1 Conformation-Based Computing: A Rationale and a Recipe

Michael Conrad and Klaus-Peter Zauner

1.1 Objectives

Biological systems possess enviable information processing abilities, which are rooted in the self-organization of context-sensitive building blocks. Molecular computing can utilize this principle. Our objective in the present chapter is to show that this opens up a realm of information processing that is inaccessible to programmable machines. Our second objective is to present a tabletop prototype that illustrates a methodology for pursuing this direction.

Algorithmic complexity theory provides a framework for elucidating the comparative capabilities of programmable and nonprogrammable systems. Programmable architectures are amenable to a more compressible description, concomitant to the fact that they must conform to a simple user manual. To implement complex input-output behavior, it is necessary to supply a complex program. The programmer therefore must be the source of complexity. Biomolecular architectures are sharply different: Complexity is inherent. The capabilities are constructed by orchestrating a repertoire of complex components through an adaptive process. The number of functions that can be implemented is limited by the time available for adaptation and may not be larger than that in programmable systems. In this chapter, we will argue that the complexity of the actual achievable behavior is greater.

John von Neumann (1951) referred to such noncompressible complexity in a discussion of the visual cortex:

It is not at all certain that in this domain a real object might not constitute the simplest description of itself, that is, any attempt to describe it by the usual literary or formal-logical method may lead to something less manageable and more involved. (1951, 24)

In our case, the real objects are proteins. We will show that it is possible to utilize the conformational dynamics of proteins to process input signal patterns—though at this stage not in a manner that transcends formal description.

1.2 Algorithmic Complexity Rationale

Digital computers are commonly referred to as general purpose machines. The seeming implication is that with sufficient memory and operating speed it should be possible to implement any computable process on such a machine. The concept of computation universality, originally expressed in terms of the Turing model of computation, captures this idea. For the present purposes, the Turing formalism can be equated to a digital machine with no a priori limit on available memory and time. Such an idealized machine would be capable of computing any computable function. Realizable machines are, of course, finite. The memory available may not be sufficient to perform the desired computation; or the computation might require an unacceptable span of time. Here we are especially concerned with a further limitation: The size of the program that can be presented to the machine is also subject to practical restrictions.

The above distinction, between limits on processing capacity and program size, has an important implication. Even if processing speed and memory space could be increased indefinitely, a large class of information processing tasks would still be inaccessible. The programs, or maps describing the input-output behavior of the system, can be too large to practically specify.

Let us take as a computer any system that, starting from a state that encodes a problem description, will change to a state interpretable as the solution of the problem. The limited precision and limited dynamic range of the computer's components, together with the requirement of a finite response time, restrict any computer to a finite set of discernible inputs and a finite repertoire of outputs.

A deterministic computer is a physical realization of a function that takes an input signal pattern as argument and returns as the value the associated output signal pattern. To make the computer perform a desired task, it is necessary to specify the appropriate function. The specification may be provided explicitly by programming or, in the case of an adaptable system, implicitly through training. In either case, the specification has to select the desired system behavior from the set of potential behaviors.

Consider a deterministic computer that is supposed to respond to each n-bit input pattern with an appropriate m-bit output pattern. The function that maps the input into output can, in principle (and for small values of n also in practice), be described by a table. The table would have 2^n rows, one for every possible input, and each row would contain the pattern that the computer should output in response to this input. Programming a computer requires that the table it should implement be communicated to it.

The amount of information necessary to specify the input-output map is given by the number of bits needed to select one specific table from the set of all possible tables. There are 2^n rows corresponding to the possible inputs in the table, and any one of the 2^m possible outputs may be assigned to each row. This gives rise to $2^{(m2^n)}$ possible tables. Selecting an arbitrary table from this set requires a specification that is $\log_2[2^{(m2^n)}] = m2^n$ bits long (Ashby 1968). The important implication is this: Even for input patterns of very moderate size, it will almost always be impossible to program a computer to perform a map arbitrarily selected from the set of possible maps. For example, consider a pattern of the size of a single character on a computer screen, say 10×10 black and white pixels ($n = 100$ bits), and suppose we want to classify such tiny images according to whether or not they contain a certain feature (meaning that $m = 1$ bit). This could require a program 10^{20} gigabytes in length.

On the surface, it might seem that for any particular job required, it should be possible to devise an appropriate program of practical size. The following considerations from algorithmic complexity theory reveal that programming a "general" purpose computer is in fact practical only in very special situations.

In the example considered above, every row of the table that describes the classification of the 10×10 pixel images has a 1-bit entry indicating the presence or absence of the feature. The content of the table corresponds to a binary string of length equal to the number of rows in the table. Chaitin (1966) asked the question: How long would a program need to be in order to generate such a sequence? For our purpose, we can take the ability to generate the contents of the table as equivalent to the capacity to implement the input-output map described by the table. Some classifications have short programs. If we want each input image to be classified according to whether it is all black, then all but one row in the table will contain the same bit. A program much shorter than the explicit table will be sufficient to generate the table. This corresponds to the fact that the table is highly compressible—the program being a compressed description of the table. The algorithmic complexity of the table is defined as the length, up to an additive constant, of the shortest program required to generate it (Li and Vitányi 1997). The additive constant reflects differences in machine architecture that, from a practical point of view, can have an immense impact as the constant becomes large (Kampis 1991).

For most tables, no significant compression is possible, as can be seen from a simple counting argument (Chaitin 1974). Under the assumption that (due to the capacity of the machine or its programmers) the longest practical program is limited to a length of b bits, there exist only 2^b distinct programs. The fraction η of tables describing n bit inputs mapped to m bit outputs, which can be compressed to a b-bit-long specification, is therefore at most

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

Furthermore, this maximum value of η can only be achieved if the machine architecture is not degenerate in the sense that two or more distinct programs yield identical input-output behavior.

The above equation shows that, in practice, only a very small fraction of the conceivable information processing tasks can be implemented by programming a putatively general-purpose computer. However, the compressibility of the tables is relative to the machine architecture on which they are specified. Different architectures can bring different input-output behaviors within reach of practical specifications. An extreme example would be a machine specifically constructed to solve a single large problem instance (Zauner and Conrad 1996).

Every realizable information processing machine can only implement a small subset of the possible input-output transforms and is therefore a special-purpose device (Zauner and Conrad 2001). The common computers, often naively assumed to be general purpose, are in fact specialized devices that have been designed to implement the narrow class of highly compressible input-output maps.

1.3 Trade-Off Principle

The comparative limits of programmable and nonprogrammable architectures can be stated in terms of a trade-off principle: Programmability, efficiency, and evolutionary adaptability are incompatible. A system, to achieve high programmability, must trade off efficiency and evolvability.

A computing system is programmable if the initial state and a chosen set of formally defined state transition rules can be explicitly invoked. The programmer communicates the intended relations among the system states to the system, which in turn interprets the rules in rigid adherence to a finite user manual. If the programmability is bound into the material structure of the system, we will refer to it as structural. Material physical systems generally have self-organizing dynamics, hence a will of their own that is incompatible with prescriptive programmability. The computer designer must quench these self-organizing aspects in order to achieve a physical realization of a formal system. Information processing systems, however, do not need to be programmable; functionality can be molded through adaptive procedures.

We can phrase the programmability-efficiency trade-off in terms of interactions. To be as generous as possible, let us make the assumption that elementary particles can serve as active components in a computing system and the system contains n such particles. The potential function of the system can call on as many as n^2 interactions. If the system is structurally programmable, the input-output behavior of components should remain the same as more components are added. This is only possible if the components have a fixed number of possible inputs. Thus the number of allowable interactions scales as Cn, where C is a constant. The fraction of interactions available for problem solving falls off as C/n as the number of components increases. If the system is run in a serial mode, therefore in an effectively programmable mode, the falloff is even faster (i.e., as K/n^2, where K is the number of components that can be active at any given time). If quantum features are pertinent to the system's problem solving, interference effects among the possible states of the particles must also be considered, further increasing the disparity between the potential complexity of natural systems and systems configured to be structurally programmable. The assumption that single particles could act in accordance with a finite user manual is of course quite unreasonable. As the number of particles per component decreases, it becomes increasingly likely that the system will self-organize in a way that escapes a simple user manual description (Conrad 1995).

The trade-off principle is intimately connected to the compression issues considered in section 1.2. The salient point is that all structurally programmable architectures must have a highly compressible description in order to conform to formal rules specified in a simple user manual. Constructing a formal component calls for a large number of particles, because this requires quenching of self-organizing characteristics that deviate from the user manual. A large number of such formal and hence low-complexity components is needed to build a system with complex behavior. Efficiency in terms of the necessary number of particles will therefore be low. In short, to make a heavyweight architecture out of lightweight components, the system must be large.

The conflict between structural programmability and evolutionary adaptability can also be understood in terms of compression. In a program that is a highly compressed description of the system's behavior, a change in any single bit will, in general, have radical effects on the behavior of the modified program. The program ordinarily describes an input-output table that is much larger than the program. Any bit modification in the program will, in general, alter many bits in the input-output table. Of course, the uncompressed input-output table can always be changed gradually (bit by bit). But it is only possible to act on this table through modifications of the program—hence the gradualism requirement for evolutionary adaptability cannot, in general, be satisfied. If biological systems were amenable to a highly compressed description, they would a fortiori be unsuitable for evolutionary adaptation.

The trade-off principle does not assert that structural programmability absolutely precludes evolutionary adaptability. Biological systems in nature are clearly highly evolvable. In principle, it should be possible to use a structurally programmable machine to simulate the structure-function plasticity that allows for this evolvability. As long as mutations are restricted to the virtual level, rather than to the program as encoded in the state of the base machine, it would be possible to duplicate the requisite evolvability. But this comes at a computational cost; the computational work required to simulate plastic structure-function relations puts a severe practical limit on the degree of evolvability that can be retained. In effect, the simulation program is a decompression of some highly compressed program that could do the same job as the simulated system. The decompression, if appropriately introduced, reduces the fragility of the program.

The decompression has an equivalent in the interaction picture. Redundancy in the number of components and interactions among them serves to buffer the effect of mutation on features of the system critical for function (Conrad 1979). This is not an entirely general fact; it is restricted to a subclass of systems with self-organizing dynamics. Protein folding, in particular, fits this picture. As the length of the amino acid chain increases or as more amino acids with similar properties are available for substitutions, the chance that a mutation will be acceptable increases. Without self-organization, the introduction of redundancy would only yield fault tolerance, not the topological distortability necessary for transformation of function (Conrad 1983).

The structure-function relations that enable high efficiency and high evolvability require context-sensitive components. This sensitivity of the components' behavior to their environment is in sharp contrast to the precisely defined and therefore context-free components of structurally programmable systems. Nevertheless, networks of context-free components run in a parallel mode can also exhibit self-organization, as in the case of artificial neural networks. The self-organization, however, causes a loss of effective programmability. With the main advantage of rigidly defined components lost, there is no reason to restrict the architecture of the network to context-free components. Instead, context-sensitive components that open the path to high efficiency and high evolvability can be employed.

The trade-off principle suggests that there are two sharply different modes of computing: the high programmability mode versus the high efficiency, high adaptability mode. Biological systems, because they are the products of evolution, must operate in the latter. The remainder of this chapter will focus on initial concrete steps in the direction of artificial systems that operate in the biological mode.

1.4 Pertinent Molecular Properties

The trade-off principle asserts that systems with nonprogrammable structure-function relations are capable of implementing transforms that are too complex to embody in general-purpose (programmable) architectures. The physical dynamics of such systems, suitably interpreted, effectuates the computation. Conceivably many types of physical dynamics could be utilized in this manner. Macromolecules afford a particularly powerful combination of properties (see table 1.1).

The main property is folded shape. This requires long, nonconjugated polymers (because rotation around single bonds is necessary). Carbon, the atom of life, sup-

Table 1.1
Computationally important properties of macromolecules

Property	Draws on	Confers
Folded shape	Long flexible chains, weak bonding, rotation around single bonds	Specificity, self-assembly
Conformational dynamics	Folded shape	Milieu sensitivity, allosteric control
Well-defined ground state	Individual molecules (not statistical ensembles)	Precisely duplicatable nonlinearity, specific shape
Brownian motion	Specific shape, low mass, heat bath	Cost-free search
High evolvability	Combinatorial variety, high dimensionality	Diverse repertoire of specialized functions
Specificity with speed	Defined shape, Brownian motion	Low dissipation pattern recognition
Supramolecular structure	Self-assembly, free energy minimization	Rich, extended 3-D architecture
Diverse specificities	Building block principle, heat bath, folded shape	Heterogeneous organization, dynamic complexity

ports this requirement. Silicon, the only competitor for carbon in this respect, is rather inferior (Henderson 1913; Conrad 1994b).

The C-C bond energy is about the same as for bonds with H or O. The energy required to break the Si-Si bond is only about half as much as the energy required to break Si-H and Si-O bonds. The number of carbon-based structures that are possible is accordingly much greater than is possible with silicon (Sidgwick 1950; Edsall and Wyman 1958). The longer chains possible with carbon allow for a greater variety of folded shapes.

The well-known lock-key metaphor (Fischer 1894) for enzyme-substrate recognition is based on this fact of folded shape. Proteins must be big enough to have significant shape features (not true for individual atoms) but small enough to scan each other's shapes through diffusion (which we can refer to as *Brownian search*). The shape fitting is in reality dynamic; conformational motions are critical to the rate of complex formation and (in the case of catalysis) complex decomposition. The conformational motions are sensitive to a variety of milieu features (e.g., temperature, ions, control molecules). The prototype device that we will shortly turn to utilizes this context selectivity for signal pattern recognition.

As in all chemical reactions, thermal fluctuation (heat motion) is sine qua non. The term *Brownian search*, used above, is intended to suggest its computational significance. Recall the discussion of complexity: Complexity must either be provided in a program fed to a system from the outside or it must have self-organizing dynamics, therefore nonprogrammable structure-function relations. Protein folding and complex formation are prime examples. The heat bath is a potent source of complexity. The amino acid sequence draws on thermal fluctuations to explore itself in the folding process. The folded structure draws on thermal fluctuations to explore molecules with which it interacts in the complex formation process. In general, physical self-organization is based either on energy minimization or entropy maximization. The randomness of the heat bath is an essential ingredient in both cases. If entropy maximization is the controlling feature, the fluctuations allow the system to assume a greater number of structural forms. If energy minimization dominates, thermal energy must be given up to the heat bath in an irreversible way. From the point of view of algorithmic complexity theory, the complexity of a pattern or process increases as the size of the shortest program required to generate it increases—that is, as its description becomes less compressible. Of all phenomena considered in physics, perhaps the heat bath has the most incompressible description.

The combinatorial variety of carbon compounds is another powerful virtue. The number of possible amino acid or nucleotide sequences is hyperastronomically large. The important point is that the notion of a general-purpose system takes on a new guise. Conventional electronic machines are constructed from simple standard building blocks—for example, NAND gates. Biological systems, in contrast, are built from an extremely large variety of macromolecular species, each capable of performing a specific complex transform. Cells and organisms with different input-output behaviors arise through adaptive processes that modify the proteins in the repertoire or that express these proteins in different combinations.

The high evolvability of proteins is requisite for the efficacy of the adaptation process. Again, folding is the key feature, because it allows for structure-function malleability. As noted in section 1.3, there is an intimate connection between evolvability and complexity. If protein folding could be described by an extremely compressed program, therefore a simple process from the algorithmic complexity point of view, then the structure-function relations would approach programmability and would be fragile. Most mutations would be cataclysmic. Evolutionary considerations thus imply that folding and (chemical) complex formation are complex processes in the algorithmic sense. At the same time, the introduction of redundant amino acids in the sequence and the utilization of amino acids with high replaceability serve to buffer the effect of mutation on conformational features critical for function (Conrad and Volkenstein 1981).

Sometimes the argument is put forward that biological molecules are insufficiently reliable for computing. The opposite is actually the case. Single molecules have definite ground states, as opposed to the macroscopic switches from which conventional computers are built. The latter are built from statistical aggregates of particles and are therefore subject to erosion. The reliability issue is rather subtle, because it is clear that with solid-state components, it is possible to perform many repetitive operations and to do so rapidly. But if we want to build a reliable information processing system out of nonlinear base components, the capability for reproducing the nonlinearity in a highly precise manner is absolutely critical. This is infeasible with conventional electronic or other macroscopic components, simply because it is impossible to exactly duplicate a statistical aggregate of particles, let alone preserve their nonlinear characteristics on an operational time scale. The discrete amino acid sequences that determine the function of proteins can be precisely specified. This is sufficient, at least for a large class of sequences, to uniquely determine the folded shape and the set of available conformational states. The shape (or conformation), of course, changes when the protein interacts with its environment, but the existence of a ground state and, more generally, discrete energy levels confer precision that is unobtainable with macroscopic processing elements.

1.5 Example: Protein Solubility as a Language

As a preliminary step, let us consider a transformation that is easy to implement with macromolecules but difficult with programmable machines. Practically speaking, any ab initio calculation of the properties of even a small cluster of particles outpaces programmable computational capabilities. For the present purpose, however, we would like to consider an example of a problem that typically arises in computer science—namely, the problem of deciding whether or not a sequence of symbols belongs to a given set of sequences. Such sets are considered in formal language theory. The question is whether it is possible to construct a machine, subject to given constraints, that can recognize the language. For example, the constraint might be that the machine is a finite automaton (as are actual computers).

Consider a language L in which the elements are protein sequences that satisfy a certain property (Davidson and Sauer 1994; Prijambada et al. 1996; Yamauchi et al. 1998). The alphabet of such a language would be a set of amino acids—for instance, the twenty amino acids that are the predominant building blocks of natural proteins. We can choose solubility S in water as the property that has to be satisfied by a sequence p composed of the amino acids that constitute the alphabet (Σ). The conditions c of the process must be fixed (e.g., temperature, pressure, pH, and cosolutes; Laidler and Bunting 1973; Cacace, Landau, and Ramsden 1997). Formally, we can write

$$L = \{p \in \Sigma^* : S_c(p) > x, |p| \leq w\}$$

where L denotes the language, x is a fixed solubility threshold ($mass_{protein}/mass_{solvent}$), and we assume that length ($|p|$) of the sequence of amino acids does not exceed some constant w. The important point is that S_c is a physical and not a formal condition.

In principle, a computer of sufficient size and speed should be able to answer the question whether a given sequence p is a member of L. In practice, however, performing physics calculations to answer the membership question for the above language by implementing formal rules is not efficient. To decide the membership of a sequence in this language, the properties of the (possibly folded) amino acid sequence need to be known, thus the language encodes the protein-folding problem. Calling on calculational methods of physics to solve this problem is clearly daunting; however, it is also possible to decide the membership by actually synthesizing the protein with the sequence in question and measuring its solubility. The synthesis and measurement procedure could be automated. The resulting machine can easily decide for any particular sequence presented to it whether it belongs to L, in effect performing a computation that may well exceed the practical capabilities of presently available general-purpose machines.

1.6 Macro-Micro Interface

Language-recognition problems of the type considered above can be viewed as pattern-recognition problems. The patterns might be computer codes that have to be compiled. Or they might be objects in the world—say, chairs. If all (and only) chairs were marked with a standard printed "C," then it would be easy for a digital computer to say "yes" whenever it is presented with a chair and "no" whenever it is presented with some other object. Without such preprocessing, however, no existing computer program can do this job. The morphology of chairs is too ambiguous and variable. The required program, though it might exist, is too complex to express in a reasonably compressed way, even assuming that we knew how to write it at all. Yet humans perform this transformation with relative ease.

The protein solubility example was intended to show that molecules can be used to perform transformations that are refractory to programmable machines. But of course that example is far from using this power to address any problem of interest. To do so, the molecular level needs to be connected to the external world and the transformation needs to be adapted into a useful function.

We will return to the adaptation issue in section 1.9. Here, it is pertinent to consider the general requirements for input and output (Conrad 1984, 1990). In biological cells, the signals that represent the patterns to be recognized could come from either the internal milieu or the environment. The former case is pertinent to regulation and the latter to perception-action activities. Three levels of scale are involved: macro, meso, and micro. The signals from the environment are generally macroscopic on some dimension of scale (energy, mass, dissipation, time, space) or represent features of the world that are macroscopic. The nerve impulse, for example, is a macroscopic signal. Signals inside the cells (say, diffusion of substances) can be either macroscopic or mesoscopic. The signals constitute the milieu patterns, or context, to which proteins and other biological macromolecules respond. Because these molecules must be sufficiently large to have significant shape features (and shape dynamics), they can be classified as mesoscopic. But the nuclear coordinates couple with the electronic coordinates, so that we also have to think in unambiguously microscopic terms (Conrad 1994a). In short, we have downward flow of influence from the macro to the meso to the micro.

This downward flow is complemented by an upward flow, triggered by the response of the macromolecule or macromolecular aggregate—say, a catalytic response in the case of an enzyme or a mechanical response in the case of a contractile unit. For the present purpose, it is sufficient to think in terms of enzymes. The chemical changes produced in the milieu link the activity of different enzymes. The linking chemicals can be thought of as signals, either because they provide context or because they serve as common intermediates. The communication between the processing macromolecules is thus essentially at a mesoscopic level. Macromolecules can also communicate through direct conformational interactions, in which case the signal energies are in the micro domain. Biological cells are replete with receptors that convert signals representing macro features of the external environment to internal signals that can be brought into the web of meso- and microlevel processing.

The amount of computational work performed at the meso- and the microlevels should be as great as possible, due to the thermodynamic cost of producing macroscopic signals. Enzymes, as catalysts, are thermodynamically reversible; their pattern-recognition work is free, driven only by the heat bath. The dissipation in a typical biochemical reaction can range from 10 to 100 kT. A nerve impulse might cost 10^5 to 10^{10} kT, depending on the size of the neuron. To the extent that processing is kept as close as possible to the microlevel, the amount of information processing obtainable is vastly enhanced.

Macro-micro communication links are essential for any computational system that utilizes the activity of individual molecules, as opposed to systems that employ only statistical aggregates of particles. The signal processing activities of the medium can itself have significant nonlinear dynamics (see chapters 3 and 4 of this volume). The whole medium, not just the controlling macromolecules, can then contribute to the input-output transform. But the controlling macromolecular components are critical, because the recognition-action events would otherwise be slow and difficult to mold for different functionalities. The addition of new signal substances and macromolecular species to the medium need not and in general does not yield an additive response. This nonlinear component interaction is where the potential for performing powerful context-sensitive transforms resides.

1.7 Prototype System

Recall (from section 1.4) that protein molecules are flexible chains of amino acids. Many sequences will curl up into a compact three-dimensional shape (cf., e.g., White, Handler, and Smith 1968; Stryer 1988). The folded shape is stabilized by electrostatic interactions among its atoms, but possesses at the same time a defined agility that enables it to assume numerous conformational states. Under given physiological conditions, a subset of these states is favored (Frauenfelder, Park, and Young 1988; Freire 1998). A change in physiochemical context can induce a switch to a different favored state. This prevalent protein behavior has two points of significance for novel information processing devices. The first is that proteins have substantial freedom to select the specific stimuli to which they respond and to associate these with a response in an essentially arbitrary way. The intricate conformational dynamics constitutes the second point, because this allows the protein to fuse information in a complex nonlinear fashion that would require large numbers of conventional components to duplicate.

The nonlinear conformational dynamics harbors the computational resource we seek to exploit but at the same time precludes direct engineering of a prototype system. An alternating sequence of exploratory and selective steps can be used instead to sculpt desired functionality. In general, there are three levels open to exploration: the coding of the input signals, the amino acid sequence and operational conditions that control the protein's capacity to fuse input signals, and the choice and interpretation of the output (figure 1.1). The output could, for example, be mediated by fluorescence probes attached to the protein. If the protein is an enzyme, however, its catalytic activity is most often critically dependent on conformational state and therefore provides a sensitive probe for conformation change. Changes in physiochemical context that alter the preferred conformational state of the enzyme will hence modulate the speed of the reaction catalyzed by the enzyme.

Enzymes that catalyze reactions involving NAD (nicotinamide adenine dinucleotide) are particularly convenient in this regard, because the oxidized form and the

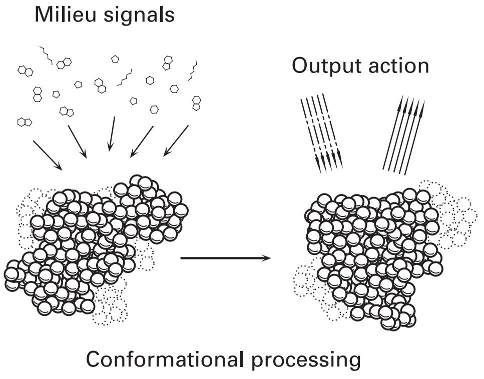

Figure 1.1
Schematic illustration of signal fusion mediated by conformational dynamics.

reduced form of NAD have quite different absorbance in the ultraviolet (UV) range. Changes in the concentration of NADH can therefore be observed with little effort by a spectrophotometer.

We used an easy-to-tend enzyme, malate dehydrogenase (MDH), which participates in the citric-acid cycle and is widely available. MDH catalyzes the oxidation of malate to oxalacetate while reducing NAD^+ to NADH. For our purposes, we can view MDH as an implementation of a function that takes selected features of its physiochemical milieu as arguments and maps these into absorbance values. Different compositions of the reaction milieu are thereby grouped by MDH into classes of UV absorbance levels (Zauner and Conrad 2000). The aim is to associate input signals with milieu features in a way that results in a useful classification.

The number of potential milieu factors that could conceivably be used to encode input signals is virtually boundless and of course not limited to chemicals of known physiological significance. Only in exceptional cases can mechanistic kinetic models predict the outcome of a specific signal encoding. Furthermore, the cases where mechanistic models apply are likely to be of limited interest from a computational point of view, because the possibility of formulating such models indicates the realm of low-complexity behavior. Instead, empirical models of factor interactions mediated by the protein are employed to discover signal encodings that yield interesting response characteristics.

Sampling the protein's performance under different milieu conditions allows for the construction of a response surface for a small number of the potentially operative factors (Box and Draper 1987; Cornell 1990). Figure 1.2 shows such a response surface for MDH with respect to changes in the $MgCl_2$ and $CaCl_2$ concentration.

Table 1.2
Exclusive-or logic function

Input 1	0	1	0	1
Input 2	0	0	1	1
Output	0	1	1	0

The response surface, once established, can be used to analyze various signal encodings. Different encoding schemes are evaluated according to a performance measure. For pattern classification tasks, the minimum difference in the response to signal patterns that should be grouped into separate classes can serve as the performance measure, to be referred to as signal strength. Only encodings yielding a positive signal strength allow for the implementation of the desired function; in general, an encoding that maximizes signal strength is advantageous.

As a concrete example, consider the exclusive-or (XOR) operation (table 1.2). This can be viewed as a simple arithmetic operation adding two bits without carry. It is also the simplest pattern classification problem that is not linearly separable. For this reason, it is used as a benchmark for learning in natural and artificial systems (Griffith et al. 1968; Minsky and Papert 1969; Ellacott and Bose 1996). The XOR operation groups patterns into one output category when both input signals are the same and into another when the signals are different. The signal strength Δs for the XOR operation can therefore be expressed as

$$\Delta s = \mathrm{Min}[r(01), r(10)] - \mathrm{Max}[r(00), r(11)]$$

where the function r denotes the response to the signal pattern (e.g., $00, 01, \ldots,$ etc.).

With this performance measure, we can ask which signal encoding best adapts the enzymatic system to the desired input-output behavior—here the XOR operation. The empirical response surface shown in figure 1.2 is used as the response function r. The question is how much $MgCl_2$ and $CaCl_2$ should be used for the input signals to maximize the signal strength Δs. Several encoding methods are possible. For example, $MgCl_2$ can be used as the signal carrier on one input line and $CaCl_2$ as carrier for the other input line. The XOR operation, however, is commutative and hence there is no need to encode the signals arriving from different input lines by different carrier substances. It is therefore possible, for example, to encode 1-signals independent of the input line by a mixture of $MgCl_2$ and $CaCl_2$ and 0-signals by a different mixture or the absence of ions. For encodings that use the same carrier substance for both input lines, only signal encodings up to half the concentration range covered by the response surface can be evaluated, because the carrier substances are additive

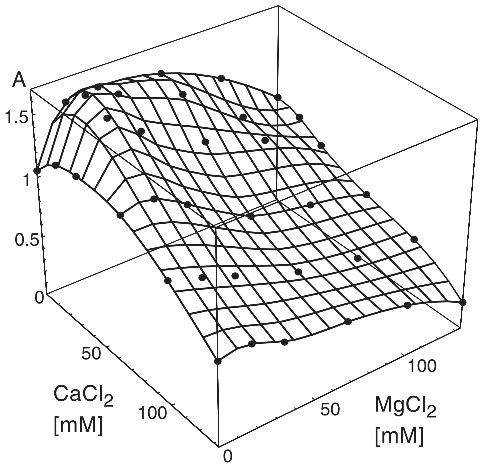

Figure 1.2
Empirical response surface of MDH with respect to $CaCl_2$ and $MgCl_2$. The dots are at concentrations where measurements were made. The surface is obtained by interpolation. (Reprinted with permission from *Biotechnol. Prog.* 2001, 17, 553–559. © 2001 American Chemical Society/AIChE.)

with respect to their contribution to the reaction milieu. Signal strengths for different encoding methods are shown in figure 1.3 as functions of the $MgCl_2$ and $CaCl_2$ concentrations used to represent the signals.

The areas of positive signal strength in figure 1.3 suggest that an enzymatic XOR based on MDH is feasible. To realize such a device, and more generally to explore enzymes as active components for the implementation of pattern classifiers, we constructed the experimental setup shown in figure 1.4. Small piston pumps, each composed of a 3 cm^3 syringe and two one-way valves, deliver input signals from reservoirs to a mixing chamber. The two signal solutions, one representing 0-signals and the other 1-signals, contain the same amount of L-malate, a substrate in the reaction catalyzed by MDH. In addition, the solution representing the 1-signal contains $MgCl_2$, while 0-signals are represented by the absence of $MgCl_2$. By injecting a defined amount of MDH/NAD^+ solution into the mixing chamber, a reaction is initiated. The reaction progresses while the mixture is pumped to a spectrophotometer and the absorbance of the NADH produced during the transit time is recorded as the output response.

Figure 1.5 illustrates the details of an improved version of the prototype in which the spectrophotometer cuvette (Cv) serves as the mixing chamber, thus permitting shorter response times and increased reliability. The injection of the enzyme solution

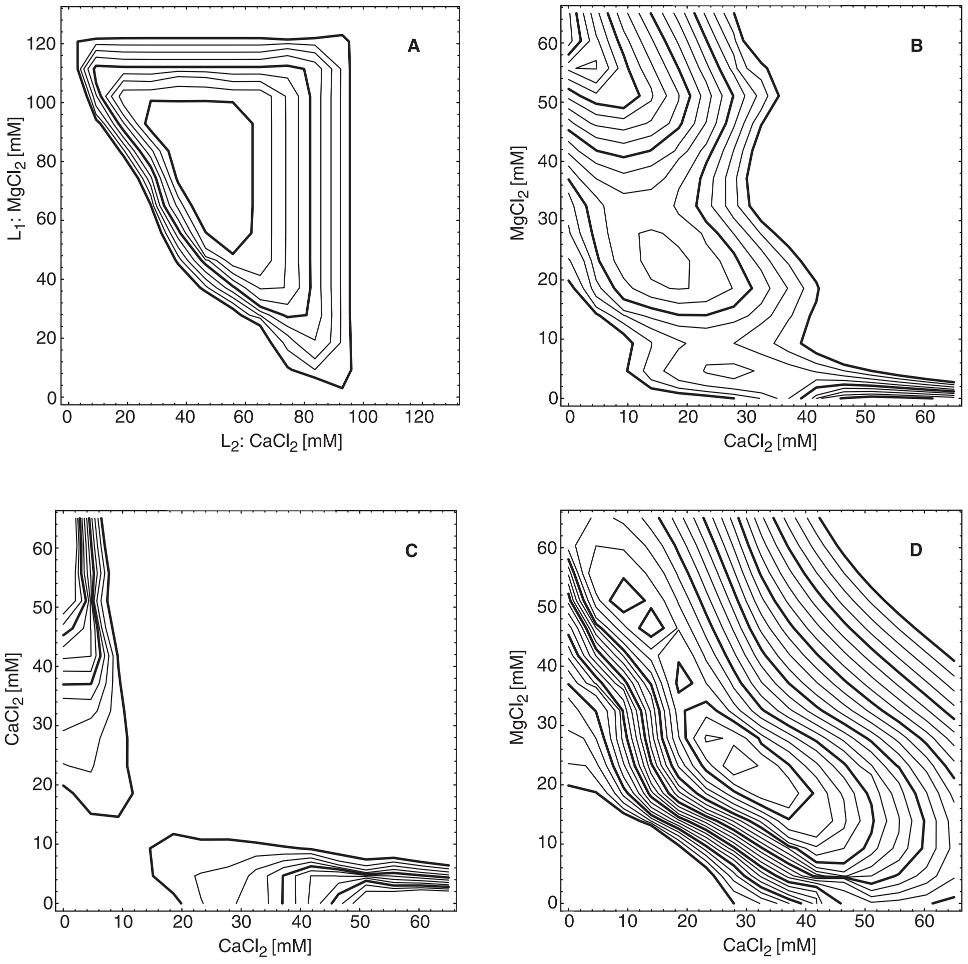

Figure 1.3
Signal strengths for the XOR operation under different signal encoding schemes. The contour lines indicate areas of positive signal strengths, therefore concentrations that make the XOR feasible. Bold contour lines indicate an increase in signal strength of 0.1, the outermost line being 0. (*A*) Input line 1 releases $MgCl_2$ when a 1-signal arrives on this line. Input line 2 releases $CaCl_2$ under the same condition. When the input is 0 no ions are released. Encoding the input lines by different signal substances makes it possible to utilize the whole concentration range of the response surface. (*B*) Here both signal lines are encoded the same way, with $MgCl_2$ representing the 1-signal and $CaCl_2$ representing the 0-signal. (*C*) Input lines 1 and 2 have the same encoding. The 0- and 1-signals are both encoded with $CaCl_2$ concentrations that consequently must be different in order to obtain a positive signal strength. The symmetry of the graph reflects the symmetry of the XOR operation with respect to negation of the input signals (cf. table 1.2). (*D*) In this case the 1-signal is encoded by a mixture of $MgCl_2$ and $CaCl_2$ for both signal lines. The 0-signal is encoded by the absence of these ions. (Reprinted in part with permission from *Biotechnol. Prog.* 2001, 17, 553–559. © 2001 American Chemical Society/AIChE.)

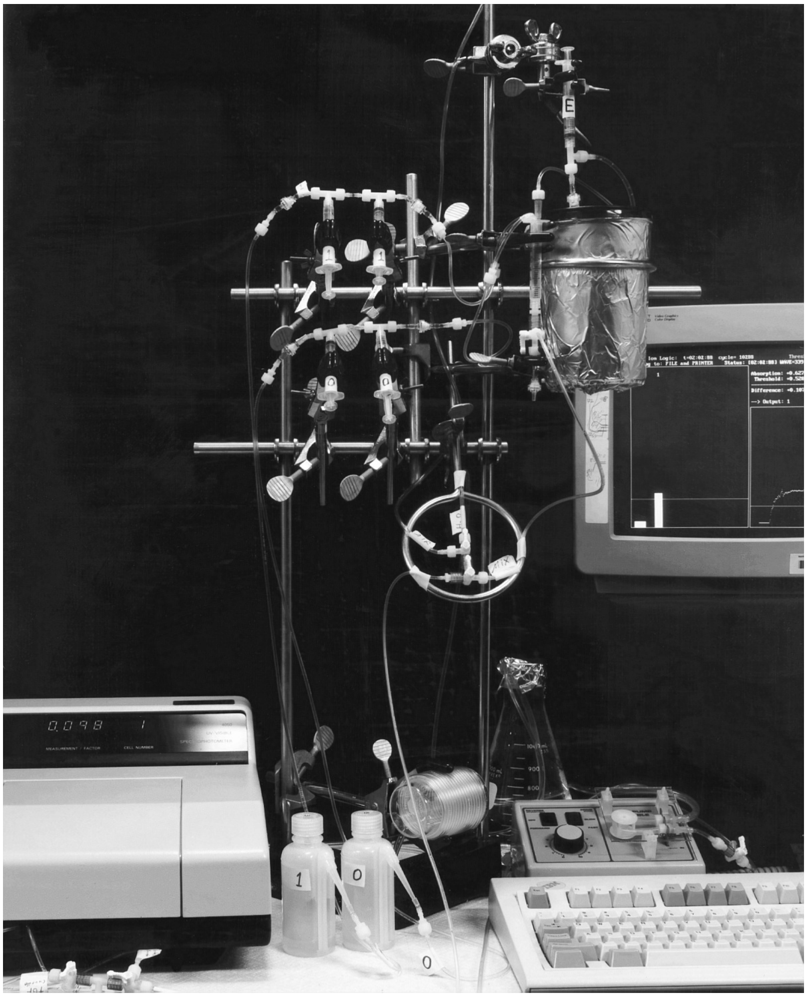

Figure 1.4
Experimental setup for first version of the tabletop XOR module. (© 2001 Zauner.)

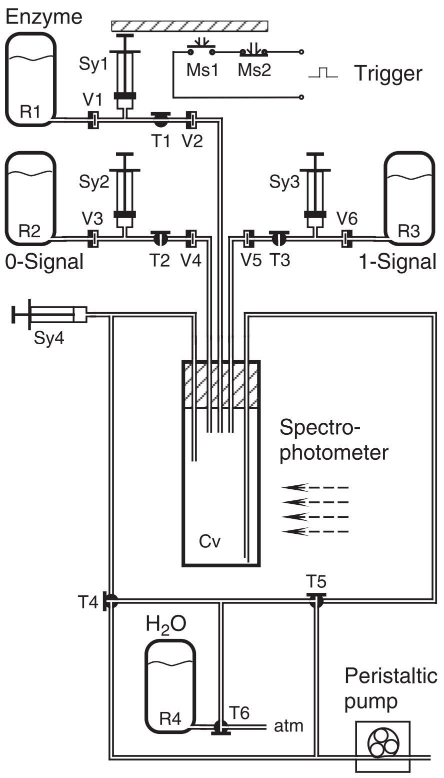

Figure 1.5
Flow diagram for direct injection version of the XOR module. Figure 1.4 shows an earlier version utilizing a mixing chamber separate from the cuvette. (Reprinted with permission from *Biotechnol. Prog.* 2001, 17, 553–559. © 2001 American Chemical Society/AIChE.)

(R1/Sy1) activates microswitches (Ms1, Ms2) that provide a trigger signal for the timing of the measurement used as the output response. A syringe (Sy4) takes up the air displaced when the cuvette (Cv) is filled. Several T-valves (T4–T6), a water reservoir (R4) and a peristaltic pump serve to clear the system between consecutive signal-processing cycles.

The XOR was also implemented with the improved setup (figure 1.5). The device was required to classify 135 consecutively presented 2-bit input patterns. The response time (i.e., the time period from injecting the enzyme/NAD solution until the output measurement is taken) was set to 10 sec. All 135 input patterns gave rise to response levels that permit correct classification by a single thresholding operation (figure 1.6). The choice of 10 sec is due to the limits of our tabletop instrumentation,

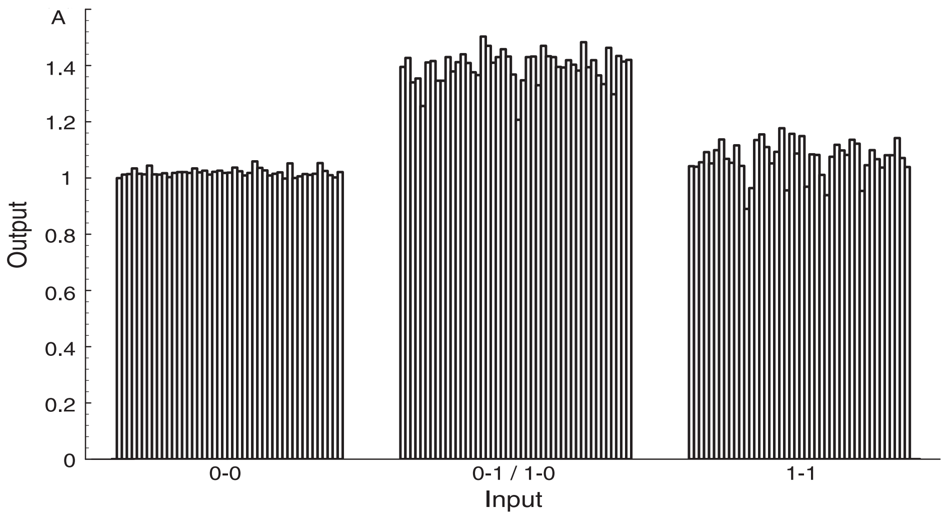

Figure 1.6
Experimental run illustrating repeated operation of the XOR module. The absorbance output separates the 01/10 inputs from the 00 and 11 inputs.

not to the underlying process. The prototype demonstrates that enzymes can be used to transform pattern classifications that are not linearly separable into simpler (linearly separable) problems. Of more importance, it points to the feasibility of developing novel computational systems that operate on the basis of high-complexity conformational processors.

1.8 Multienzyme Response Surfaces: A Simulated Example

The XOR demonstration points to the possibility of using networks of enzymes to create computationally richer response surfaces. This would only be of interest if the response of the individual components of the network interact in a nonlinear fashion. Placing multiple enzyme species in a common milieu can then lead to a response surface that is quite different from the summation of the surfaces yielded by the enzymes taken in isolation.

We have developed a software simulation tool to investigate the interaction of conformational, kinetic (reaction-diffusion), structural, and dynamic (force) interactions of protein networks in three-dimensional space that for the present purposes can be used to illustrate this nonadditivity (Zauner 1996; Zauner and Conrad 1997).

Recipe

Materials

UV-spectrophotometer ($\lambda = 339$ nm); analytic scale; adjustable micropipettes (200 µl, 1 ml); pH meter; timer.

Malate dehydrogenase from porcine heart, as ammonium sulfate suspension (store refrigerated); NAD^+ (oxidized β-nicotinamide adenine dinucleotide), as free acid (store refrigerated or frozen); l-malic acid, as free acid; $MgCl_2$ as magnesium chloride hexahydrate ($MgCl_2 \cdot 6H_2O$); MOPS (3-[N-morpholino]propanesulfonic acid); glycine (aminoacetic acid), as free acid; 10 N HCl and 10 N NaOH (for pH adjustment); pure (distilled) H_2O. (Below, "water" always refers to pure H_2O.)

Method

1. Basis Solution for signals (120 mM glycine, 7.5 mM l-malic acid, 1 l): Dissolve 9 g glycine in about 950 ml water. Add 1 g l-malic acid and allow to dissolve while stirring. Adjust to pH 10.5 with 10 N NaOH. Fill with water to a final volume of 1000 ml.

2. $MgCl_2$ Solution (4 M $MgCl_2$, 50 ml): Dissolve 40.66 g $MgCl_2 \cdot 6H_2O$ in 15 ml hot water. Let the solution cool to room temperature. Fill with water to 50 ml.

3. Signal solutions: Add 5 ml of water to 100 ml of the signal basis solution (1). The resulting solution is used for 0-signals. Add 5 ml of the $MgCl_2$ solution (2) to 100 ml of the signal basis solution (1). The resulting solution is used for 1-signals.

4. Enzyme solution (MDH/NAD^+, 10 ml): Dissolve 20.93 g MOPS in about 300 ml water, then fill up to 475 ml. Adjust pH to 7.4 with 10 N NaOH. Fill up with water to a final volume of 500 ml. This is the 0.2 M MOPS buffer. Weigh 36 mg of NAD^+ into a test tube that can hold 10 ml fluid and is wide enough to access with the 1 ml micropipette. Add 10 ml of the 0.2 M MOPS buffer and shake to dissolve the NAD^+. Add about 20 µl malate dehydrogenase suspension and shake. If the response time for the signal processing is found to be too slow, more of the enzyme suspension can be added to the solution.

5. The volume of the signal solutions and the reaction solution may need to be adjusted for the particular spectrophotometer used. The minimum volume required to cover the beam path can be determined by marking the beam at $\lambda \approx 540$ nm on a white piece of paper fixed to the cuvette. If this volume is larger than 2.1 ml, the volume for the signals and the enzyme solution (6 and 8) should be adjusted proportionally.

6. The input signal pattern is composed of two 0.8 ml portions taken in any combination from the two signal solutions (3). The signal solutions are pipetted into a cuvette.

7. Set the spectrophotometer to continuously record absorbance at $\lambda = 339$ nm.

8. To start the processing, pipette 0.5 ml of the enzyme solution (4) into the cuvette containing the signal solutions (6). A timer is started and the cuvette content is mixed (e.g., by inverting the sealed cuvette or by stirring when the enzyme solution is added).

9. Record the progress of the reaction for various combinations of the input signals by repeating steps 6 through 8. Choose a response time that will separate 00 and 11 input patterns from 01 and 10 inputs and determine the threshold level from the corresponding absorbance values.

10. Signals can now be processed using the time and threshold determined in the calibration step (9).

Note: The above protocol can serve as a starting point to explore other signaling substances. It is quite robust and could easily be adapted (e.g., replacing the micropipettes with disposable syringes) for classroom use.

The basic concept of the simulator is as follows. The simulation space, a three-dimensional lattice, contains two classes of components: macrocomponents and microcomponents. The former represent proteins, and the latter represent milieu substances—that is, metabolites on which the proteins act catalytically, as well as control molecules and ions that trigger conformational changes. The microcomponents are represented by the integer number present in each unit cell. Each catalytic or diffusional event is associated with an integer increment or decrement of this number.

The macrocomponents are represented in the simulation space by dodecahedra, each consisting of up to twelve coupled finite-state automata that model active protein domains. Recognition, binding, control, and catalytic properties are assigned to the states of these domains. The state transitions of the domains correspond to conformational changes. Transition probabilities depend on the local milieu, and therefore on the microcomponents present in the location of the dodecahedra and on adjacent macrocomponents. The local milieu can change through reaction (catalyzed by macrocomponents) and by diffusion. The whole system forms a loop encompassing context, conformation, and action. Milieu molecules and adjacent macrocomponents provide the context in which enzymes function. This influences conformation. Conformation controls action, including catalysis and structure formation. Catalysis and structure formation in turn control context, and so on (figure 1.7).

For illustrative purposes, we consider two toy reactions running separately and then consider the response of the combined reaction. The first reaction, catalyzed by enzyme e_1, is

$$A + B \overset{e_1}{\longleftrightarrow} C + D$$

We assume that e_1 has ten conformational states that differ in the catalytic activity that they confer. The transition probabilities and activity associated with the different states are illustrated in figure 1.8. R and S in the figure denote substances used as milieu signals. The product D is chosen as output signal. The response surface of e_1 with respect to R and S, illustrated in figure 1.9, shows that even a relatively small number of conformational states can yield a nontrivial surface.

For the second reaction, catalyzed by enzyme e_2, we take

$$A + E \overset{e_2}{\longleftrightarrow} F + 2D$$

Here we assume that e_2 has only four conformational states. As shown in the state transition diagram (figure 1.10), the enzyme is sensitive to the same two signaling substances, R and S, as e_1. The response surface is shown in figure 1.11.

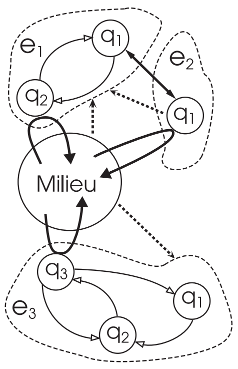

Figure 1.7
Schematic of interactions supported by the CKSD simulator (for simplicity limited to a three enzyme system). The enzymes (labeled by e_1, e_2, and e_3) have from one to three states (labeled by the q_i). States represent conformations. Arrows connecting states represent conformational transitions. These are typically influenced by the milieu components (dashed arrows) and also may be influenced by direct interactions between two enzymes (dashed arrow from e_2 to e_1). Specific conformational states catalyze milieu reactions (indicated by bent arrows). Enzymes in complementary conformational states may self-assemble to form quaternary structures (indicated by the double arrow between e_1 and e_2). Note that the transitions of distant enzymes may be coupled through their catalytic effect on the milieu.

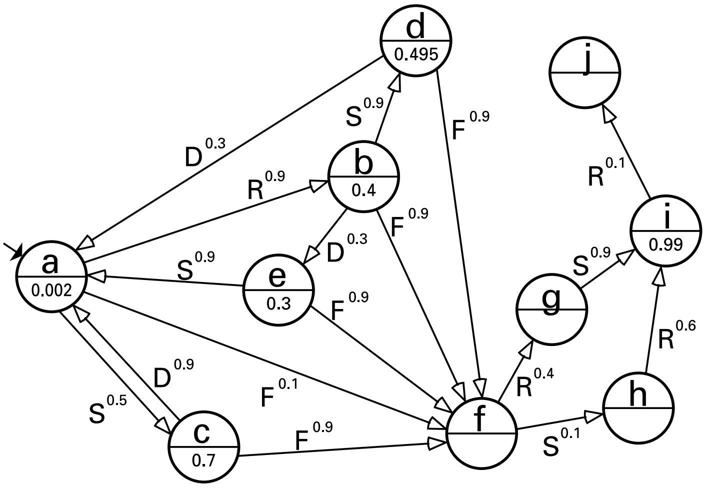

Figure 1.8
Conformational transition used to simulate enzyme e_1. The diagram is not based on any actual enzyme. The numbers below the state name indicate the relative catalytic activity of the state. Capital letters on the transitions refer to metabolites and signal molecules. The transition probabilities in the presence of these molecules is specified by superscripts.

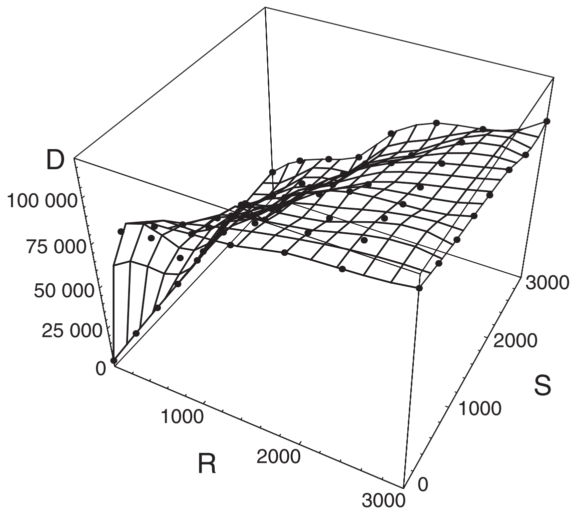

Figure 1.9
Simulated response surface for enzyme e_1 with respect to signaling substances R and S. The product D is used as the output value. The values in the diagram show the actual number of molecules present in the simulation space. The latter contained 200 e_1 enzymes distributed on a $61 \times 61 \times 21$ lattice.

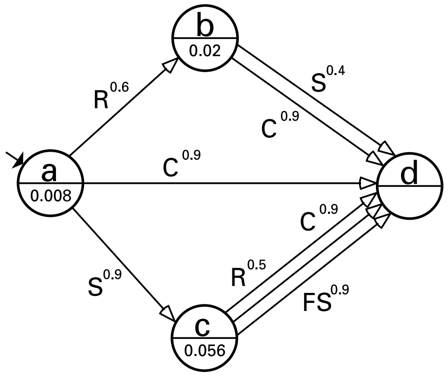

Figure 1.10
Conformational transition diagram for enzyme e_2. See caption of figure 1.8 for explanation.

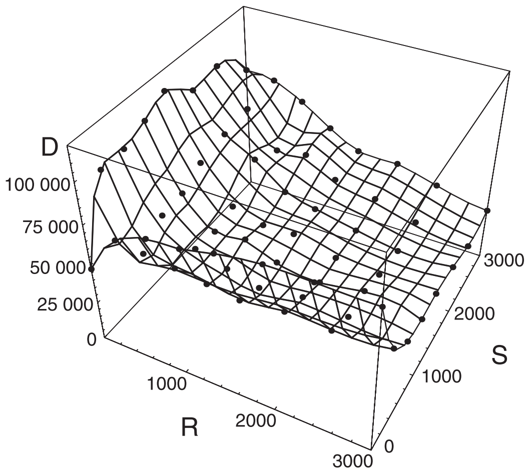

Figure 1.11
Simulated response surface for enzyme e_2. The space contained 300 e_2 enzymes; cf. figure 1.9.

Now suppose that both enzymes are introduced into the reactor. As can be seen from the reaction schemes above, e_1 and e_2 will then compete for substrate A and both will contribute to the output signal D. Furthermore, they affect each other's conformational transitions via the products C, D, and F (see figures 1.8 and 1.10). The resulting response surface is shown in figure 1.12. The response obtained by combining the enzymes cannot be easily predicted from knowledge of the response of the individual enzymes. This nonadditivity precludes the possibility of using a simpler user manual to anticipate the effect of adding components on the input-output map of the system. From our point of view, this means that it should be possible to build up molecular signal-processing modules that can implement transforms that cannot be achieved by linking the processing components in a context-independent way. The joint system self-organizes into a de novo transform.

1.9 Architectures and Adaptive Procedures

The tabletop prototype discussed in the previous section can be thought of as an extreme abstraction of the recognition-action dynamics of a biological cell. The cell is crudely pictured as a mixing chamber. The syringes roughly correspond to receptors that serve to introduce signaling substances into the chamber. The enzyme is the primary processing component, acting on the medium to trigger an output signal that could potentially control an action.

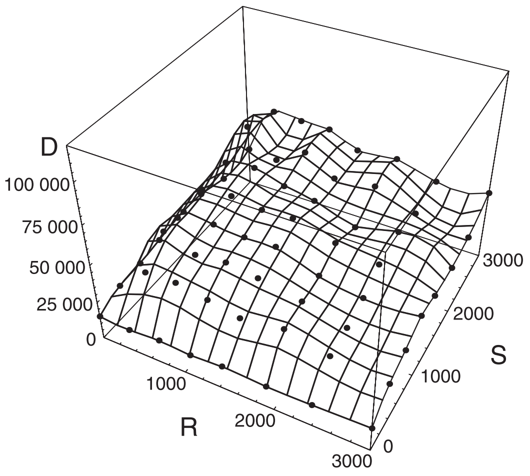

Figure 1.12
Combined response surface resulting from interaction between enzymes e_1 and e_2.

As noted above, more enzymes and signaling substances could be added. Alternative designs are possible—for example, designs with enzymes that are embedded in a matrix in an ordered way. The potential nonadditivity of the superposed response surface increases, thereby increasing the complexity of the transformation. The goal is to create a repertoire of high-complexity basis functions for implementing input-output transforms that cannot be accommodated by programmable architectures (as discussed in section 1.2).

Three issues arise: how to migrate the tabletop prototype to a chip, how to generate a useful repertoire of transformations, and how to use these chips as molecular coprocessors for a conventional architecture or to organize them into novel architectural designs.

Current advances in lab-on-a-chip technology open up a number of possible migration pathways. Figure 1.13 visualizes one of these (Zauner and Conrad 1997). This coprocessor comprises two layers; a molecular layer that contains the macromolecules and milieu components, and an optoelectronic layer that serves as the input-output interface. The molecular layer could be a sealed fluid film, gel matrix, or Langmuir-Blodgett film (Blodgett 1935). Proteins could be embedded in the film and materials moved around using microfluidic techniques (Hadd et al. 1997; Chohen et al. 1999; Unger et al. 2000). Specific molecular components are selected to couple the molecular layer to the optoelectronic layer for input and output. A pattern of light signals introduces the pattern to be classified. The induced pattern

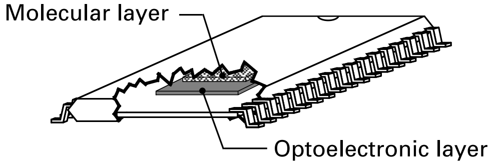

Figure 1.13
Hypothetical molecular coprocessor combining microfluidics and integrated optoelectronics. (Reproduced with permission from *Optical Memory and Neural Networks* 1997, 6: 157–173. © 1997 Allerton Press, Inc.)

of milieu features is then fused by the conformational dynamics of the embedded proteins.

The resulting conformation change produces spectroscopically identifiable signals, either directly or indirectly through catalytic change in the concentration of a light-absorbing substance. The optoelectronic layer would include integrated optics (e.g., waveguides, gratings) for coupling to the molecular layer and could incorporate integrated circuits for interfacing with a conventional electronic environment. Activities of multiple proteins in the molecular layer could be used for readout, but this depends on spectrophotometers with parallel capabilities in an appropriate wavelength range to come on-line. The choice of parameters for readout of the dynamics constitutes the interpretation.

The second issue concerns the adaptation of the physical dynamics and the interpretation. The tuning of our tabletop prototype was done by varying the substances used for coding of the inputs and essentially by ad hoc variation of the substrate concentration. A response surface was then constructed that could be used to elicit different functionalities, attention being focused in the present case on the two-variable logic functions (because only two input lines were used). The number of signal substances could be increased. The number of enzyme species included could be increased and their type varied. New macromolecular species could be evolved with specific capabilities—using, for example, protein-engineering techniques (Beaudry and Joyce 1992; Gao et al. 1997). The combinatorics clearly grows explosively, as it does in natural biological evolution. Response-surface methodology (Box and Draper 1987) can be used to prune this gigantic search space. The surfaces would be explored for features that could provide useful input-output transformations and the next steps of variation could be focused on the most interesting regions of the surface. The whole process can be automated.

The technology is available for this development program, but the evolution of suitable transforms must, of course, be a long-term, continuing process. As a first step, we envisage the development of a limited class of modules that can serve as molecular coprocessors for conventional machines. These could be used as preprocessors to transform complex input patterns into rigidly defined output patterns that can be rapidly processed by digital techniques. The conventional architecture would provide the procedural capabilities, but these would be complemented and synergized by the self-organizing dynamics of the molecular coprocessors.

As more molecular basis functions become available, it should be possible to build up an architecture with a more neuromolecular character. Artificial neural networks are essentially built up out of a set of fairly simple transforms. The situation in the brain is arguably quite different. The neuronal units exhibit a diversity of capabilities that draw on internal molecular dynamics. Complex interweavings of self-organization and procedural processes mediate what, according to our earlier considerations, are the high complexity programs that cannot be accommodated by conventional architectures.

Our group has developed a virtual system, referred to as the artificial neuromolecular (ANM) architecture, along this line (Chen 1993). The system consists of neurons controlled by an internal signal integration mechanism modeled after the neuronal cytoskeleton. Read-in elements represent molecules of the input layer in a molecular chip; readout elements correspond to molecules that trigger output firing. Neurons fire when a locus occupied by a readout element is sufficiently activated. The input-output transform performed by the neuron is adapted by varying internal parameters (read-in locations, readout locations, structure of the signal integration network) and varying the connections to other neurons. A repertoire of special-purpose transforms is thus created. Memory manipulation mechanisms that are essentially procedural in nature are then used to orchestrate the different neuron types into assemblages capable of executing yet higher complexity transforms, again using a variation-selection evolutionary technique.

The ANM architecture has been applied to a variety of 64-bit pattern-recognition problems (the input interface being currently limited in this way). These include maze navigation (Chen and Conrad 1994), Chinese character recognition (Chen and Conrad 1997), and most recently, hepatitis diagnosis (Chen 2000). The power of the system lies in its computational adaptability properties. It is a virtual system run on top of a conventional base machine. It uses the limited resources of a low-complexity machine to achieve computational adaptability, but this must be at the expense of other desirable features that programs using the same resources differently might exhibit. The molecular processing in the neurons—particularly the readout—is, of course, nominal. The readouts are just threshold elements. It would be too computationally costly to simulate the conformational dynamics that allows context-sensitive fusion of milieu features. The reasonable supposition is that implementing the architecture with real molecules would enormously increase the complexity of the programs that it is capable of embodying, thereby affording concomitant expansion of the problem domains that it is capable of managing.

1.10 Transformal Computing

The processing capabilities of the prototype described in this chapter are, of course, extremely modest, indeed even minimal, in comparison to the architectural projections of the previous section. It is to be regarded only as an initial step designed to concretize the conformation-driven computing concept and to demonstrate its technological feasibility at the level of what might be called macroscopic fluidics. The step to lab-on-a-chip integration can readily be seen.

The important question concerns the basic claim—namely, that the conformation-driven approach should provide access to computational processes that cannot practically fit into a conventional architecture. The term *transformal computing* is apt. How would we even recognize whether a computational system performs an operation that is refractory to digital (i.e., formal) machines?

The famous thesis of Church and Turing asserts, in its strong form, that all processes in nature can be brought into the circle of formal computation (Hofstadter 1980). This is an open question. Whether the answer is affirmative or negative is not the issue with which we are concerned here. It is the practical question that is relevant. Many examples could be cited: human aesthetic judgments, legal judgments, ethical rules (like the Golden Rule), or any decision that involves an indefinitely large number of situations. Arguably, an unambiguous description of such general decision rules by formal rules (i.e., by a program in the Turing sense) is infeasible. We here enter the realm of what was referred to above as transformal computations.

Of course, we do not expect the conformation-driven technology proposed here to perform such complex human operations either. Constructing an artificial brain that comes close to the human brain, even under the reasonable assumption that conformational processing plays a key role in the human mental process, exceeds by far any expectations that we would care to project. The proper question is: Can conformational processors perform transformations that exceed the practical capabilities of formal machines; and how could such transformations be identified?

Take as a concrete example the functioning of an assembly line. Automation is limited by the speed of visual processing and by the fact that quality control problems are often ambiguous. If conformational processors were evolved and harvested that could preprocess ambiguous patterns in a manner that made them suitable for processing by vision algorithms, it would constitute what in practice might be called a transformal computation.

By choosing to look at the benchmark XOR operation, we have a fortiori precluded the possibility of finding a transformal transformation. Our objective was to demonstrate that even a single enzyme species could do more processing than is standardly attributed to the threshold elements utilized in many current neural net models. Our working hypothesis—that we can use the conformation-driven approach to escape the practical limitations of programmable machines—is based on three considerations: the complexity arguments indicating that systems with self-organizing dynamics can perform more-complex operations than systems with programmable architectures, the technological feasibility of fabricating conformation-driven modules that utilize self-organizing dynamics, and the feasibility of using an evolutionary response surface methodology for developing a repertoire of high-complexity basis transforms that can be embedded in or conjoined with higher level architectures. This is a three-point landing on theory, technology, and architecture. The pieces are present; bringing them together should yield computational capabilities complementary to and synergistic with digital capabilities.

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under grant numbers ECS-9704190 and CCR-9610054, and by NASA under grant number NCC2-1189.

References

Ashby, W. R. 1968. Some consequences of Bremermann's limit for information-processing systems. In *Cybernetic Problems in Bionics*, ed. H. L. Oestreicher and D. R. Moore, 69–76. New York: Gordon and Breach.

Beaudry, A. A., and G. F. Joyce. 1992. Directed evolution of an RNA enzyme. *Science* 257: 635–641.

Blodgett, K. B. 1935. Films built by depositing successive monomolecular layers on a solid surface. *J. Am. Chem. Soc.* 57: 1007–1022.

Box, G. E. P., and N. R. Draper. 1987. *Empirical Model-Building and Response Surfaces*. New York: John Wiley and Sons.

Cacace, M. G., E. M. Landau, and J. J. Ramsden. 1997. The Hofmeister series: Salt and solvent effects on interfacial phenomena. *Q. Rev. Biophys.* 30: 241–278.

Chaitin, G. J. 1966. On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Mach.* 13: 547–569.

Chaitin, G. J. 1974. Information-theoretic computational complexity. *IEEE Trans. Inf. Theor.* 20: 10–15.

Chen, J.-C. 1993. Computer Experiments on Evolutionary Learning in a Multilevel Neuromolecular Architecture. Ph.D. thesis, Wayne State University, Detroit.

Chen, J.-C. 2000. Data differentiation and parameter analysis of a chronic heptatitis B database with an artificial neuromolecular system. *BioSystems* 57: 23–36.

Chen, J.-C., and M. Conrad. 1994. Learning synergy in a multilevel neuronal architecture. *BioSystems* 32: 111–142.

Chen, J.-C., and M. Conrad. 1997. Evolutionary learning with a neuromolecular architecture: A biologically motivated approach to computational adaptability. *Soft Computing* 1: 19–34.

Chohen, C. B., E. Chin-Dixon, S. Jeong, and T. T. Nikiforov. 1999. A microchip-based enzyme assay for protein kinase A. *Anal. Biochem.* 273: 89–97.

Conrad, M. 1979. Mutation-absorption model of the enzyme. *Bull. Math. Biol.* 41: 387–405.

Conrad, M. 1983. *Adaptability: The Significance of Variability from Molecule to Ecosystem*. New York: Plenum Press.

Conrad, M. 1984. Microscopic-macroscopic interface in biological information processing. *BioSystems* 16: 345–363.

Conrad, M. 1990. Molecular computing. In *Advances in Computers*. Vol. 31, ed. M. C. Yovits, 235–324. San Diego: Academic Press.

Conrad, M. 1994a. Amplification of superpositional effects through electronic-conformational interactions. *Chaos, Solitons, and Fractals* 4: 423–438.

Conrad, M. 1994b. The fitness of carbon for computing. In *Molecular and Biomolecular Electronics*, ed. R. R. Birge, 43–62. Washington, D.C.: American Chemical Society.

Conrad, M. 1995. Scaling of efficiency in programmable and nonprogrammable systems. *BioSystems* 35: 161–166.

Conrad, M., and M. Volkenstein. 1981. Replaceability of amino acids and the self-facilitation of evolution. *J. Theor. Biol.* 92: 293–299.

Cornell, J. A. 1990. *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data*. New York: John Wiley and Sons.

Davidson, A. R., and R. T. Sauer. 1994. Folded proteins occur frequently in libraries of random amino acid sequences. *Proc. Natl. Acad. Sci. USA* 91: 2146–2150.

Edsall, J. T., and J. Wyman. 1958. *Biophysical Chemistry*. New York: Academic Press.

Ellacott, S., and D. Bose. 1996. *Neural Networks: Deterministic Methods of Analysis*. London: International Thomson Computer Press.

Fischer, E. 1894. Einfluss der Configuration auf die Wirkung der Enzyme. *Berichte der Deutschen Chemischen Gesellschaft* (since 1947: *Chemische Berichte*) 27-3: 2985–2993 (in German).

Frauenfelder, H., F. Park, and R. D. Young. 1988. Conformational substates in proteins. *Annu. Rev. Biophys. Biophys. Chem.* 17: 451–479.

Freire, E. 1998. Statistical thermodynamic linkage between conformational and binding equilibria. *Adv. Prot. Chem.* 51: 255–279.

Gao, C., C.-H. Lin, C.-H. L. Lo, S. Mao, P. Wirsching, R. A. Lerner, and K. D. Janda. 1997. Making chemistry selectable by linking it to infectivity. *Proc. Natl. Acad. Sci. USA* 94: 11777–11782.

Griffith, V. V., J. A. Davis, and R. H. Kause. 1968. Learning of the exclusive-or logic function in rats. In *Cybernetic Problems in Bionics*, ed. H. L. Oestreicher and D. R. Moore, 587–595. New York: Gordon and Breach.

Hadd, A. G., D. E. Raymond, J. W. Halliwell, S. C. Jacobson, and J. M. Ramsey. 1997. Microchip device for performing enzyme assays. *Anal. Chem.* 69: 3407–3412.

Henderson, L. J. 1913. *The Fitness of the Environment*. New York: Macmillan.

Hofstadter, D. 1980. *Gödel, Escher, Bach: An Eternal Golden Braid*. New York: Vintage Books.

Kampis, G. 1991. *Self-Modifying Systems in Biology and Cognitive Science*, chap. 6, 278–343. Oxford: Pergamon Press.

Laidler, K. J., and P. S. Bunting. 1973. *The Chemical Kinetics of Enzyme Action*. 2d ed. Oxford: Clarendon Press.

Li, M., and P. Vitányi. 1997. *An Introduction to Kolmogorov Complexity and its Applications*. 2d ed. New York: Springer.

Minsky, M. L., and S. Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. Cambridge: MIT Press.

Prijambada, I. D., T. Yomo, F. Tanaka, T. Kawama, K. Yamamoto, A. Hasegawa, Y. Shima, S. Negoro, and I. Urabe. 1996. Solubility of artificial proteins with random sequences. *FEBS Lett.* 382: 21–25.

Sidgwick, N. V. 1950. *The Chemical Elements and Their Compounds*. Vol. 2. New York: Oxford University Press.

Stryer, L. 1988. *Biochemistry*. New York: W. H. Freeman.

Unger, M. A., H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake. 2000. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science* 288: 113–116.

von Neumann, J. 1951. The general and logical theory of automata. In *Cerebral Mechanisms in Behaviour—The Hixon Symposium*. New York: John Wiley. (Reprinted in *J. von Neumann, Collected Works*, ed. A. H. Taub. Vol. 5, 288–328. New York: Pergamon Press, 1963.)

White, A., P. Handler, and E. L. Smith. 1968. *Principles of Biochemistry*. New York: McGraw-Hill.

Yamauchi, A., T. Yomo, F. Tanaka, I. D. Prijambada, S. Ohhashi, K. Yamamoto, Y. Shima, K. Ogasahara, K. Yutani, M. Kataoka, and I. Urabe. 1998. Characterization of soluble artificial proteins with random sequences. *FEBS Lett.* 421: 147–151.

Zauner, K.-P. 1996. Simulation system for studying spatially structured biochemical interactions. Master's thesis, Wayne State University.

Zauner, K.-P., and M. Conrad. 1996. Parallel computing with DNA: Toward the anti-universal machine. In *Parallel Problem Solving from Nature: PPSN IV*. Vol. 1141 of Lecture Notes in Computer Science, ed. H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, 696–705. Berlin: Springer-Verlag.

Zauner, K.-P., and M. Conrad. 1997. Conformation-driven molecular computing: The optical connection. *Opt. Mem. Neural Netw.* 6: 157–173.

Zauner, K.-P., and M. Conrad. 2000. Enzymatic pattern processing. *Naturwissenschaften* 87: 360–362.

Zauner, K.-P., and M. Conrad. 2001. Molecular approach to informal computing. *Soft Computing* 5: 39–44.