterns of presynaptic input is an important open question.

Malate dehydrogenase can be used to perform other two-bit pattern classifications (e.g., OR, AND, NAND) by changing the concentrations used for the signal coding and changing the threshold level used for interpretation of the output. Alternatively by using both $CaCl_2$ and $MgCl_2$ to encode signals and changing their proportion it is possible to shift between different logic operations without changing the coding or interpretation. This shows that the function performed by malatate dehyrogenase depends on chemical context and that the function is therefore situationally transformable.

6 Concluding remarks

Our prototype is as yet very simple. Obviously it does not perform any operation that could not be performed by a general purpose computer. The compression arguments presented earlier clearly demonstrate the limits of programmable general purpose machines. Biological cells, with thousands of interlinked enzyme types, undoubtedly transcend these limitations. The inhomogeneity which is the source of complexity here extends to enzyme structure. The question is whether this can be exploited in devices that expand on the complexity of the biochemical media used in our prototype. Expansion here means going from individual enzymes to networks of enzymes interacting directly or through a common reaction medium.

The price of entering the space of possibilities lying beyond what is formally specifiable is that prescriptive programmability has to be abandoned. The complexity of incompressible processing maps means that a complex physical-dynamical system is required for implementation. Planned designs are infeasible. Evolutionary and adaptive approaches are called for.

What might this domain of possibilities be like? The main fact is that its characteristics are not formally specifiable. The types of problems that could be usefully addressed would therefore be quite different from those addressed by programmable systems. Problems of the type that hard artificial intelligence has stalled on presumably belong to this class. The softness of natural language, as opposed to the precision of computer languages, is perhaps the best pointer. We envisage that high complexity molecular devices could serve as co-processors that complement the formally specifiable capabilities of conventional machines.

References

1. **Asselmeyer T, Ebeling W, Rose H** (1996) Smoothing representation of fitness landscapes – the genotype–phenotype map of evolution, Biosystems **39**: 63–76
2. **Stibitz GR, Larrivee JA** (1957) Mathematics and Computers, chapters 2 & 9. McGraw-Hill, New York
3. **Conrad M, Hastings HM** (1985) Scale change and the emergence of information processing primitives, J Theor Biol **112**: 741–755
4. **Zauner KP, Conrad M** (1997) Conformation-driven molecular computing: The optical connection, Optical Memory and Neural Networks **6**(3): 157–173
5. **Chaitin GJ** (1966) On the length of programs for computing finite binary sequences, J Assoc Comput Mach **13**: 547–569
6. **Li M, Vitányi PMB** (1997) An Introduction to Kolmogorov Complexity and its Applications. Springer, New York
7. **Chaitin GJ** (1974) Information-theoretic computational complexity, IEEE Trans Inf Theor **20**: 10–15
8. **Thompson A, Layzell P, Zebulum RS** (1999) Explorations in design space: Unconventional electronics design through artificial evolution, IEEE Trans Evol Comp **3**(3): 167–196
9. **Conrad M** (1988) The price of programmability. In Herken R (ed) The Universal Turing Machine: A Fifty Year Survey, pp. 285–307. Oxford University Press, Oxford
10. **Conrad M** (1995) Scaling of efficiency in programmable and nonprogrammable systems, Biosystems **35**: 161–166
11. **Yon JM, Perahia D, Ghélis C** (1998) Conformational dynamics and enzyme activity, Biochimie **80**: 33–42
12. **Frauenfelder H, Park F, Young RD** (1988) Conformational substates in proteins, Ann Rev Biophysics and Biophysical Chem **17**: 451–479
13. **Conrad M** (1992) Quantum molecular computing: the self-assembly model. Int J Quant Chem: Quant Biol Symposium **19**: 125–143
14. **Conrad M** (1994) Amplification of superpositional effects through electronic-conformational interactions, Chaos, Solitons & Fractals **4**: 423–438
15. **Hadd AG, Raymond DE, Halliwell JW, Jacobson SC, Ramsey JM** (1997) Microchip device for performing enzyme assays, Anal Chem **69**: 3407–3412
16. **Chohen CB, Chin-Dixon E, Jeong S, Nikiforov TT** (1999) A microchip-based enzyme assay for protein kinase A, Anal Biochem **273**: 89–97
17. **Ellacott S, Bose D** (1996) Neural Networks: Deterministic Methods of Analysis, chapter 2. Intern. Thomson Computer Press, London
18. **Minsky ML, Papert S** (1969) Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, MA